

Embedded Controllers Databook



EMBEDDED CONTROLLERS DATABOOK

1992 Edition

COP400 Family

COP800 Family

COPS Applications

HPC™ Family

HPC Applications

**MICROWIRE™ and MICROWIRE/PLUS™
Peripherals**

Microcontroller Development Support

Appendices/Physical Dimensions

1

2

3

4

5

6

7

8

TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

ABIC TM	FACT TM	MICROWIRE/PLUS TM	SCX TM
Abuseable TM	FACT Quiet Series TM	MOLE TM	SERIES/800 TM
Anadig TM	FAIRCAD TM	MPA TM	Series 900 TM
ANS-R-TRAN TM	Fairtech TM	MST TM	Series 3000 TM
APPST TM	FAST [®]	Naked-8 TM	Series 32000 [®]
ASPECT TM	FAST ^r TM	National [®]	Shell ^r /Chek TM
Auto-Chem Deflasher TM	5-Star Service TM	National Semiconductor [®]	Simple Switcher TM
BCP TM	Flash TM	National Semiconductor Corp. [®]	SofChek TM
BI-FET TM	GENIX TM	NAX 800 TM	SONIC TM
BI-FET II TM	GNX TM	Nitride Plus TM	SPIRE TM
BI-LINE TM	GTO TM	Nitride Plus Oxide TM	Staggered Refresh TM
BIPLAN TM	HAMR TM	NML TM	STAR TM
BLC TM	HandiScan TM	NOBUST TM	Starlink TM
BLX TM	HEX 3000 TM	NSC800 TM	STARPLEX TM
BMACT TM	HPCT TM	NSCISE TM	ST-NICT TM
Brite-Lite TM	HyBaI TM	NSX-16 TM	SuperATT TM
BSI TM	I ³ L [®]	NS-XC-16 TM	Super-Block TM
BSI-2 TM	ICM TM	NTERCOM TM	SuperChip TM
CDD TM	INFOCHEX TM	NURAM TM	SuperScript TM
CheckTrack TM	Integral ISE TM	OPAL TM	SYS32 TM
CIM TM	Intellisplay TM	OXISS TM	TapePak [®]
CIMBUST TM	ISE TM	P ² CMOST TM	TDST TM
CLASIC TM	ISE/06 TM	PC Master TM	TeleGate TM
Clock ^r /Chek TM	ISE/08 TM	Perfect Watch TM	The National Anthem [®]
COMBO [®]	ISE/16 TM	Pharma ^r /Chek TM	Time ^r /Chek TM
COMBO I TM	ISE32 TM	PLAN TM	TINAT TM
COMBO II TM	ISOPLANAR TM	PLANAR TM	TLC TM
COPST TM microcontrollers	ISOPLANAR-Z TM	PLAYER TM	Trapezoidal TM
CRD TM	KeyScan TM	Plus-2 TM	TRI-CODE TM
DA4 TM	LERIC TM	Polycraft TM	TRI-POLY TM
Datachecker [®]	LMCMOS TM	POSilink TM	TRI-SAFE TM
DENSPAK TM	M ² CMOS TM	POSitalker TM	TRI-STATE [®]
DIB TM	Macrobus TM	Power + Control TM	TROPIC TM
DISCERN TM	Macrocomponent TM	POWERplanar TM	TURBOTRANSCEIVER TM
DISTILL TM	MAPL TM	QUAD3000 TM	VIPT TM
DNR [®]	MAXI-ROM [®]	QUICKLOOK TM	VR32 TM
DPVM TM	Meat ^r /Chek TM	RAT TM	WATCHDOG TM
E ² CMOST TM	MenuMaster TM	RICT TM	XMOST TM
ELSTAR TM	Microbus TM data bus	RTX16 TM	XPU TM
Embedded System Processor TM	MICRO-DAC TM	SABR TM	Z STAR TM
EPT TM	μtalker TM	SCAN TM	883B/RETST TM
E-Z-LINK TM	Microtalker TM	Script ^r /Chek TM	883S/RETST TM
	MICROWIRE TM		

ABELTM is a trademark of Data I/O Corporation.

CP/TM is a trademark of Digital Research Corporation.

DECTM, VAXTM and VMSTM are trademarks of Digital Equipment Corporation.

IBM[®], PC[®], PC-AT[®] and PC-XT[®] are registered trademarks of International Business Machines Corporation.

iceMASTERTM is a trademark of MetaLink Corporation.

MicrosoftTM and MS-DOSTM are trademarks of Microsoft Corporation.

PAL[®] is a registered trademark of and used under license from Advanced Micro Devices, Inc.

Sun[®] is a registered trademark of Sun Microsystems.

SunOSTTM is a trademark of Sun Microsystems.

TouchToneTM is a trademark of Western Electric Co., Inc.

UNIX[®] is a registered trademark of AT & T Bell Laboratories.

Z80[®] is a registered trademark of Zilog Corporation.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, California 95052-8090 1-800-272-9959 TWX (910) 339-9240

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied, and National reserves the right, at any time without notice, to change said circuitry or specifications.



Microcontroller Introduction

Practical Solutions to Real Problems

Microcontrollers have always been driven by customer need rather than technological capability.

They were designed to meet specific needs with specific performance in specific applications with specific cost.

That also meant, however, that your choices were limited to what was available on the market—which meant possibly having to compromise your design objectives because you couldn't get exactly the microcontroller you needed.

No more.

Now you can get a microcontroller from National that spans a wide range of system solutions—to go almost anywhere your design imagination takes you.

Whether you need a low-cost 4-bit workhorse or a 16-bit 30 MHz powerhouse, whether you want 1/2 kbyte of ROM or over 64 kbytes, whether you're building a simple singing greeting card or a complex telecommunications network, we have a microcontroller for the job.

With on-board CPU, memory, internal logic, and I/Os, National microcontrollers are helping more and more designers lower system costs and shrink system size.

And as technology brings more peripheral functions onto the chip, including user-programmable memory, fast SRAM, timers, UARTs, comparators, A/D converters, and LAN interfaces, the microcontroller will become the cost-efficient choice for even such real-time "microprocessor" applications as laser printers, ISDN, and digital signal processing.

That's why National continues to lead the industry in the development of microcontroller technology.

That's why we have 8-bit and 16-bit controller cores.

That's why we're scaling our common M²CMOSTTM process for submicron feature sizes, hypermegahertz frequencies, and unparalleled performance levels.

That's why we offer you "Hot-Line" applications support and a 24-hour-a-day digital information service.

That's why we offer you MS DOS-based development tools and high-level-language (C) compilers

And that's why we've committed the full resources of our company to provide you with the most complete, most reliable, most cost-effective systems solution for all your needs.

This databook is a reflection of that commitment.

It will give you an overview of microcontrollers in general and of National's microcontrollers in particular.

It will help you evaluate your microcontroller options from both a business perspective and an engineering perspective.

It will help you make reasoned judgements about selecting the best microcontroller for your needs.

And it will show you what the microcontroller future holds in store for all of us.

If you'd like more information, or you'd like to find out how to put a microcontroller to work in your own application, just contact your local National Semiconductor Sales Office.



How to Select a Microcontroller

Microcontrollers have evolved far beyond their origins as control chips in calculators.

Today, microcontrollers can be the perfect solution for simplifying a wide range of designs. And for giving those designs a clear competitive advantage in the marketplace.

Whether used for simple logic replacement or as an integral part of a high performance system, a microcontroller can reduce system costs, shrink system size, and shorten system design cycles. And yet deliver performance often superior to "traditional" digital solutions.

Still, all microcontrollers are not created equal. And it's important to consider a number of factors before committing to a particular device:

1. Is the microcontroller optimized for your specific application in terms of speed, performance, features, and cost?
2. Is it code-efficient, and based on a true microcontroller architecture for the highest performance and efficiency?

3. Is it fabricated in the most advanced CMOS process technology, and is it fully scalable to maintain its performance edge in the future?

4. Is it supported by a comprehensive family of development tools that run on standard platforms such as the IBM-PC?

5. Is it backed by a dedicated team of professionals who are available not only to provide expert training for new users, to get them on-line quickly and efficiently, but also to provide technical guidance for even the most experienced user?

6. Is it designed for the future, with the capability of expanding on-chip functionality.

If you answered "yes" to all these questions, then you already know that there's only one company with the product depth and technology capability to provide you with a microcontroller optimized for your specific application.

National Semiconductor.

You'll find National Microcontrollers in:

Laser Printers
 Disc Controllers
 Telecommunications Systems
 Keyboards
 Airplane Multiplex Systems
 Car Radios
 Engine Control Systems
 Anti-Skid Brake Systems
 Armaments
 Factory Automation
 Medical Equipment
 Fuses
 Scales
 Refrigerators
 Security Systems
 Garage Door Openers
 Camera Aperture Controls
 Office Copiers
 Cable TV Converters
 Televisions
 Video Recorders
 Solar Heating Controls
 Thermostats
 Climate Control Systems
 Intelligent Toys
 Kitchen Timers

Why Select a National Microcontroller

National has created the most complete selection of 4-, 8-, and 16-bit microcontrollers of any company in the industry.

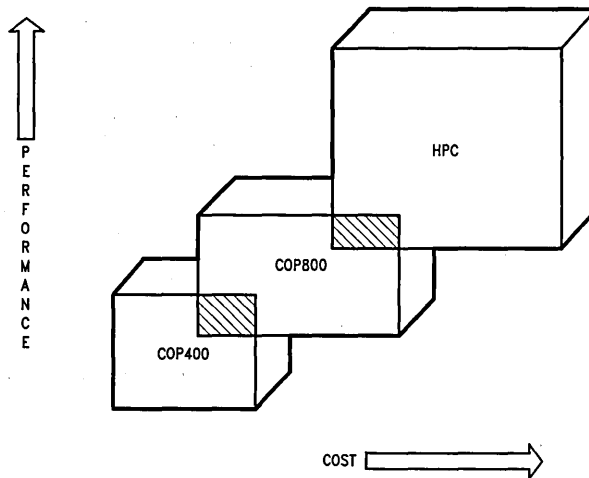
Which means that no matter what the specific needs of your application are, you can find a National microcontroller to meet them.

Our COP400 family offers the lowest-cost, 4-bit solutions for timing, counting, and control functions.

Our COP800 family offers low-cost, feature-rich, 8-bit solutions.

And our High Performance microController (HPC™) family offers the highest performance with the world's fastest 16-bit CMOS solution.

Microcontroller Family of Products



TL/XX/0071-1

With a full range of performance- and feature-options, National's microcontroller families can be customized to meet the needs of your specific application.

1.0 COMMON FEATURES FOR A CUSTOM FIT

All our microcontrollers are designed to provide not just a one-time-only solution, but a continuum of solutions to meet the changing demands of your product and the marketplace.

Our COP400 family, for example, which consists of over 60 devices, is designed with a common instruction set, so you can migrate from one member of the family to others without having to recode, so you can take efficient advantage of the application-specific flexibility of the COP400 family's programmable I/O options.

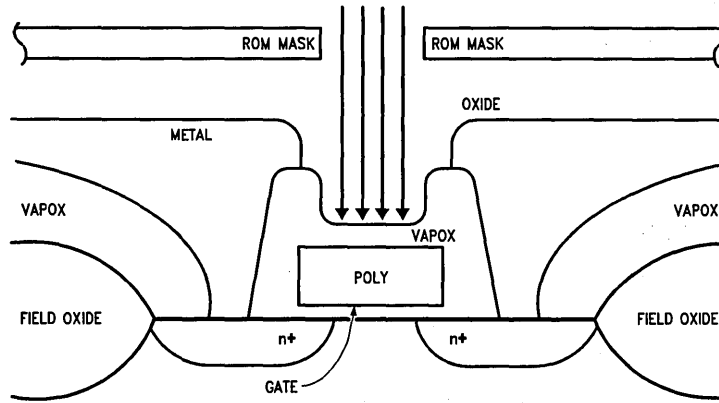
Our COP800 and HPC families, on the other hand, are each designed around a common CPU core that then can be surrounded by a variety of standard functional building blocks such as RAM, ROM, user programmable memory, fast SRAM, DMA, UART, comparator, A/D, HDLC, and I/O.

This unique core approach allows us to offer you a microcontroller with the exact combination of CPU power and peripheral function you need for your specific application. So you don't have to compromise your design parameters by using an inappropriate device, and you don't have to compromise your cost parameters by paying for performance and features you don't need.

This core concept also allows us to bring new microcontroller products to market fast and at a lower cost to help you keep pace with the rapidly changing conditions in your own market.

And it allows us to implement designs for both the COP800 and the HPC cores, for the highest levels of integration and flexibility in your own proprietary design.

COP400 Post-Metal ION Implant



TL/XX/0071-2

2.0 TRUE MICROCONTROLLER ARCHITECTURE

Our microcontrollers are designed as true controllers, not modified microprocessors.

The COP400 family is designed with a two-bus Harvard architecture; the COP800 family with a memory-mapped, modified Harvard architecture, and the HPC family with a memory-mapped, von Neumann architecture.

All three control-oriented families, however, are optimized for high code efficiency. Most instructions are only 1 byte long—yet each can typically execute several functions. This “function-dense” code provides a substantial increase in memory efficiency and processing speed.

3.0 ADVANCED PROCESS AND PACKAGING TECHNOLOGIES

National offers you not only the right microcontroller for your needs, but also the right process technology for your microcontroller.

COP400 devices are available in both high-speed NMOS and low-power CMOS fabrications, while the higher-performance COP800 and HPC families are both fabricated in National's advanced M²CMOS process.

M²CMOS. This double-metal CMOS process offers significant design advantages. It combines the speed of NMOS, the ruggedness of bipolar, and the low power consumption of bulk CMOS to produce fast, dense, highly efficient, highly scalable devices for a wide variety of integrated-circuit designs.

It's for these reasons that M²CMOS has become the standard process technology for all of National's advanced-

technology LSI and VLSI products, including microprocessors, gate arrays, standard cells, telecommunications devices, linear devices and, of course, microcontrollers.

Post-Metal Programming (PMP). This is a new process technology available from no other semiconductor manufacturer in the world. It offers the fastest, guaranteed prototype programmed-ROM turn-time in the industry.

PMP is a high-energy implantation process that allows microcontroller ROM to be programmed **after** final metallization.

This is a true innovation, because ROM is usually implemented in the second die layer, with nine or ten other layers then added on top. And that means the ROM pattern must be specified early in the production process, and completed prototype devices won't be available typically for six weeks.

With PMP, however, dice can be fully manufactured through metallization and electrical tests (only the passivation layers need to be added), and held in inventory. Which means ROM can be programmed late in the production cycle, **making prototypes available in only two weeks!**

PMP allows you to adapt to fast-changing market conditions and to take maximum advantage of narrow windows of opportunity.

And shorter production lead times can simplify your inventory control and reduce safety stock by up to 20%, giving you significant cost reductions.

Currently, Post-Metal Programming is available for selected members of the COP400 family, and will be expanded to the COP800 and HPC families in the near future.

Military versions. All National microcontrollers have CMOS parts available in the full military temperature range (-55°C to $+125^{\circ}\text{C}$).

In addition, parts are available that have been certified under MIL-STD-883, Rev. C, the most rigorous non-JAN screening flow in the electronics industry.

Packaging. One major reason that National microcontrollers demonstrate such consistently high levels of reliability is that we've developed special advanced packaging processes to protect the die.

For example, we've designed a unique leadframe with "locking holes" that helps block any penetrating moisture from reaching the die itself.

And the leadframes themselves are made of an unusual high-strength copper alloy that has a lower thermal resistance (θ_{JA}) than typical Alloy 42-leadframes.

We've also employed a unique low-stress, high-purity epoxy molding compound for our packages, which gives them a coefficient of expansion that nearly matches that of the leadframes. As a result, many of our microcontrollers are also offered in plastic packages for military-temperature-range operation.

Reliability is built-in at the die level as well. Our M²CMOS microcontrollers are fabricated on dedicated lines at our world-class, six-inch wafer-fab facility in Arlington, Texas. With its Class-10 clean rooms and automated-handling system, Arlington has set a standard of reliability equalled by few other companies in the industry.

And this reliability is available to you in a wide variety of microcontroller packages, ranging in size from 20 to 84 pins.

Package types include plastic and ceramic DIPs, small outline (S.O.) surface mounts, plastic and ceramic leaded chip carriers, and pin grid arrays.

Or, you can select the world's most advanced, high-density packaging option, TapePak™.

TapePak combines the advantages of an automated tape-and-reel-type delivery system with built-in testing pads for reliability and a unique plastic package carrier. The result is a surface-mounted package that can be as small as $\frac{1}{10}$ the size of conventional surface mounts, with lead spacings of 20 mils.

4.0 FULL DEVELOPMENT SUPPORT

Even the right microcontroller, of course, is useless without the right development tool to put that controller to work in your application.

That's why National offers you a full range of development support. Ready-to-run evaluation boards. Emulators. Software. Prototyping devices. Training and seminars for beginning and advanced users. Everything you need to take your design from concept to reality.

And you don't need an expensive, dedicated, development environment to do it. With our development systems, a standard IBM PC becomes a full-featured platform.

And with our comprehensive library of prewritten routines, from keyboard scanners to Fast Fourier Transforms, you can reduce software programming to a minimum. This "user-friendly" service can help you bring your design to market quickly and cost-effectively.

5.0 FULL APPLICATIONS SUPPORT

At National, we believe that applications support should be immediate and "hands-on".

That's why we established the unique Dial-A-Helper program.

With a computer, modem, and telephone, you can tie directly into our Microcontroller Applications Group for fast, direct assistance in developing your design.

You can leave messages on our electronic bulletin board for our Applications Engineers, who will respond to you directly. You can access applications files.

You can download those files for later reference.

Or, if you're having a real problem, you can actually turn the control of your Microcontroller On-Line Emulator development system over to our engineering staff, who can perform remote diagnostic routines to locate and eliminate any bugs.

The point is, when you buy a microcontroller from National, you're buying more than silicon—you're buying the commitment of an entire company of dedicated professionals who share a single goal: to help you put that silicon to **work**.

6.0 THE FUTURE

National's microcontrollers were designed to meet two objectives: to adapt to your evolving needs, and to adapt to evolving technology.

Both "evolutions," however, are leading to the same goal: the complete "system-on-chip" solution.

The key to achieving this goal, of course, is a common, advanced, scalable process technology.

That's why both the COP800 and HPC families are fabricated in our high-performance double-metal CMOS process. This is a highly scalable technology that can accommodate die shrinks to submicron feature sizes, increasing performance and cutting power consumption with each step.

Moreover, because M²CMOS is now the standard process technology for all new National LSI and VLSI devices, the COP800 and HPC cores are able to support one of the broadest range of functional blocks available from any semiconductor manufacturer—all aligned on the same set of design rules.

So you can standardize your designs on just one or two core processors, and, as we introduce new technologies and functions, you can maintain that design knowledge base while taking advantage of these new, higher levels of functional integration.

And because National gives you the option of using standard parts or designing with our functional blocks—both supported by common design tools and a common process—you can create highly competitive, highly secure, highly optimized solutions in minimal space at minimal cost in minimal time.

And that's the name of the game.



Product Status Definitions

Definition of Terms

Data Sheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This data sheet contains preliminary data, and supplementary data will be published at a later date. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

National Semiconductor Corporation reserves the right to make changes without further notice to any products herein to improve reliability, function or design. National does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

Table of Contents

Alphanumeric Index	xii
Section 1 COP400 Family	
COP400 Family	1-3
ROM'd Devices	
COP224C/COP225C/COP226C/COP244C/COP245C Single-Chip 1k and 2k CMOS Microcontrollers	1-8
COP410C/COP411C/COP310C/COP311C Single-Chip CMOS Microcontrollers	1-26
COP410L/COP411L/COP310L/COP311L Single-Chip N-Channel Microcontrollers ...	1-41
COP413L/COP313L Single-Chip Microcontrollers	1-59
COP413C/COP413CH/COP313C/COP313CH Single-Chip CMOS Microcontrollers ...	1-72
COP414L/COP314L Single-Chip N-Channel Microcontrollers	1-86
COP420/COP421/COP422/COP320/COP321/COP322 Single-Chip N-Channel Microcontrollers	1-101
COP420L/COP421L/COP422L/COP320L/COP321L/COP322L Single-Chip N-Channel Microcontrollers	1-125
COP424C/COP425C/COP426C/COP324C/COP325C/COP326C and COP444C/COP445C/COP344C/COP345C Single-Chip 1k and 2k CMOS Microcontrollers	1-152
COP444L/COP445L/COP344L/COP345L Single-Chip N-Channel Microcontrollers ...	1-172
ROMless Devices	
COP401L ROMless N-Channel Microcontroller	1-196
COP401L-X13/COP401L-R13 ROMless N-Channel Microcontrollers	1-210
COP402-5 ROMless N-Channel Microcontroller	1-223
COP404C ROMless CMOS Microcontroller	1-240
COP404LSN-5 ROMless N-Channel Microcontroller	1-257
COP420P/COP444CP/COP444LP Piggyback EPROM Microcontrollers	1-271
Section 2 COP800 Family	
COP800 Family	2-3
COP620C/COP622C/COP640C/COP642C/COP820C/COP822C/COP840C/ COP842C/COP920C/COP922C/COP940C/COP942C Single-Chip microCMOS Microcontrollers	2-5
COP820CJ/COP822CJ/COP823CJ Single-Chip microCMOS Microcontrollers	2-27
COP8640C/COP8642C/COP8620C/COP8622C/COP86L20C/COP86L22C/ COP86L40C/COP86L42C Single-Chip microCMOS Microcontrollers	2-53
COP680C/COP681C/COP880C/COP881C/COP980C/COP981C Microcontrollers ...	2-75
COP688CL/COP684CL/COP888CL/COP884CL/COP988CL/COP984CL Single-Chip microCMOS Microcontrollers	2-98
COP888CF/COP884CF/COP988CF/COP984CF Single-Chip microCMOS Microcontrollers	2-133
COP888CG/COP884CG Single-Chip microCMOS Microcontrollers	2-167
COP688EG/COP684EG/COP888EG/COP884EG Single-Chip microCMOS Microcontrollers	2-203
COP688CS/COP684CS/COP888CS/COP884CS/COP988CS/COP984CS Single-Chip microCMOS Microcontrollers	2-243
COP8780C/COP8781C/COP8782C Single-Chip EPROM/OTP Microcontrollers	2-283
COP842CMH Microcontroller Emulator	2-300
COP880CMH/COP881CMH Microcontroller Emulators	2-307
COP8640CMH/COP8642CMH Microcontroller Emulators	2-316
COP888CLMH Single-Chip microCMOS Microcontroller Emulator	2-325
COP888CFMH Single-Chip microCMOS Microcontroller Emulator	2-334

Table of Contents (Continued)

Section 2 COP800 Family (Continued)

COP888CGMH/COP884CGMH/COP888EGMH Single-Chip microCMOS Microcontroller Emulators	2-344
COP820CJMH/COP822CJMH Single-Chip microCMOS Microcontrollers	2-354
COP888CSMH microCMOS Microcontroller Emulator	2-363

Section 3 COPS Applications

COP Brief 2 Easy Logarithms for COP400	3-3
COP Brief 6 RAM Keep-Alive	3-14
COP Note 1 Analog to Digital Conversion Techniques with COPS Family Microcontrollers	3-15
COP Note 4 The COP444L Evaluation	3-47
COP Note 5 Oscillator Characteristics of COPS Microcontrollers	3-52
COP Note 6 Triac Control Using the COP400 Microcontroller Family	3-69
COP Note 7 Testing of COP400 Family Devices	3-77
AB-3 Current Consumption in NMOS COPS Microcontrollers	3-86
AB-4 Further Information on Testing of COPS Microcontrollers	3-88
AB-6 COPS Interrupts	3-90
AB-15 Protecting Data in Serial EEPROMs	3-91
AN-326 A User's Guide to COPS Oscillator Operation	3-93
AN-329 Implementing an 8-Bit Buffer in COPS	3-97
AN-338 Designing with the NMC9306/COP494 a Versatile Simple to Use EEPROM ...	3-101
AN-400 A Study of the Crystal Oscillator for CMOS-COPS	3-107
AN-401 Selecting Input/Output Options on COPS Microcontrollers	3-111
AN-440 New CMOS Vacuum Fluorescent Drivers Enable Three Chip System to Provide Intelligent Control of Dot Matrix V.F. Display	3-121
AN-452 MICROWIRE Serial Interface	3-131
AN-454 Automotive Multiplex Wiring	3-142
AN-521 Dual Tone Multiple Frequency (DTMF)	3-146
AN-579 MICROWIRE/PLUS Serial Interface for COP800 Family	3-155
AN-596 COP800 MathPak	3-167
AN-607 Pulse Width Modulation A/D Conversion Techniques with COP800 Family Microcontrollers	3-203
AN-662 COP800 Based Automated Security/Monitoring System	3-210
AN-663 Sound Effects for the COP800 Family	3-218
AN-666 DTMF Generation with a 3.58 MHz Crystal	3-241
AN-673 2-Way Multiplexed LCD Drive and Low Cost A/D Converter Using V/F Techniques with COP8 Microcontrollers	3-269
AN-681 PC MOUSE Implementation Using COP800	3-288
AN-714 Using COP800 Devices to Control DC Stepper Motors	3-313
AN-734 MF2 Compatible Keyboard with COP8 Microcontrollers	3-323
AN-739 RS-232C Interface with COP800	3-343
AN-749 Quadrature Signal Interface to a COP400 Microcontroller	3-355

Section 4 HPC Family

The 16-Bit HPC Family: Optimized for Performance	4-3
HPC16083/HPC26083/HPC36083/HPC46083/HPC16003/HPC26003/HPC36003/ HPC46003 High-Performance Microcontrollers	4-6
HPC36164/HPC46164/HPC36104/HPC46104 High-Performance Microcontrollers with A/D	4-39
HPC16064/HPC26064/HPC36064/HPC46064/HPC16004/HPC26004/HPC36004/ HPC46004 High-Performance Microcontrollers	4-75
HPC36400E/HPC46400E High-Performance Communications Microcontrollers	4-108

Table of Contents (Continued)

Section 4 HPC Family (Continued)

HPC167064/HPC467064 High-Performance Microcontrollers with a 16K UV Erasable CMOS EPROM	4-134
HPC46100 High-Performance Microcontroller with DSP Capability	4-165

Section 5 HPC Applications

AN-474 HPC MICROWIRE/PLUS Master-Slave Handshaking Protocol	5-3
AN-484 Interfacing Analog Audio Bandwidth Signals to the HPC	5-11
AN-485 Digital Filtering Using the HPC	5-21
AN-486 A Floating Point Package for the HPC	5-36
AN-487 A Radix 2 FFT Program for the HPC	5-89
AN-497 Expanding the HPC Address Space	5-114
AN-510 Assembly Language Programming for the HPC	5-125
AN-550 A Software Driver for the HPC Universal Peripheral Interface Port	5-130
AN-551 The HPC as a Front-End Processor	5-185
AN-552 Interfacing a Serial EEPROM to the National HPC16083	5-249
AN-561 I ² C-Bus—Interface with HPC	5-266
AN-577 Extended Memory Support for HPC	5-286
AN-585 High Performance Controller in Information Control Applications	5-330
AN-586 Pulse Width Modulation Using HPC	5-338
AN-587 C in Embedded Systems and the Microcontroller World	5-346
AN-593 HPC16400 A Communication Microcontroller with HDLC Support	5-352
AN-603 Signed Integer Arithmetic on the HPC	5-362
AN-643 EMI/RFI Board Design	5-374
AN-736 Interfacing the HPC46064 to the DP83200 FDDI Chip Set	5-391
AN-786 LCD Direct Drive Using HPC	5-397
AN-798 Improved UART Clocking Techniques on New Generation HPCs	5-413

Section 6 MICROWIRE and MICROWIRE/PLUS Peripherals

MICROWIRE and MICROWIRE/PLUS: 3-Wire Serial Interface	6-3
COP472-3 Liquid Crystal Display Controller	6-7

Section 7 Microcontroller Development Support

Development Support	7-3
COP400 Microcontroller Development Support	7-5
COP800 Development System	7-12
HPC Microcontroller Development System	7-22
HPC Software Support Package	7-35
ISDN Basic Rate Interface Software for the HPC16400 High Performance Data Communications Microcontroller	7-45

Section 8 Appendices/Physical Dimensions

Surface Mount	8-3
PLCC Packaging	8-23
Physical Dimensions	8-27
Bookshelf	
Distributors	

Alpha-Numeric Index

AB-3 Current Consumption in NMOS COPS Microcontrollers	3-86
AB-4 Further Information on Testing of COPS Microcontrollers	3-88
AB-6 COPS Interrupts	3-90
AB-15 Protecting Data in Serial EEPROMs	3-91
AN-326 A User's Guide to COPS Oscillator Operation	3-93
AN-329 Implementing an 8-Bit Buffer in COPS	3-97
AN-338 Designing with the NMC9306/COP494 a Versatile Simple to Use EEPROM	3-101
AN-400 A Study of the Crystal Oscillator for CMOS-COPS	3-107
AN-401 Selecting Input/Output Options on COPS Microcontrollers	3-111
AN-440 New CMOS Vacuum Fluorescent Drivers Enable Three Chip System to Provide Intelligent Control of Dot Matrix V.F. Display	3-121
AN-452 MICROWIRE Serial Interface	3-131
AN-454 Automotive Multiplex Wiring	3-142
AN-474 HPC MICROWIRE/PLUS Master-Slave Handshaking Protocol	5-3
AN-484 Interfacing Analog Audio Bandwidth Signals to the HPC	5-11
AN-485 Digital Filtering Using the HPC	5-21
AN-486 A Floating Point Package for the HPC	5-36
AN-487 A Radix 2 FFT Program for the HPC	5-89
AN-497 Expanding the HPC Address Space	5-114
AN-510 Assembly Language Programming for the HPC	5-125
AN-521 Dual Tone Multiple Frequency (DTMF)	3-146
AN-550 A Software Driver for the HPC Universal Peripheral Interface Port	5-130
AN-551 The HPC as a Front-End Processor	5-185
AN-552 Interfacing a Serial EEPROM to the National HPC16083	5-249
AN-561 I ² C-Bus—Interface with HPC	5-266
AN-577 Extended Memory Support for HPC	5-286
AN-579 MICROWIRE/PLUS Serial Interface for COP800 Family	3-155
AN-585 High Performance Controller in Information Control Applications	5-330
AN-586 Pulse Width Modulation Using HPC	5-338
AN-587 C in Embedded Systems and the Microcontroller World	5-346
AN-593 HPC16400 A Communication Microcontroller with HDLC Support	5-352
AN-596 COP800 MathPak	3-167
AN-603 Signed Integer Arithmetic on the HPC	5-362
AN-607 Pulse Width Modulation A/D Conversion Techniques with COP800 Family Microcontrollers	3-203
AN-643 EMI/RFI Board Design	5-374
AN-662 COP800 Based Automated Security/Monitoring System	3-210
AN-663 Sound Effects for the COP800 Family	3-218
AN-666 DTMF Generation with a 3.58 MHz Crystal	3-241
AN-673 2-Way Multiplexed LCD Drive and Low Cost A/D Converter Using V/F Techniques with COP8 Microcontrollers	3-269
AN-681 PC MOUSE Implementation Using COP800	3-288
AN-714 Using COP800 Devices to Control DC Stepper Motors	3-313
AN-734 MF2 Compatible Keyboard with COP8 Microcontrollers	3-323
AN-736 Interfacing the HPC46064 to the DP83200 FDDI Chip Set	5-391
AN-739 RS-232C Interface with COP800	3-343
AN-749 Quadrature Signal Interface to a COP400 Microcontroller	3-355
AN-786 LCD Direct Drive Using HPC	5-397
AN-798 Improved UART Clocking Techniques on New Generation HPCs	5-413
COP Brief 2 Easy Logarithms for COP400	3-3
COP Brief 6 RAM Keep-Alive	3-14

Alpha-Numeric Index (Continued)

COP Note 1 Analog to Digital Conversion Techniques with COPS Family Microcontrollers	3-15
COP Note 4 The COP444L Evaluation	3-47
COP Note 5 Oscillator Characteristics of COPS Microcontrollers	3-52
COP Note 6 Triac Control Using the COP400 Microcontroller Family	3-69
COP Note 7 Testing of COP400 Family Devices	3-77
COP86L20C Single-Chip microCMOS Microcontroller	2-53
COP86L22C Single-Chip microCMOS Microcontroller	2-53
COP86L40C Single-Chip microCMOS Microcontroller	2-53
COP86L42C Single-Chip microCMOS Microcontroller	2-53
COP224C Single-Chip CMOS Microcontroller	1-8
COP225C Single-Chip CMOS Microcontroller	1-8
COP226C Single-Chip CMOS Microcontroller	1-8
COP244C Single-Chip CMOS Microcontroller	1-8
COP245C Single-Chip CMOS Microcontroller	1-8
COP310C Single-Chip CMOS Microcontroller	1-26
COP310L Single-Chip N-Channel Microcontroller	1-41
COP311C Single-Chip CMOS Microcontroller	1-26
COP311L Single-Chip N-Channel Microcontroller	1-41
COP313C Single-Chip CMOS Microcontroller	1-72
COP313CH Single-Chip CMOS Microcontroller	1-72
COP313L Single-Chip Microcontroller	1-59
COP314L Single-Chip N-Channel Microcontroller	1-86
COP320 Single-Chip N-Channel Microcontroller	1-101
COP320L Single-Chip N-Channel Microcontroller	1-125
COP321 Single-Chip N-Channel Microcontroller	1-101
COP321L Single-Chip N-Channel Microcontroller	1-125
COP322 Single-Chip N-Channel Microcontroller	1-101
COP322L Single-Chip N-Channel Microcontroller	1-125
COP324C Single-Chip CMOS Microcontroller	1-152
COP325C Single-Chip CMOS Microcontroller	1-152
COP326C Single-Chip CMOS Microcontroller	1-152
COP344C Single-Chip CMOS Microcontroller	1-152
COP344L Single-Chip N-Channel Microcontroller	1-172
COP345C Single-Chip CMOS Microcontroller	1-152
COP345L Single-Chip N-Channel Microcontroller	1-172
COP400 Family	1-3
COP400 Microcontroller Development Support	7-5
COP401L ROMless N-Channel Microcontroller	1-196
COP401L-R13 ROMless N-Channel Microcontroller	1-210
COP401L-X13 ROMless N-Channel Microcontroller	1-210
COP402-5 ROMless N-Channel Microcontroller	1-223
COP404C ROMless CMOS Microcontroller	1-240
COP404LSN-5 ROMless N-Channel Microcontroller	1-257
COP410C Single-Chip CMOS Microcontroller	1-26
COP410L Single-Chip N-Channel Microcontroller	1-41
COP411C Single-Chip CMOS Microcontroller	1-26
COP411L Single-Chip N-Channel Microcontroller	1-41
COP413C Single-Chip CMOS Microcontroller	1-72
COP413CH Single-Chip CMOS Microcontroller	1-72
COP413L Single-Chip Microcontroller	1-59
COP414L Single-Chip N-Channel Microcontroller	1-86

Alpha-Numeric Index (Continued)

COP420 Single-Chip N-Channel Microcontroller	1-101
COP420L Single-Chip N-Channel Microcontroller	1-125
COP420P Piggyback EPROM Microcontroller	1-271
COP421 Single-Chip N-Channel Microcontroller	1-101
COP421L Single-Chip N-Channel Microcontroller	1-125
COP422 Single-Chip N-Channel Microcontroller	1-101
COP422L Single-Chip N-Channel Microcontroller	1-125
COP424C Single-Chip CMOS Microcontroller	1-152
COP425C Single-Chip CMOS Microcontroller	1-152
COP426C Single-Chip CMOS Microcontroller	1-152
COP444C Single-Chip CMOS Microcontroller	1-152
COP444CP Piggyback EPROM Microcontroller	1-271
COP444L Single-Chip N-Channel Microcontroller	1-172
COP444LP Piggyback EPROM Microcontroller	1-271
COP445C Single-Chip CMOS Microcontroller	1-152
COP445L Single-Chip N-Channel Microcontroller	1-172
COP472-3 Liquid Crystal Display Controller	6-7
COP620C Single-Chip microCMOS Microcontroller	2-5
COP622C Single-Chip microCMOS Microcontroller	2-5
COP640C Single-Chip microCMOS Microcontroller	2-5
COP642C Single-Chip microCMOS Microcontroller	2-5
COP680C Microcontroller	2-75
COP681C Microcontroller	2-75
COP684CL Single-Chip microCMOS Microcontroller	2-98
COP684CS Single-Chip microCMOS Microcontroller	2-243
COP684EG Single-Chip microCMOS Microcontroller	2-203
COP688CL Single-Chip microCMOS Microcontroller	2-98
COP688CS Single-Chip microCMOS Microcontroller	2-243
COP688EG Single-Chip microCMOS Microcontroller	2-203
COP800 Development System	7-12
COP800 Family	2-3
COP820C Single-Chip microCMOS Microcontroller	2-5
COP820CJ Single-Chip microCMOS Microcontroller	2-27
COP820CJM Single-Chip microCMOS Microcontroller	2-354
COP822C Single-Chip microCMOS Microcontroller	2-5
COP822CJ Single-Chip microCMOS Microcontroller	2-27
COP822CJM Single-Chip microCMOS Microcontroller	2-354
COP823CJ Single-Chip microCMOS Microcontroller	2-27
COP840C Single-Chip microCMOS Microcontroller	2-5
COP842C Single-Chip microCMOS Microcontroller	2-5
COP842CMH Microcontroller Emulator	2-300
COP880C Microcontroller	2-75
COP880CMH Microcontroller Emulator	2-307
COP881C Microcontroller	2-75
COP881CMH Microcontroller Emulator	2-307
COP884CF Single-Chip microCMOS Microcontroller	2-133
COP884CG Single-Chip microCMOS Microcontroller	2-167
COP884CGMH Single-Chip microCMOS Microcontroller Emulator	2-344
COP884CL Single-Chip microCMOS Microcontroller	2-98
COP884CS Single-Chip microCMOS Microcontroller	2-243
COP884EG Single-Chip microCMOS Microcontroller	2-203

Alpha-Numeric Index (Continued)

COP888CF Single-Chip microCMOS Microcontroller	2-133
COP888CFMH Single-Chip microCMOS Microcontroller Emulator	2-334
COP888CG Single-Chip microCMOS Microcontroller	2-167
COP888CGMH Single-Chip microCMOS Microcontroller Emulator	2-344
COP888CL Single-Chip microCMOS Microcontroller	2-98
COP888CLMH Single-Chip microCMOS Microcontroller Emulator	2-325
COP888CS Single-Chip microCMOS Microcontroller	2-243
COP888CSMH microCMOS Microcontroller Emulator	2-363
COP888EG Single-Chip microCMOS Microcontroller	2-203
COP888EGMH Single-Chip microCMOS Microcontroller Emulator	2-344
COP920C Single-Chip microCMOS Microcontroller	2-5
COP922C Single-Chip microCMOS Microcontroller	2-5
COP940C Single-Chip microCMOS Microcontroller	2-5
COP942C Single-Chip microCMOS Microcontroller	2-5
COP980C Microcontroller	2-75
COP981C Microcontroller	2-75
COP984CF Single-Chip microCMOS Microcontroller	2-133
COP984CL Single-Chip microCMOS Microcontroller	2-98
COP984CS Single-Chip microCMOS Microcontroller	2-243
COP988CF Single-Chip microCMOS Microcontroller	2-133
COP988CL Single-Chip microCMOS Microcontroller	2-98
COP988CS Single-Chip microCMOS Microcontroller	2-243
COP8620C Single-Chip microCMOS Microcontroller	2-53
COP8622C Single-Chip microCMOS Microcontroller	2-53
COP8640C Single-Chip microCMOS Microcontroller	2-53
COP8640CMH Microcontroller Emulator	2-316
COP8642C Single-Chip microCMOS Microcontroller	2-53
COP8642CMH Microcontroller Emulator	2-316
COP8780C Single-Chip EPROM/OTP Microcontroller	2-283
COP8781C Single-Chip EPROM/OTP Microcontroller	2-283
COP8782C Single-Chip EPROM/OTP Microcontroller	2-283
Development Support	7-3
HPC Microcontroller Development System	7-22
HPC Software Support Package	7-35
HPC16003 High-Performance Microcontroller	4-6
HPC16004 High-Performance Microcontroller	4-75
HPC16064 High-Performance Microcontroller	4-75
HPC16083 High-Performance Microcontroller	4-6
HPC26003 High-Performance Microcontroller	4-6
HPC26004 High-Performance Microcontroller	4-75
HPC26064 High-Performance Microcontroller	4-75
HPC26083 High-Performance Microcontroller	4-6
HPC36003 High-Performance Microcontroller	4-6
HPC36004 High-Performance Microcontroller	4-75
HPC36064 High-Performance Microcontroller	4-75
HPC36083 High-Performance Microcontroller	4-6
HPC36104 High-Performance Microcontroller with A/D	4-39
HPC36164 High-Performance Microcontroller with A/D	4-39
HPC36400E High-Performance Communications Microcontroller	4-108
HPC46003 High-Performance Microcontroller	4-6
HPC46004 High-Performance Microcontroller	4-75

Alpha-Numeric Index (Continued)

HPC46064 High-Performance Microcontroller	4-75
HPC46083 High-Performance Microcontroller	4-6
HPC46100 High-Performance Microcontroller with DSP Capability	4-165
HPC46104 High-Performance Microcontroller with A/D	4-39
HPC46164 High-Performance Microcontroller with A/D	4-39
HPC46400E High-Performance Communications Microcontroller	4-108
HPC167064 High-Performance Microcontroller with a 16K UV Erasable CMOS EPROM	4-134
HPC467064 High-Performance Microcontroller with a 16K UV Erasable CMOS EPROM	4-134
ISDN Basic Rate Interface Software for the HPC16400 High Performance Data Communications Microcontroller	7-45



Section 1
COP400 Family



Section 1 Contents

COP400 Family	1-3
ROM'd Devices	
COP224C/COP225C/COP226C/COP244C/COP245C Single-Chip 1k and 2k CMOS Microcontrollers	1-8
COP410C/COP411C/COP310C/COP311C Single-Chip CMOS Microcontrollers	1-26
COP410L/COP411L/COP310L/COP311L Single-Chip N-Channel Microcontrollers	1-41
COP413L/COP313L Single-Chip Microcontrollers	1-59
COP413C/COP413CH/COP313C/COP313CH Single-Chip CMOS Microcontrollers	1-72
COP414L/COP314L Single-Chip N-Channel Microcontrollers	1-86
COP420/COP421/COP422/COP320/COP321/COP322 Single-Chip N-Channel Microcontrollers	1-101
COP420L/COP421L/COP422L/COP320L/COP321L/COP322L Single-Chip N-Channel Microcontrollers	1-125
COP424C/COP425C/COP426C/COP324C/COP325C/COP326C and COP444C/COP445C/COP344C/COP345C Single-Chip 1k and 2k CMOS Microcontrollers	1-152
COP444L/COP445L/COP344L/COP345L Single-Chip N-Channel Microcontrollers	1-172
ROMless Devices	
COP401L ROMless N-Channel Microcontroller	1-196
COP401L-X13/COP401L-R13 ROMless N-Channel Microcontrollers	1-210
COP402-5 ROMless N-Channel Microcontroller	1-223
COP404C ROMless CMOS Microcontroller	1-240
COP404LSN-5 ROMless N-Channel Microcontroller	1-257
COP420P/COP444CP/COP444LP Piggyback EPROM Microcontrollers	1-271



The 4-Bit COP400 Family: Optimized for Low-Cost Control

National's COP400 family offers the broadest range of low-priced, 4-bit microcontrollers on the market.

Key Features

- High-performance 4-bit microcontroller
- 4 μ s–16 μ s instruction-cycle time
- ROM-efficient instruction set
- On-chip ROM from 0.5k to 2k
- On-chip RAM from 32 x 4 to 160 x 4
- More than 60 compatible devices in family
- Common pin-outs
- NMOS and P²C²MOS™
- MICROWIRE™ serial interface
- Wide operating voltage range: +2.4V to +6.3V
- Military temp range available: –55°C to +125°C
- 20- to 28-pin packages
(incl. 20-, 24-pin SO and 28-pin PLCC)

And far from being "old technology," 4-bit microcontrollers are meeting significant market needs in more applications than ever before. In fact, National shipped more than 40 million 4-bit devices last year alone. The reason for the continuing strength of the COP400 family is its versatility. You can select from over 60 different, compatible devices. The first under 50¢ microcontroller set a new standard of value for cost/performance. You can select devices with a wide variety of ROM and RAM combinations, from 0.5k ROM and 32 x 4 RAM to 2k ROM and 160 x 4 RAM.

And every COP400 family member shares the same powerful, ROM-efficient instruction set and the same pin-out, so you can migrate between devices without re-engineering.

And like all of National's microcontrollers, the COP400 can be optimized to meet your specific application needs, with a variety of I/O options, pin-outs, and package types, from DIPs to SMDs.

COPST™ microcontrollers can be used to replace discrete logic in high-volume consumer products and low-volume industrial products allowing you to add features, miniaturize and reduce component count.

Key Applications

- Consumer electronics
- Automotive
- Industrial control
- Toys/games
- Telephones

Wide Acceptance

COPS wide acceptance comes from innovative products. National has built on this established family with continued and enhanced devices.

- The first under-a-dollar microcontroller led to a broader range of automotive and consumer applications.
- The first high-speed, low-power CMOS microcontrollers with 0.5k ROM provides design flexibility at low cost.
- The first microcontroller implementing MICROWIRE/PLUST™ allowing two-way communication across only three lines.
- The first microcontroller implementing Post-Metal Programming (PMP™) for quick turns prototyping and production.

PMP

Post-Metal Programming (PMP), another NSC microcontroller first. Takes advantage of:

- Seasonal or volatile market demand
- Narrow windows of opportunity in highly competitive markets
- Simplified inventory control
- Reduced safety stock

Get all the advantages of custom-programmed microcontrollers with all the business advantages of low cost, quick-turn prototyping and production.

The secret is an entirely new process technology called Post-Metal Programming.

PMP (Continued)**INSIDE PMP**

Post-Metal Programming is a high energy implantation process that allows the ROM layer of a microcontroller to be programmed after final metallization. That means every die layer can be fully fabricated, except for the passivation layers, and held in inventory. Then when you request a ROM pattern, a ROM implant mask is generated and the buried ROM layer is programmed with an ion beam.

The wafer is passivated and cut into dice which are then packaged on a quick-turn line.

So in as little as two weeks, you've got prototypes.

See COP400 Family of Microcontroller selection

4-WEEK PRE-PRODUCTION QUANTITIES

Wafer fab accounts for the majority of prototyping and production time for integrated circuits.

With PMP, however, the dice are essentially complete and in inventory.

So we can take your approved prototypes right into full production or small quantity pre-production in as little as four weeks.

WINNING THE TIME-TO-MARKET RACE

The electronics market won't wait for anyone. If your competitors make a move, you've got to respond now.

You can't wait around for proof-of-design prototypes. Even a week can make a difference between success or failure. Between gaining market share or losing it. Between staying ahead of the other guys or falling behind. With PMP, you can stretch that lead by *weeks*. In fact, if you compare the quick-turn PMP process to conventional prototype-and-production timetables, you'll see that *you can actually gain as much as 3 1/2 months over your competitors!*

NO EXTRA COST

PMP is available at *no extra cost*.

Compare that with the traditional "alternative" for quick-turn prototyping of user-programmable ROM. EPROM and EEPROM can easily drive your unit costs up to as much as \$6!

And when you consider the additional cost-savings of being able to reduce your safety stock in inventory, knowing you can get quick-turns in a few weeks, the PMP process and

National Semiconductor microcontrollers not only make good *engineering* sense, they make good *business* sense.

System Solutions

The COP400 family provides a flexible, cost-effective system solutions to all applications requiring timing, counting, or control functions.

And, bottom line, if a 4-bit controller can do the job, why pay more?

Development Support**DEVELOPMENT SYSTEM**

The Microcomputer On-Line Emulator Development System is a low cost development system and emulator for COPs microcontroller products. The Development System consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the COP400 Development System is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations. It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem or to connect to other Development Systems in a multi-Development System environment.

The Development System can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

See AN-456 for more information.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP400	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COPS-PB1	Personality Board	COP400 Personality Board Users Manual	420408189-001
	MOLE-COPS-IBM	Assembler Software for IBM	COP400 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424409479-002 420040416-001
	424410284-001	Programmers Manual		424410284-001

COP400 Family of Microcontrollers

Commercial Temp Version 0°C to +70°C	Industrial Temp Version -40°C to +85°C	Military Temp Version -55°C to +125°C	Technology	Description		Features									Development Tools		Data Sheet Page
				Memory		I/O		Interrupt	Stack	Time Base Counter	Micro Bus	Typ. 5V Operat. Power	Max Standby at 3.3V	Size (Pins)	ROMless Device	Piggyback	
				ROM (Bytes)	RAM (Digits)	I/O Pins	Serial I/O										
COP413L*	COP313L		NMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401L- X13/R13 COP401LN COP401LN COP401LN	1-59	
COP414L*	COP314L		NMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	15 mW	7.5 mW	20		1-86	
COP410L	COP310L		NMOS Low Power	0.5k	32	19	Yes	No	2 Level	No	No	15 mW	7.5 mW	24		1-41	
COP411L	COP311L		NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	15 mW	7.5 mW	20		1-41	
COP413C	COP313C		CMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN COP404CN COP404CN COP404CN	1-72	
COP413CH	COP313CH		CMOS Hi Speed	0.5k	32	15	Yes	No	2 Level	No	No	1 mW	0.1 mW	20		1-72	
COP410C	COP310C		CMOS Hi Speed	0.5k	32	19	Yes	No	2 Level	No	No	1 mW	0.1 mW	24		1-26	
COP411C	COP311C		CMOS Hi Speed	0.5k	32	16	Yes	No	2 Level	No	No	1 mW	0.1 mW	20		1-26	
COP420	COP320		NMOS Hi Speed	1.0k	64	23	Yes	1 Source	3 Level	Yes	No	100 mW	N/A mW	28	COP402N-5 COP402N-5 COP402N-5	1-101	
COP421	COP321		NMOS Hi Speed	1.0k	64	19	Yes	No	3 Level	Yes	No	100 mW	N/A mW	24		1-101	
COP422	COP322		NMOS Hi Speed	1.0k	64	16	Yes	No	3 Level	Yes	No	100 mW	N/A mW	20		1-101	
COP424C*	COP324C	COP224C (Note 1)	CMOS Hi Speed	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	28	COP404CN COP404CN COP404CN	1-152	
COP425C*	COP325C	COP225C (Note 1)	CMOS Hi Speed	1.0k	64	19	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	24		1-152	
COP426C*	COP326C	COP226C (Note 1)	CMOS Hi Speed	1.0k	64	16	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	20		1-152	
COP420L*	COP320L		NMOS Low Power	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	45 mW	9.9 mW	28	COP404LSN-5 COP404LSN-5 COP404LSN-5	1-125	
COP421L*	COP321L		NMOS Low Power	1.0k	64	19	Yes	No	3 Level	Yes	No	45 mW	9.9 mW	24		1-125	
COP422L*	COP322L		NMOS Low Power	1.0k	64	16	Yes	No	3 Level	Yes	No	45 mW	9.9 mW	20		1-125	
COP444C*	COP344C	COP244C (Note 1)	CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	28	COP404CN COP404CN	1-152	
COP445C*	COP345C	COP245C (Note 1)	CMOS Hi Speed	2.0k	128	19	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	24		1-152	
COP444L	COP344L		NMOS Low Power	2.0k	128	23	Yes	1 Source	3 Level	Yes	No	65 mW	9.9 mW	28	COP404LSN-5 COP404LSN-5	1-172	
COP445L	COP345L		NMOS Low Power	2.0k	128	19	Yes	No	3 Level	Yes	No	65 mW	9.9 mW	24		1-172	

Note 1: Datasheet found on page 1-8.

*Microcontrollers available with Quick-Turns Post-Metal Programming (PMP).

COPS Family Development Tools

Commercial Temp Version 0°C to +70°C			Technology	Description		Features									Supplementary Description	Data Sheet Page
				Memory		I/O		Interrupt	Stack	Time Base Counter	Micro Bus	Typ. 5V Operat. Power	Max Standby at 3.3V	Size (Pins)		
				ROM (Bytes)	RAM (Digits)	I/O Pins	Serial I/O									
ROMless																
COP401L-X13			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	Has XTAL Oscillator Option	1-210
COP401L-R13			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	Has RC Oscillator Option	1-210
COP401L			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	ROMless Version of COP410L	1-196
COP402-5			NMOS Hi Speed	1.0k	63	20	Yes	1 Source	3 Level	Yes	No	50 mW	N/A mW	40	Has Interrupt, No Microbus	1-223
COP404LSN-5			NMOS Low Power	1.0k	128	20	Yes	1 Source	3 Level	Yes	No	125 mW	N/A mW	40	W/Push-Pull Mem Interface	1-257
COP404C			CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	48	CMOS ROMless Device	1-240
PIGGYBACK																
COP420P			NMOS Hi Speed	1.0k	64	23	Yes	3 Sources	3 Level	Yes	No	50 mW	N/A mW	28	Includes: CPU, RAM, I/O	1-271
COP444LP			NMOS Low Power	2.0k	128	23	Yes	3 Sources	3 Level	Yes	No	125 mW	N/A mW	28	and EPROM Socket	1-271
COP444CP			CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	1 mW	28	Will Accept Standard EPROM	1-271

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

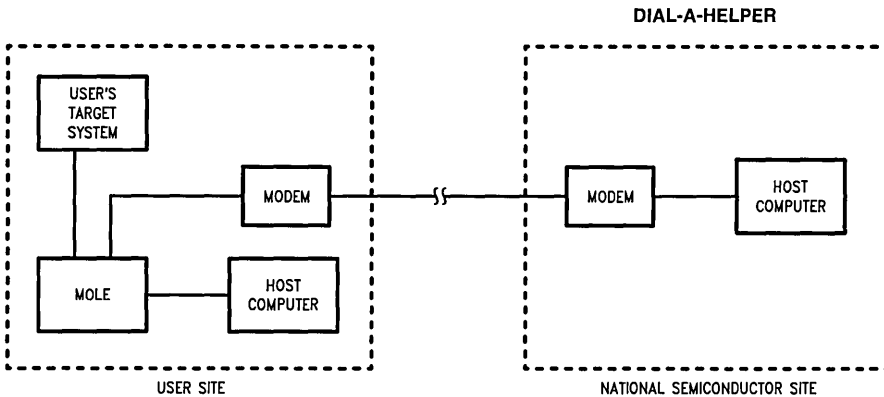
If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP
 Information System Package Contains
 DIAL-A-HELPER Users Manual P/N
 Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

- Voice: (408) 721-5582
- Modem: (408) 739-1162
- Baud: 300 or 1200 baud
- Set-Up: Length: 8-bit
- Parity: None
- Stop bit: 1
- Operation: 24 hrs., 7 days



TL/XX/0072-1



COP224C/COP225C/COP226C/COP244C/COP245C Single-Chip 1k and 2k CMOS Microcontrollers

General Description

The COP224C, COP225C, COP226C, COP244C and COP245C fully static, Single-Chip CMOS Microcontrollers are members of the COPSTM family, fabricated using double-poly, silicon gate microCMOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP224C and COP244C are 28 pin chips. The COP225C and COP245C are 24-pin versions (4 inputs removed) and COP226C is 20-pin version with 15 I/O lines. Standard test procedures and reliable high-density techniques provide the medium to large volume customers with a customized microcontroller at a low end-product cost. These microcontrollers are appropriate choices in many demanding control environments especially those with human interface.

Features

- Lowest power dissipation (600 μ W typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4.4 μ s instruction time
- 2k x 8 ROM, 128 x 4 RAM (COP244C/COP245C)
- 1k x 8 ROM, 64 x 4 RAM (COP224C/COP225C/COP226C)
- 23 I/O lines (COP244C and COP224C)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (4.5V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS output compatible
- Software/hardware compatible with COP400 family
- Military temperature (-55°C to +125°C) operation

Block Diagram

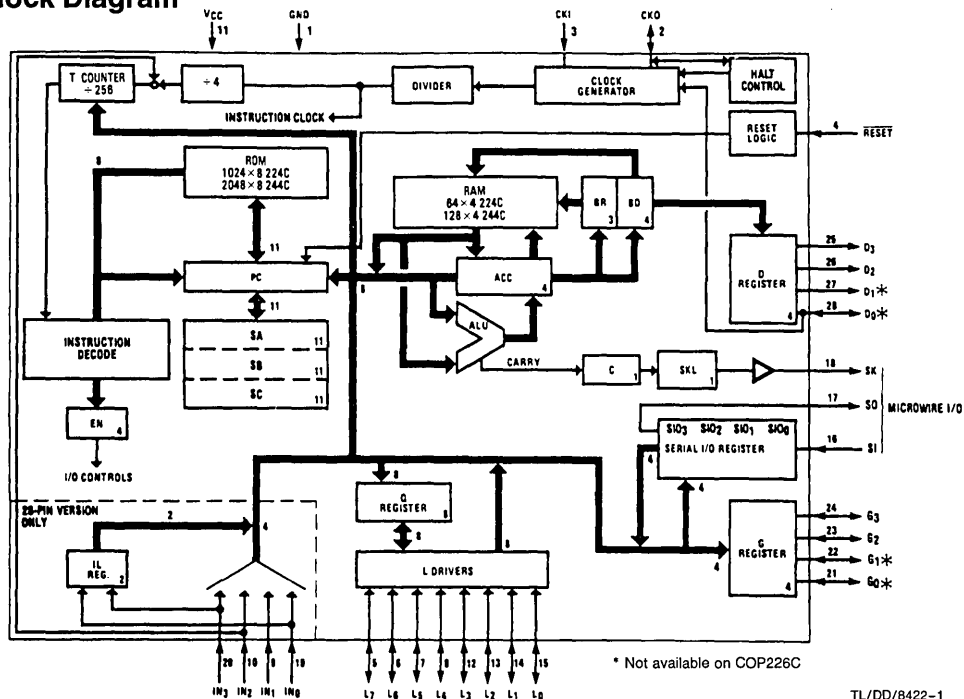


FIGURE 1

TL/DD/8422-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Total Allowable Power Dissipation	150 mW

Operating Temperature Range	-55°C to +125°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics -55°C ≤ T_A ≤ +125°C, +4.5V ≤ V_{CC} ≤ +5.5V unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Power Supply Ripple (Notes 4, 5)	Peak to Peak		0.25 V_{CC}	V
Supply Current (Note 1)	$V_{CC} = 5.0V$, $t_c = 4.4 \mu s$ (t_c is instruction cycle time)		5	mA
HALT Mode Current (Note 2)	$V_{CC} = 5.0V$, $F_{IN} = 0$ kHz		200	μA
Input Voltage Levels				
RESET, CKI, D ₀ (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		-10	+10	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels (except CKO)				
LSTTL Operation	Standard Outputs $V_{CC} = 5.0V \pm 10\%$			
Logic High	$I_{OH} = -100 \mu A$	2.7		V
Logic Low	$I_{OL} = 400 \mu A$		0.6	V
CMOS Operation				
Logic High	$I_{OH} = -10 \mu A$	$V_{CC} - 0.2$		V
Logic Low	$I_{OL} = 10 \mu A$		0.2	V
CKO Current Levels (As Clock Out)				
Sink	} $CKI = V_{CC}$, $V_{OUT} = V_{CC}$	0.2		mA
÷ 8		0.4		mA
÷ 16		0.8		mA
Source	} $CKI = 0V$, $V_{OUT} = 0V$	-0.2		mA
÷ 8		-0.4		mA
÷ 16		-0.8		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			50	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	$V_{IN} = 0.2 V_{CC}$		2.0	mA
To Halt	$V_{IN} = 0.7 V_{CC}$		3.0	mA
TRI-STATE or Open Drain Leakage Current		-10	+10	μA

AC Electrical Characteristics $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$, $+4.5\text{V} \leq V_{\text{CC}} \leq +5.5\text{V}$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)		4.4	DC	μs
Operating CKI Frequency	$\div 4$ mode $\div 8$ mode $\div 16$ mode	DC	0.9	MHz
		DC	1.8	MHz
		DC	3.6	MHz
Duty Cycle (Note 4)	$f_1 = 3.6$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 3.6$ MHz External Clock		60	ns
Fall Time (Note 4)	$f_1 = 3.6$ MHz External Clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)	R = $30\text{k} \pm 5\%$ C = $82\text{ pF} \pm 5\%$ ($\div 4$ Mode)	6	18	μs
Inputs: (See Figure 3) (Note 4) t_{SETUP}	G Inputs SI Input All Others	$t_c/4 + 0.8$		μs
				μs
				μs
				μs
t_{HOLD}		0.4		μs
Output Propagation Delay t_{PD1} , t_{PD0}	$V_{\text{OUT}} = 1.5\text{V}$, $C_L = 100\text{ pF}$, $R_L = 5\text{k}$		1.4	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC} , L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 4: This parameter is not tested but guaranteed by design. Variation due to the device included.

Note 5: Voltage change must be less than 0.25 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

RETS COP244CX DC Parameters Test Conditions $5.3\text{V} \leq V_{\text{CC}} \leq 3\text{V}$ Unless Otherwise Specified

Symbol	Parameter (Note 1)	V_{CC}	Conditions	Test #	SBGRP 1 +25°C		SBGRP 2 +125°C		SBGRP 3 -55°C		Drift Limits (25°C)	Units
					Min	Max	Min	Max	Min	Max		
I_{DD1}	Supply Current		$V_{\text{DD}} = 4\text{V}$, $F_{\text{IN}} = 64\text{ kHz}$			85		155		85		μA
I_{DD2}	Halt Current		$V_{\text{DD}} = 4\text{V}$			35		125		35		μA
V_{IH1} V_{IL1}	Input Voltage Reset, CKI: Logic High Logic Low All Other Inputs:				9 V_{CC}	1 V_{CC}	9 V_{CC}	1 V_{CC}	9 V_{CC}	1 V_{CC}		V
V_{IH2} V_{IL2}												Logic High Logic Low
V_{OH1} V_{OL1}	Output Voltage LSTTL Operation: Logic High Logic Low CMOS Operation:	4.75V 4.75V	$I_{\text{OH}} = -100\ \mu\text{A}$ $I_{\text{OL}} = 400\ \mu\text{A}$		2.7	0.4	2.7	0.4	2.7	0.4		V
V_{OH2} V_{OL2}												Logic High Logic Low
I_{OH} I_{OL}	Output Current Logic High Logic Low	3V 3V	$V = 0\text{V}$ $V = 3\text{V}$		-100 200		-100 200		-100 200			μA μA
I_{IN1} I_{IN2}	Input Leakage High-Z TRI-STATE or Open Drain				-2.5 -4	2.5 4	-2.5 -4	2.5 4	-2.5 -4	2.5 4		μA μA

RETS COP244CX

DEVICE: COP244C-XXX/883

FUNCTION: 4-BIT CMOS MICROCONTROLLER

RETS COP244CX AC Parameters Test Conditions $5.3V \leq V_{CC} \leq 3V$ Unless Otherwise Specified

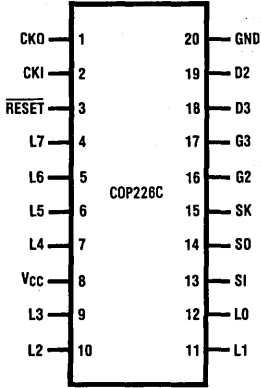
Symbol	Parameter	V _{CC}	Conditions	Test #	SBGRP 9 + 25°C		SBGRP 10 + 125°C		SBGRP 11 - 55°C		Drift Limits (25°C)	Units
					Min	Max	Min	Max	Min	Max		
t _{CC}	Instruction Cycle Time (Note 1)		Mode Divided by 8, V _{DD} = 3V		80	125	80	125	80	125		μs
F _{IN}	Operating Clock Frequency (Note 1)		V _{DD} = 3V, 30% ≤ Duty Cycle ≤ 50%		64	100	64	100	64	100		kHz
	Inputs t _{SETUP} (Note 2) t _{SETUP,G} Inputs For SKGZ & SKGBZ (Note 2) t _{HOLD} (Note 1)		V = 4.5V		2 32 0.6		2 32 0.6		2 32 0.6			μs μs μs
t _{PD1} t _{PD2}	Output Prop Delay (Note 1)	4.5V	R _L = 5k, C _L = 100 pF V _{OUT} = 1.5V			6 6		6 6		6 6		μs μs

RETS COP244CX DEVICE: COP244C-XXXD/883 FUNCTION: 4-BIT CMOS MICROCONTROLLER

Note 1: Parameter tested go-no-go only.
Note 2: Guaranteed by design and not tested.

Connection Diagrams

S.O. Wide and DIP

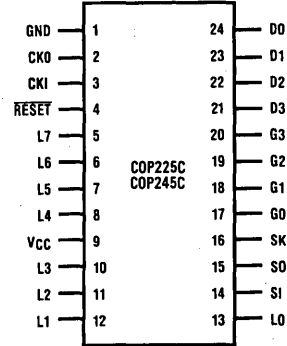


Top View

TL/DD/8422-2

- Order Number COP226C-XXX/N
See NS Molded Package Number N20A
- Order Number COP226C-XXX/D
See NS Hermetic Package Number D20A
- Order Number COP226C-XXX/WM
See NS Surface Mount Package Number M20B

S.O. Wide and DIP

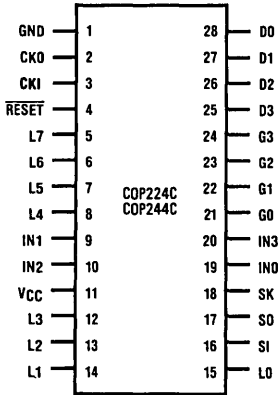


Top View

TL/DD/8422-3

- Order Number COP225C-XXX/N
or COP245C-XXX/N
See NS Molded Package Number N24A
- Order Number COP225C-XXX/D
or COP245C-XXX/D
See NS Hermetic Package Number D24C

DIP

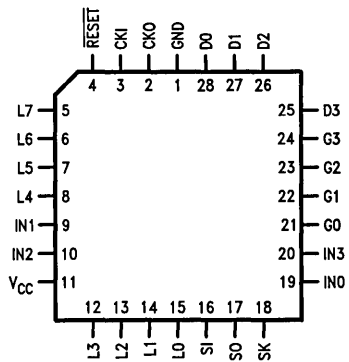


Top View

TL/DD/8422-4

- Order Number COP224C-XXX/N
or COP244C-XXX/N
See NS Molded Package Number N28B
- Order Number COP224C-XXX/D
or COP244C-XXX/D
See NS Hermetic Package Number D28C

28 PLCC



TL/DD/8422-13

- Order Number COP224C-XXX/V
or COP244C-XXX/V
See NS PLCC Package Number V28A

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L7-L0	8-bit bidirectional port with TRI-STATE	SK	Logic controlled clock output
G3-G0	4-bit bidirectional I/O port	CKI	Chip oscillator input
D3-D0	4-bit output port	CKO	Oscillator output, HALT I/O port or general purpose input
IN3-IN0	4-bit input port (28 pin package only)	RESET	Reset input
SI	Serial input or counter input	V _{CC}	Most positive power supply
SO	Serial or general purpose output	GND	Ground

Functional Description

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

Caution:

The output options available on the COP224C/225C/226C and COP244C/245C are not the same as those available on the COP324C/325C/326C, COP344C/345C, COP424C/425C/426C and COP444C/445C. Options not available on the COP224C/225C/226C and COP244C/245C are: Option 2 value 2; Option 4 value 0; Option 5 value 1; Option 9 value 0; Option 17 value 1; Option 30, Dual Clock, all values; Option 32, Microbus™, all values; Option 33 values 2, 4, and 6; Option 34 all values; and Option 35 all values.

PROGRAM MEMORY

Program Memory consists of ROM, 1024 bytes for the COP224C/225C/226C and 2048 bytes for the COP244C/245C. These bytes of ROM may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by an 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

Data memory consists of a 512-bit RAM for the COP244C/245C, organized as 8 data registers of 16×4 -bit digits.

RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register.

Data memory consists of a 256-bit RAM for the COP224C/225C/226C, organized as 4 data registers of 16×4 -bit digits. The B register is 6 bits long. Upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions.

The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, to input 4 bits of L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes depending on a mask-programmable option: as a timer or as an external event counter. When the T counter overflows, an

Functional Description (Continued)

overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in Figure 7.

Four general-purpose inputs, IN3–IN0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of

SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.

1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.
3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC + 1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.
 1. EN1 has been set.
 2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN₁ input.
 3. A currently executing instruction has been completed.

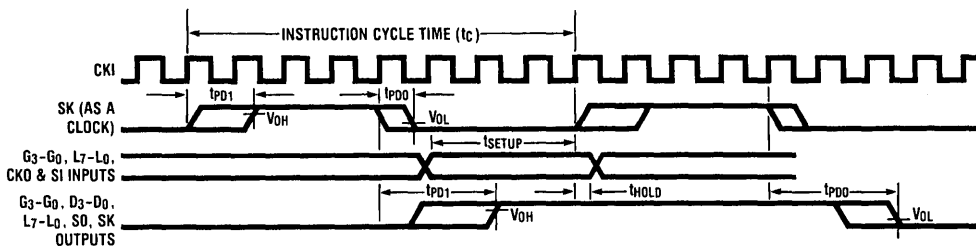


FIGURE 3. Input/Output Timing Diagrams (divide by 8 mode)

TL/DD/8422-5

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).
- c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.
- d. The instruction at hex address 0FF must be a NOP.
- e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in Figure 4 must be connected to the RESET pin (the conditions in Figure 4 must be met). The RESET pin is configured as a Schmitt trigger input. If not used, it should be connected to VCC. Initialization will occur whenever a logic "0" is applied to the RESET input, providing it stays low for at least three instruction cycle times.

Note: If CKI clock is less than 32 kHz, the internal reset logic (option #29=1) MUST be disabled and the external RC circuit must be used.

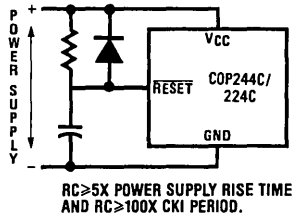


FIGURE 4. Power-Up Circuit

TL/DD/8422-6

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).

TIMER

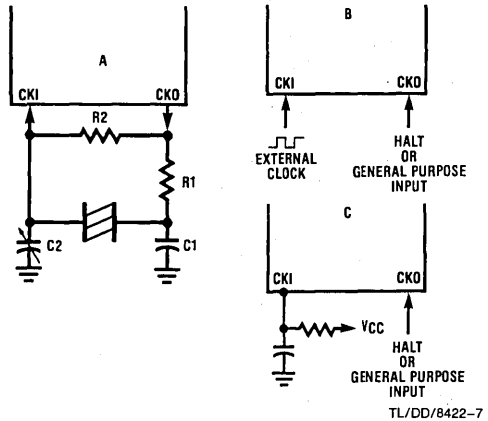
There are two modes selected by mask option:

- a. Time-base counter. In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset.

For example, using a 3.58 MHz crystal with a divide-by-16 option, the instruction cycle frequency of 223.70 kHz increments the 10-bit timer every 4.47 μ s. By presetting the counter and detecting overflow, accurate timeouts between 17.88 μ s (4 counts) and 4.577 ms (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

- b. External event counter. In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

Note: The IT instruction is not allowed in this mode.



Crystal or Resonator

Crystal Value	Component Values			
	R1	R2	C1(pF)	C2(pF)
32 kHz	220k	20M	30	6-36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6-36
3.6 MHz	1k	1M	30	6-36

RC Controlled Oscillator

R	C	Cycle Time	VCC
30k	82 pF	6-18 μ s	≥ 4.5 V

Note: $15k \leq R \leq 150k$
 $50 \text{ pF} \leq C \leq 150 \text{ pF}$

FIGURE 5. Oscillator Component Values

Functional Description (Continued)

HALT MODE

The COP244C/245C/224C/225C/226C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by the HALT instruction or by forcing CKO high when it is mask-programmed as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The chip may be awakened by one of two different methods:

- Continue function: by forcing CKO low, if it mask-programmed as a HALT I/O port, the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization).

The HALT mode is the minimum power dissipation state.

CKO PIN OPTIONS

a. Two-pin oscillator—(Crystal). See *Figure 6a*.

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. The HALT mode may be entered by program control (HALT instruction) which forces CKO high, thus inhibiting the crystal network. The circuit can be awakened only by forcing the $\overline{\text{RESET}}$ pin to a logic "0" (restart).

b. One-pin oscillator—(RC or external). See *Figure 6b*.

If a one-pin oscillator system is chosen, two options are available for CKO:

- CKO can be selected as the HALT I/O port. In that case, it is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and CKO output will stay high to keep the chip stopped if

the external driver returns to high impedance state. By forcing a low level to CKO, the chip will continue and CKO will stay low.

- As another option, CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction.

OSCILLATOR OPTIONS

There are three basic clock oscillator configurations available as shown by *Figure 5*.

- Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency optionally divided by 4, 8 or 16.
- External Oscillator. The external frequency is optionally divided by 4, 8 or 16 to give the instruction cycle time. CKO is the HALT I/O port or a general purpose input.
- RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port or a general purpose input.

Figure 7 shows the clock and timer diagram.

COP245C AND COP225C 24-PIN PACKAGE OPTION

If the COP244C/224C is bonded in a 24-pin package, it becomes the COP245C/225C, illustrated in *Figure 2*, Connection diagrams. Note that the COP245C/225C does not contain the four general purpose IN inputs (IN3–IN0). Use of this option precludes, of course, use of the IN options, interrupt feature, external event counter feature.

Note: If user selects the 24-pin package, options 9, 10, 19 and 20 must be selected as a "2". See option list.

COP226C 20-PIN PACKAGE OPTION

If the COP225C is bonded as 20-pin device it becomes the COP226C. Note that the COP226C contains all the COP225C pins except D₀, D₁, G₀, and G₁.

Block Diagram

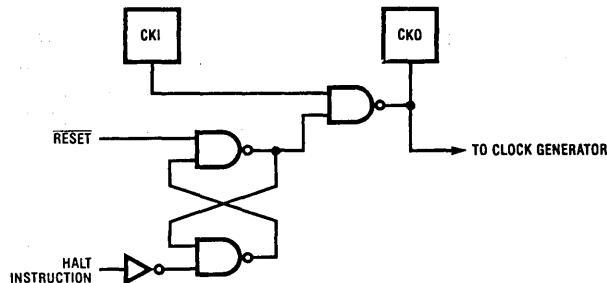
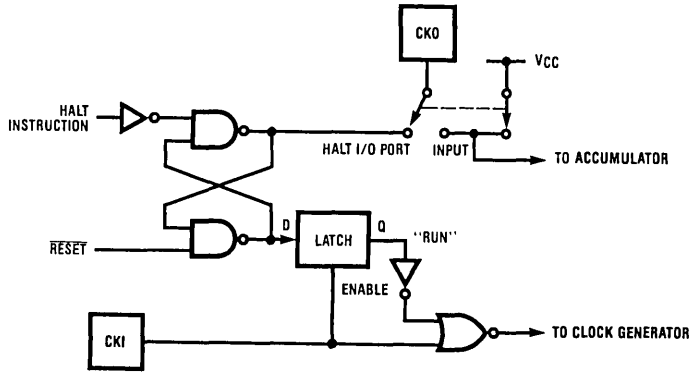


FIGURE 6a. Halt Mode—Two-Pin Oscillator

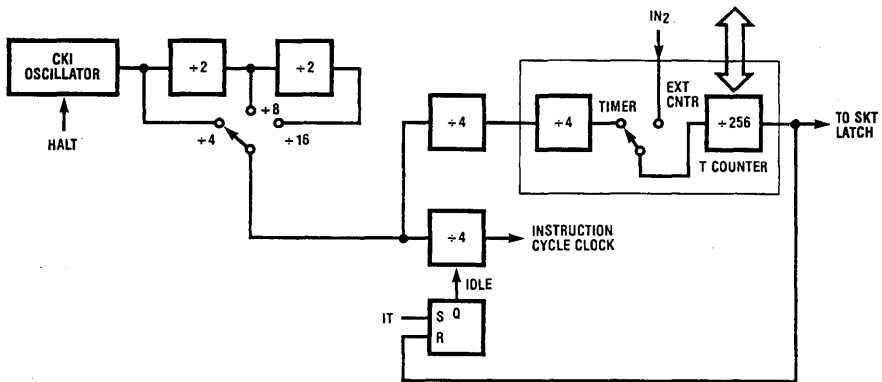
TL/DD/8422-8

Block Diagrams (Continued)



TL/DD/8422-9

FIGURE 6b. Halt Mode—One-Pin Oscillator



TL/DD/8422-10

FIGURE 7. Clock and Timer

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

TABLE II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit accumulator
B	7-bit RAM address register (6-bit for COP224C)
Br	Upper 3 bits of B (register address) (2-bit for COP224C)
Bd	Lower 4 bits of B (digit address)
C	1-bit carry register
D	4-bit data output port
EN	4-bit enable register
G	4-bit general purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA,SB,SC	11-bit 3-level subroutine stack
SIO	4-bit shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Instruction Operand Symbols

d	4-bit operand field, 0–15 binary (RAM digit select)
r	3(2)-bit operand field, 0–7(3) binary (RAM register select)
a	11-bit operand field, 0–2047 (1023)
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x

Operational Symbols

+	Plus
–	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
\bar{A}	One's complement of A
⊕	Exclusive-or
:	Range of values

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

TABLE III. COP244C/245C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
ADT		4A	<u>0100</u> <u>1010</u>	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5–	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add Immediate. Skip on Carry ($y \neq 0$)
CASC		10	<u>0001</u> <u>0000</u>	$\bar{A} + \text{RAM}(B) + C \rightarrow A$ Carry $\rightarrow C$	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	Ones complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER CONTROL INSTRUCTIONS						
JID		FF	<u>1111</u> <u>1111</u>	ROM (PC _{10:8} A,M) → PC _{7:0}	None	Jump Indirect (Notes 1, 3)
JMP	a	6--	<u>0110</u> <u>0</u> <u>a_{10:8}</u> <u>a_{7:0}</u>	a → PC	None	Jump
JP	a	--	<u>1</u> <u>a_{6:0}</u> (pages 2, 3 only)	a → PC _{6:0}	None	Jump within Page (Note 4)
		--	<u>11</u> <u>a_{5:0}</u> (all other pages)	a → PC _{5:0}		
JSRP	a	--	<u>10</u> <u>a_{5:0}</u>	PC+1 → SA → SB → SC 00010 → PC _{10:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 5)
JSR	a	6--	<u>0110</u> <u>1</u> <u>a_{10:8}</u> <u>a_{7:0}</u>	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	<u>0100</u> <u>1000</u>	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	<u>0100</u> <u>1001</u>	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33	<u>0011</u> <u>0011</u>		None	HALT Processor
IT		38	<u>0011</u> <u>1000</u>			
		33	<u>0011</u> <u>0011</u>			IDLE till Timer
		39	<u>0011</u> <u>1001</u>		None	Overflows then Continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33	<u>0011</u> <u>0011</u>	A → T _{7:4}	None	Copy A, RAM to T
		3F	<u>0011</u> <u>1111</u>	RAM(B) → T _{3:0}		
CTMA		33	<u>0011</u> <u>0011</u>	T _{7:4} → RAM(B)	None	Copy T to RAM, A
		2F	<u>0010</u> <u>1111</u>	T _{3:0} → A		
CAMQ		33	<u>0011</u> <u>0011</u>	A → Q _{7:4}	None	Copy A, RAM to Q
		3C	<u>0011</u> <u>1100</u>	RAM(B) → Q _{3:0}		
CQMA		33	<u>0011</u> <u>0011</u>	Q _{7:4} → RAM(B)	None	Copy Q to RAM, A
		2C	<u>0010</u> <u>1100</u>	Q _{3:0} → A		
LD	r	-5	<u>00</u> <u>r</u> <u>0101</u> (r=0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23	<u>0010</u> <u>0011</u>	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		--	<u>0</u> <u>r</u> <u>d</u>			
LQID		BF	<u>1011</u> <u>1111</u>	ROM(PC _{10:8} A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	<u>0100</u> <u>1100</u>	0 → RAM(B) ₀	None	Reset RAM Bit
	1	45	<u>0100</u> <u>0101</u>	0 → RAM(B) ₁		
	2	42	<u>0100</u> <u>0010</u>	0 → RAM(B) ₂		
	3	43	<u>0100</u> <u>0011</u>	0 → RAM(B) ₃		
SMB	0	4D	<u>0100</u> <u>1101</u>	1 → RAM(B) ₀	None	Set RAM Bit
	1	47	<u>0100</u> <u>0111</u>	1 → RAM(B) ₁		
	2	46	<u>0100</u> <u>0110</u>	1 → RAM(B) ₂		
	3	4B	<u>0100</u> <u>1011</u>	1 → RAM(B) ₃		

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate 1 and Increment Bd
X	r	-6	00 r 0110 (r=0:3)	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 0011 1 r d	RAM(r,d) ↔ A	None	Exchange A with RAM Pointed to Directly by r,d
XDS	r	-7	00 r 0111 (r=0:3)	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	-4	00 r 0100 (r=0:3)	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	--	00 r (d-1) (r=0:3: d=0,9:15) or 33 -- 1 r d (any r, any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A ↔ Br	None	Exchange A with Br (Note 8)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ	0	33 01	0011 0011 0000 0001	1st byte	G ₀ = 0 G ₁ = 0 G ₂ = 0 G ₃ = 0	Skip if G Bit is Zero
	1	11	0001 0001			
	2	03	0000 0011	2nd byte		
	3	13	0001 0011			
SKMBZ	0	01	0000 0001		RAM(B) ₀ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001		RAM(B) ₁ = 0	
	2	03	0000 0011		RAM(B) ₂ = 0	
	3	13	0001 0011		RAM(B) ₃ = 0	
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
ININ		33	0011 0011	IN → A	None	Input IN Inputs to A (Note 2)
		28	0010 1000			
INIL		33	0011 0011	IL ₃ , CKO, "0", IL ₀ → A	None	Input IL Latches to A (Note 3)
		29	0010 1001			
INL		33	0011 0011	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM,A
		2E	0010 1110			
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OGI	y	33	0011 0011	y → G	None	Output to G Ports Immediate
		5-	0101 y			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 3)

- Note 1:** All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.
- Note 2:** The ININ instruction is not available on the 24-pin packages since these devices do not contain the IN inputs.
- Note 3:** For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.
- Note 4:** The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.
- Note 5:** A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.
- Note 6:** LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B(Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).
- Note 7:** Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)
- Note 8:** For 2K ROM devices, A ↔ Br (0 → A3). For 1K ROM devices, A ↔ Br (0,0 → A3, A2).



Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10:PC8,A,M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC+1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC7:4, RAM(B) → PC3:0, leaving PC10, PC9 and PC8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10:8,A,M. PC10,PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT ; load T counter
SKT  ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is mask-programmed as an external event counter (option #31 = 1).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKO and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input

pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A0 is input into A1. IL latches are cleared on reset. IL latches are not available on the COP245C/225C, and COP226C.

INSTRUCTION SET NOTES

- The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.
- The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Note: The COP224C/225C/226C needs only 10 bits to address its ROM. Therefore, the eleventh bit (P10) is ignored.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal or resonator generated clock input will draw additional current. For example, a 500 kHz crystal input will typically draw 100 μ A more than a square-wave input. An R/C oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate operating current drain.

$$I_{CO} = I_Q + V \times 70 \times F_i + V \times 2400 \times F_i / D_v \quad \text{where:}$$

$$I_{CO} = \text{chip operating current drain in microamps}$$

$$I_Q = \text{quiescent leakage current (from curve)}$$

$$F_i = \text{CKI frequency in MegaHertz}$$

$$V = \text{chip } V_{CC} \text{ in volts}$$

$$D_v = \text{divide by option selected}$$

For example at 5 volts V_{CC} and 400 kHz (divide by 4)

$$I_{CO} = 120 + 5 \times 70 \times 0.4 + 5 \times 2400 \times 0.4 / 4$$

$$I_{CO} = 120 + 140 + 1200 = 1460 \mu\text{A}$$

Power Dissipation (Continued)

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$I_{ci} = I_Q + V \times 70 \times F_i$$

For example, at 5 volts V_{CC} and 400 kHz

$$I_{ci} = 120 + 5 \times 70 \times 0.4 = 260 \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$I_{ta} = I_{CO} \times \frac{T_o}{T_o + T_i} + I_{ci} \times \frac{T_i}{T_o + T_i}$$

where: I_{ta} = total average current

I_{CO} = operating current

I_{ci} = idle current

T_o = operating time

T_i = idle time

I/O OPTIONS

Outputs have the following optional configurations, illustrated in *Figure 8*:

- a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC} , compatible with CMOS and LSTTL.
- b. Open Drain — An N-channel device to ground only, allowing external pull-up as required by the user's application.
- c. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control.
- d. Open-Drain TRI-STATE L Output — This has the N-channel device to ground only.

All inputs have the following option:

- e. Hi-Z input which must be driven by the users logic.

All output drivers use two common devices numbered 1 to 2. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 9* for each of these devices to allow the designer to effectively use these I/O configurations.

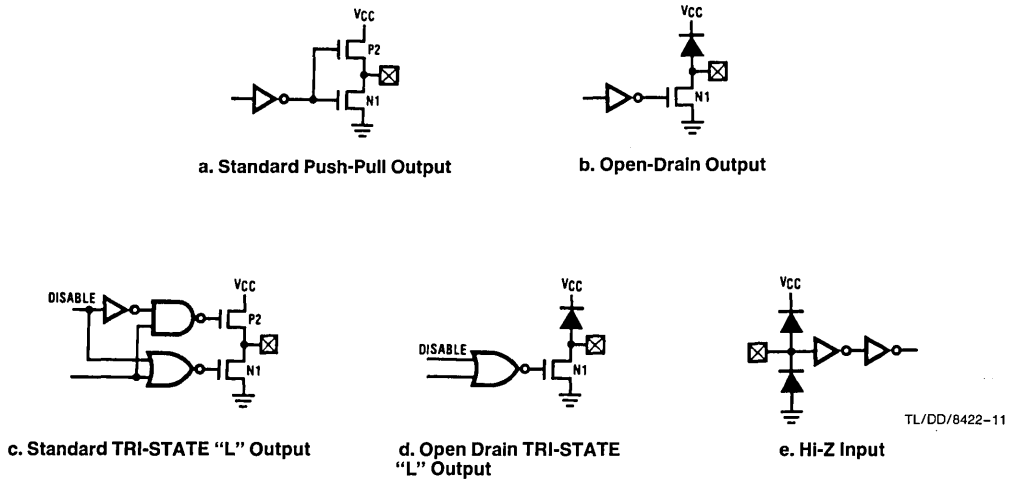


FIGURE 8. Input/Output Configurations

Power Dissipation (Continued)

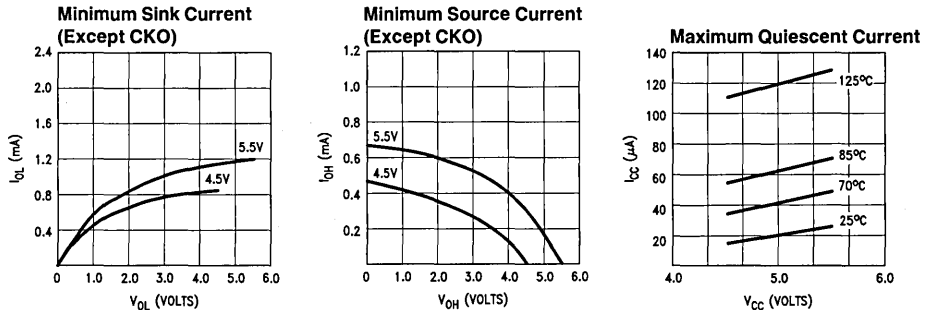


FIGURE 9. Input/Output Characteristics

TL/DD/8422-12

Option List

The COP244C/245C/224C/225C/COP226C mask-programmable options are assigned numbers which correspond with the COP244C/224C pins.

The following is a list of options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Caution:

The output options available on the COP224C/225C/226C and COP244C/245C are not the same as those available on the COP324C/325C/326C, COP344C/345C, COP424C/425C/426C and COP444C/445C. Options not available on the COP224C/225C/226C and COP244C/245C are: Option 2 value 2; Option 4 value 0; Option 5 value 1; Option 9 value 0; Option 17 value 1; Option 30, Dual Clock, all values; Option 32, Microbus, all values; Option 33 values 2, 4, and 6; Option 34 all values; and Option 35 all values.

PLEASE FILL OUT THE OPTION TABLE on the next page. Photocopy the option data and send it in with your disk or EPROM.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Pin

- = 0: clock generator output to crystal/resonator
- = 1: HALT I/O port
- = 3: general purpose input, high-Z

Option 3: CKI input

- = 0: Crystal controlled oscillator input divide by 4
- = 1: Crystal controlled oscillator input divide by 8
- = 2: Crystal controlled oscillator input divide by 16
- = 4: Single-pin RC controlled oscillator (divide by 4)
- = 5: External oscillator input divide by 4
- = 6: External oscillator input divide by 8
- = 7: External oscillator input divide by 16

Option 4: $\overline{\text{RESET}}$ input

- = 1: Hi-Z input

Option 5: L7 Driver

- = 0: Standard TRI-STATE push-pull output
- = 2: Open-drain TRI-STATE output

Option 6: L6 Driver — (same as option 5)

Option 7: L5 Driver — (same as option 5)

Option 8: L4 Driver — (same as option 5)

Option 9: IN1 input

- = 1: Hi-Z input, mandatory for 28 Pin Package
- = 2: Mandatory for 20 and 24 Pin Packages

Option 10: IN2 input — (same as option 9)

Option 11 = 0: V_{CC} Pin — no option available

Option 12: L3 Driver — (same as option 5)

Option 13: L2 Driver — (same as option 5)

Option 14: L1 Driver — (same as option 5)

Option 15: L0 Driver — (same as option 5)

Option 16: SI input — (same as option 4)

Option 17: SO Driver

- = 0: Standard push-pull output
- = 2: Open-drain output

Option 18: SK Driver — (same as option 17)

Option 19: IN0 Input — (same as option 9)

Option 20: IN3 Input — (same as option 9)

Option 21: G0 I/O Port — (same as option 17)

Option 22: G1 I/O Port — (same as option 17)

Option 23: G2 I/O Port — (same as option 17)

Option 24: G3 I/O Port — (same as option 17)

Option 25: D3 Output — (same as option 17)

Option 26: D2 Output — (same as option 17)

Option 27: D1 Output — (same as option 17)

Option List (Continued)

Option 28: D0 Output — (same as option 17)
 Option 29: Internal Initialization Logic
 = 0: Normal operation
 = 1: No internal initialization logic
 Option 30 = 0: No Option Available
 Option 31: Timer
 = 0: Time-base counter
 = 1: External event counter
 Option 32 = 0: No Option Available

Option 33: COP bonding. See note.
 (1k and 2k Microcontroller)
 = 0: 28-pin package
 = 1: 24-pin package
 (1k Microcontroller only)
 = 3: 20-pin package
 = 5: 24- and 20-pin package

Note:—If opt. #33=0 then opt. #9, 10, 19, and 20 must=1.

If opt. #33=1 then opt. #9, 10, 19 and 20 must=2, and option #31 must=0.

If opt. #33=3 or 5 then opt. #9, 10, 19, 20 must=2 and opt. #21, 22, 31 must=0.

Option 34 = 0: No Option Available

Option 35 = 0: No Option Available

Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA	OPTION DATA
OPTION 1 VALUE = <u> 0 </u> IS: GROUND PIN	OPTION 19 VALUE = <u> </u> IS: IN0 INPUT
OPTION 2 VALUE = <u> </u> IS: CKO PIN	OPTION 20 VALUE = <u> </u> IS: IN3 INPUT
OPTION 3 VALUE = <u> </u> IS: CKI INPUT	OPTION 21 VALUE = <u> </u> IS: G0 I/O PORT
OPTION 4 VALUE = <u> 1 </u> IS: RESET INPUT	OPTION 22 VALUE = <u> </u> IS: G1 I/O PORT
OPTION 5 VALUE = <u> </u> IS: L7 DRIVER	OPTION 23 VALUE = <u> </u> IS: G2 I/O PORT
OPTION 6 VALUE = <u> </u> IS: L6 DRIVER	OPTION 24 VALUE = <u> </u> IS: G3 I/O PORT
OPTION 7 VALUE = <u> </u> IS: L5 DRIVER	OPTION 25 VALUE = <u> </u> IS: D3 OUTPUT
OPTION 8 VALUE = <u> </u> IS: L4 DRIVER	OPTION 26 VALUE = <u> </u> IS: D2 OUTPUT
OPTION 9 VALUE = <u> </u> IS: IN1 INPUT	OPTION 27 VALUE = <u> </u> IS: D1 OUTPUT
OPTION 10 VALUE = <u> </u> IS: IN2 INPUT	OPTION 28 VALUE = <u> </u> IS: D0 OUTPUT
OPTION 11 VALUE = <u> 0 </u> IS: VCC PIN	OPTION 29 VALUE = <u> </u> IS: INT INIT LOGIC
OPTION 12 VALUE = <u> </u> IS: L3 DRIVER	OPTION 30 VALUE = <u> 0 </u> IS: N/A
OPTION 13 VALUE = <u> </u> IS: L2 DRIVER	OPTION 31 VALUE = <u> </u> IS: TIMER
OPTION 14 VALUE = <u> </u> IS: L1 DRIVER	OPTION 32 VALUE = <u> 0 </u> IS: N/A
OPTION 15 VALUE = <u> </u> IS: L0 DRIVER	OPTION 33 VALUE = <u> </u> IS: COP BONDING
OPTION 16 VALUE = <u> 1 </u> IS: SI INPUT	OPTION 34 VALUE = <u> 0 </u> IS: N/A
OPTION 17 VALUE = <u> </u> IS: SO DRIVER	OPTION 35 VALUE = <u> 0 </u> IS: N/A
OPTION 18 VALUE = <u> </u> IS: SK DRIVER	



COP410C/COP411C/COP310C/COP311C Single-Chip CMOS Microcontrollers

General Description

The COP410C, COP411C, COP310C, and COP311C fully static, single-chip CMOS microcontrollers are members of the COPSTM family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP411C is identical to the COP410C but with 16 I/O lines instead of 20. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller-oriented processor at a low end-product cost.

The COP310C/COP311C is the extended temperature range version of the COP410C/COP411C.

The COP404C should be used for exact emulation.

Features

- Lowest power dissipation (40 μ W typical)
- Low cost
- Power-saving HALT Mode with Continue function
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 20 I/O lines (COP410C)
- Two-level subroutine stack
- DC to 4 μ s instruction time
- Single supply operation (2.4V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of the COP400 family
- Extended temperature (-40°C to +85°C) devices available
- The military temperature range devices (-55°C to +125°C) are specified on COP210C/211C data sheet.

Block Diagram

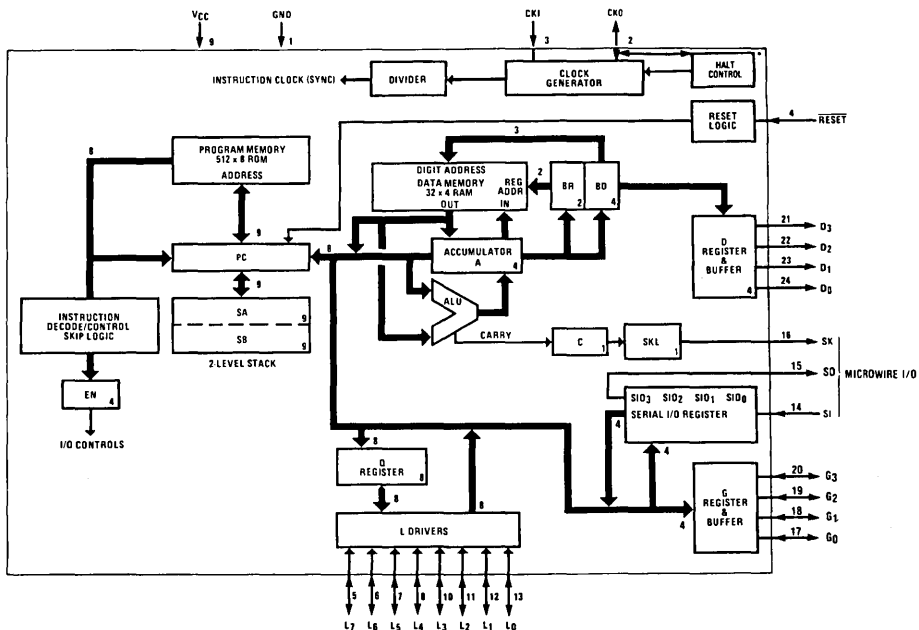


FIGURE 1. COP410C

TL/DD/5015-1

COP410C/COP411C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA

Operating Temperature Range	0°C to +70°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple (Notes 5, 6)			0.1 V_{CC}	V
Supply Current (Note 1)	$V_{CC} = 2.4V, t_c = 125 \mu\text{s}$ $V_{CC} = 5.0V, t_c = 16 \mu\text{s}$ $V_{CC} = 5.0V, t_c = 4 \mu\text{s}$ (t_c is instruction cycle time)		80 500 2000	μA μA μA
HALT Mode Current (Note 2)	$V_{CC} = 5.0V, F_{IN} = 0 \text{ kHz}$ $V_{CC} = 2.4V, F_{IN} = 0 \text{ kHz}$		30 10	μA μA
Input Voltage Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		0.9 V_{CC} 0.7 V_{CC}	0.1 V_{CC} 0.2 V_{CC}	V V V V
Hi-Z Input Leakage		-1	+1	μA
Input Capacitance (Note 6)			7	pF
Output Voltage Levels LSTTL Operation Logic High Logic Low CMOS Operation Logic High Logic Low	Standard Outputs $V_{CC} = 5.0V \pm 10\%$ $I_{OH} = -25 \mu\text{A}$ $I_{OL} = 400 \mu\text{A}$ $I_{OH} = -10 \mu\text{A}$ $I_{OL} = 10 \mu\text{A}$	2.7 $V_{CC} - 0.2$	0.4 0.2	V V V V
Output Current Levels (Note 4) (Except CKO) Sink Source (Standard Option) Source (Low Current Option)	$V_{CC} = 4.5V, V_{OUT} = V_{CC}$ $V_{CC} = 2.4V, V_{OUT} = V_{CC}$ $V_{CC} = 4.5V, V_{OUT} = 0V$ $V_{CC} = 2.4V, V_{OUT} = 0V$ $V_{CC} = 4.5V, V_{OUT} = 0V$ $V_{CC} = 2.4V, V_{OUT} = 0V$	1.2 0.2 -0.5 -0.1 -30 -6	-330 -80	mA mA mA mA μA μA
CKO Current Levels (As Clock Out) Sink Source	$V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC}$ $V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V$	0.3 0.6 1.2 -0.3 -0.6 -1.2		mA mA mA mA mA mA
Allowable Sink/Source Current Per Pin (Note 4)			5	mA

COP410C/COP411C**DC Electrical Characteristics** (Continued)

Parameter	Conditions	Min	Max	Units
Allowable Loading on CKO (as HALT I/O pin)			100	pF
Current Needed to Override HALT ³				
To Continue	$V_{CC} = 4.5V, V_{IN} = 0.2 V_{CC}$		0.6	mA
To Halt	$V_{CC} = 4.5V, V_{IN} = 0.7 V_{CC}$		1.6	mA
TRI-STATE or Open Drain Leakage Current		-2	+2	μA

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating in the RC and crystal configurations.

Note 3: When forcing HALT, current is only needed for a short time (approximately 200 ns) to flip the HALT flip-flop.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 5: Voltage change must be less than 0.5V in a 1 ms period.

Note 6: This parameter is only sampled and not 100% tested.

Note 7: Variation due to the device included.

COP410C/COP411C**AC Electrical Characteristics** $0^{\circ}C \leq T_A \leq 70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)	$V_{CC} \geq 4.5V$ $4.5V > V_{CC} \geq 2.4V$	4 16	DC DC	μs μs
Operating CKI Frequency	$\left. \begin{array}{l} \div 4 \text{ mode} \\ \div 8 \text{ mode} \\ \div 16 \text{ mode} \\ \div 4 \text{ mode} \\ \div 8 \text{ mode} \\ \div 16 \text{ mode} \end{array} \right\} \begin{array}{l} V_{CC} \geq 4.5V \\ 4.5V > V_{CC} \geq 2.4V \end{array}$	DC	1.0	MHz
		DC	2.0	MHz
		DC	4.0	MHz
		DC	250	kHz
		DC	500	kHz
		DC	1.0	MHz
Instruction Cycle Time RC Oscillator ⁷	$R = 30k \pm 5\%, V_{CC} = 5V$ $C = 82 pF \pm 5\% (\div 4 \text{ Mode})$	8	16	μs
Duty Cycle ⁶	$f_l = 4 \text{ MHz}$	40	60	%
Rise Time ⁶	$f_l = 4 \text{ MHz External Clock}$		60	ns
Fall Time ⁶	$f_l = 4 \text{ MHz External Clock}$		40	ns
Inputs (See Figure 3)				
t_{SETUP}	$\left. \begin{array}{l} G \text{ Inputs} \\ SI \text{ Input} \\ \text{All Others} \end{array} \right\} V_{CC} \geq 4.5V$	$t_c/4 + 0.7$		μs
		0.3		μs
		1.7		μs
t_{HOLD}	$V_{CC} \geq 4.5V$ $V_{CC} \geq 2.4V$	0.25 1.0		μs μs
Output Propagation Delay				
t_{PD1}, t_{PD0}	$V_{OUT} = 1.5V, C_L = 100 pF, R_L = 5k$ $V_{CC} \leq 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$V_{CC} \leq 2.4V$		4.0	μs

COP310C/COP311C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA

Operating Temperature Range	-40°C to +85°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		3.0	5.5V	V
Power Supply Ripple (Notes 5, 6)			0.1 V_{CC}	V
Supply Current (Note 1)	$V_{CC} = 3.0V, t_c = 125 \mu\text{s}$ $V_{CC} = 5.0V, t_c = 16 \mu\text{s}$ $V_{CC} = 5.0V, t_c = 4 \mu\text{s}$ (t_c is instruction cycle time)		100 600 2500	μA μA μA
HALT Mode Current (Note 2)	$V_{CC} = 5.0V, F_{IN} = 0 \text{ kHz}$ $V_{CC} = 3.0V, F_{IN} = 0 \text{ kHz}$		50 20	μA μA
Input Voltage Levels				
RESET, CKI				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		-2	+2	μA
Input Capacitance (Note 6)			7	pF
Output Voltage Levels				
LSTTL Operation	Standard Outputs $V_{CC} = 5.0V \pm 10\%$			
Logic High	$I_{OH} = -25 \mu\text{A}$	2.7		V
Logic Low	$I_{OL} = 400 \mu\text{A}$		0.4	V
CMOS Operation				
Logic High	$I_{OH} = -10 \mu\text{A}$	$V_{CC} - 0.2$		V
Logic Low	$I_{OL} = 10 \mu\text{A}$		0.2	V
Output Current Levels (Note 4) (Except CKO)				
Sink	$V_{CC} = 4.5V, V_{OUT} = V_{CC}$ $V_{CC} = 3.0V, V_{OUT} = V_{CC}$	1.2 0.2		mA mA
Source (Standard Option)	$V_{CC} = 4.5V, V_{OUT} = 0V$ $V_{CC} = 3.0V, V_{OUT} = 0V$	-0.5 -0.1		mA mA
Source (Low Current Option)	$V_{CC} = 4.5V, V_{OUT} = 0V$ $V_{CC} = 3.0V, V_{OUT} = 0V$	-30 -8	-440 -200	μA μA
CKO Current Levels (As Clock Out)				
Sink	$V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC}$	0.3 0.6		mA mA
Source	$V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V$	1.2 -0.3 -0.6 -1.2		mA mA mA mA
Allowable Sink/Source Current Per Pin (Note 4)			5	mA

COP310C/COP311C

DC Electrical Characteristics (Continued)

Parameter	Conditions	Min	Max	Units
Allowable Loading on CKO (as HALT I/O pin)			100	pF
Current Needed to Override HALT ³ To Continue To Halt	$V_{CC} = 4.5V, V_{IN} = 0.2 V_{CC}$ $V_{CC} = 4.5V, V_{IN} = 0.7 V_{CC}$		0.8 2.0	mA mA
TRI-STATE or Open Drain Leakage Current		-4	+4	μA

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating in the RC and crystal configurations.

Note 3: When forcing HALT, current is only needed for a short time (approximately 200 ns) to flip the HALT flip-flop.

Note 4: SO output sink current must be limited to keep V_{OL} less than $0.2 V_{CC}$ when part is running in order to prevent entering test mode.

Note 5: Voltage change must be less than 0.5V in a 1 ms period.

Note 6: This parameter is only sampled and not 100% tested.

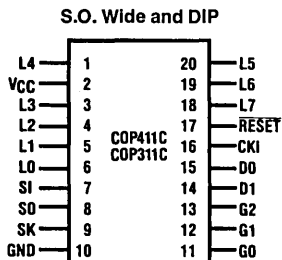
Note 7: Variation due to the device included.

COP310C/COP311C

AC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)	$V_{CC} \geq 4.5V$ $4.5V > V_{CC} \geq 3.0V$	4 16	DC DC	μs μs
Operating CKI Frequency ÷ 4 mode ÷ 8 mode ÷ 16 mode ÷ 4 mode ÷ 8 mode ÷ 16 mode	$V_{CC} \geq 4.5V$ $4.5V > V_{CC} \geq 3.0V$	DC DC DC DC DC DC	1.0 2.0 4.0 250 500 1.0	MHz MHz MHz kHz kHz MHz
Instruction Cycle Time RC Oscillator ⁷	$R = 30k \pm 5\%, V_{CC} = 5V$ $C = 82 pF \pm 5\% (\div 4 \text{ Mode})$	8	16	μs
Duty Cycle ⁶	$f_1 = 4 \text{ MHz}$	40	60	%
Rise Time ⁶	$f_1 = 4 \text{ MHz External Clock}$		60	ns
Fall Time ⁶	$f_1 = 4 \text{ MHz External Clock}$		40	ns
Inputs (See Figure 3) t_{SETUP} t_{HOLD}	G Inputs } SI Input } $V_{CC} \geq 4.5V$ All Others } $V_{CC} \geq 4.5V$ $V_{CC} \geq 3.0V$	$t_c/4 + 0.7$ 0.3 1.7 0.25 1.0		μs μs μs μs μs
Output Propagation Delay t_{PD1}, t_{PD0} t_{PD1}, t_{PD0}	$V_{OUT} = 1.5V, C_L = 100 pF, R_L = 5k$ $V_{CC} \leq 4.5V$ $V_{CC} \leq 3.0V$		1.0 4.0	μs μs

Connection Diagrams



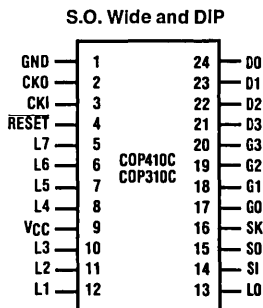
TL/DD/5015-2

Top View

Order Number COP311C-XXX/D or COP411C-XXX/D
See NS Hermetic Package Number D20A
(Prototype Package Only)

Order Number COP311C-XXX/N or COP411C-XXX/N
See NS Molded Package Number N20A

Order Number COP311C-XXX/WM or
COP411C-XXX/WM
See NS Surface Mount Package Number M20B



TL/DD/5015-3

Top View

Order Number COP310C-XXX/D or COP410C-XXX/D
See NS Hermetic Package Number D24C
(Prototype Package Only)

Order Number COP310C-XXX/N or COP410C-XXX/N
See NS Molded Package Number N24A

Order Number COP310C-XXX/WM or
COP410C-XXX/WM
See NS Surface Mount Package Number M24B

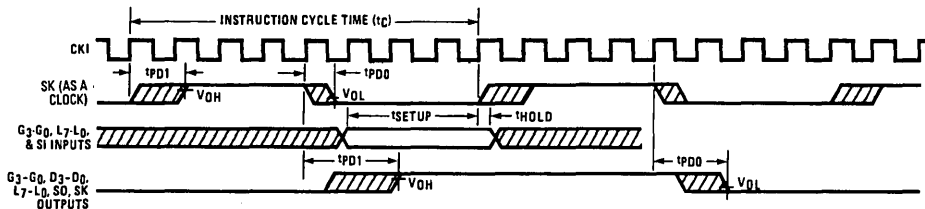
FIGURE 2

Pin Descriptions

Pin	Description
L ₇ -L ₀	8-bit bidirectional I/O port with TRI-STATE
G ₃ -G ₀	4-bit bidirectional I/O port (G ₂ -G ₀ for 20-pin package)
D ₃ -D ₀	4-bit general purpose output port (D ₁ -D ₀ for 20-pin package)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)

Pin	Description
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	Crystal oscillator output, or HALT mode I/O port (24-pin package only)
RESET	System reset input
V _{CC}	System power supply
GND	System Ground

Timing Diagram



TL/DD/5015-4

FIGURE 3. Input/Output (Divide-by-8 Mode)

Functional Description

To ease reading of this description, only COP410C and/or COP411C are referenced; however, all such references apply equally to COP310C and/or COP311C, respectively.

A block diagram of the COP410C is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP410C/411C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8×4 -bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4*. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP410C/411C is designed to ensure fully static operation of the device.

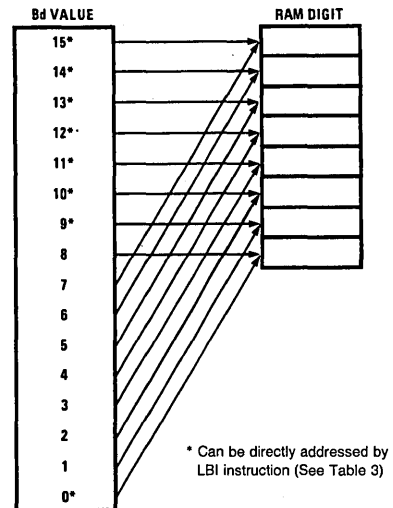
The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP410C/411C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.



LL/DD/5015-5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP410C/411C is MICROWIRE compatible.

The D register provides four general purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.
2. EN 1 is not used, it has no effect on the COP410C/411C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

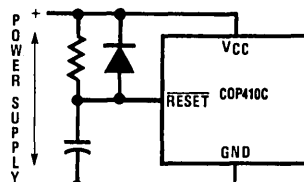
INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC} . Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

When V_{CC} power is applied, the internal reset logic will keep the chip in initialization mode for up to 2500 instruction cycles. If the CKI clock is running at a low frequency, this could take a long time, therefore, the internal logic should be disabled by a mask option with initialization controlled solely by $\overline{\text{RESET}}$ pin.

Note: If CKI clock is less than 32 kHz, the internal reset logic (Option 25 = 1) *must* be disabled and the external RC network *must* be present.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/5015-6

$RC > 5 \times \text{Power Supply Rise Time}$
and $RC > 100 \times \text{CKI Period}$

FIGURE 5. Power-Up Clear Circuit

COP411C

If the COP410C is bonded as a 20-pin package, it becomes the COP411C, illustrated in *Figure 2*, COP410C/411C Connection Diagrams. Note that the COP411C does not contain D2, D3, G3, or CKO. Use of this option, of course, precludes use of D2, D3, G3, and CKO options. All other options are available for the COP411C.

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

HALT MODE

The COP410C/411C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip also may be halted by the HALT instruction or by forcing CKO high when it is used as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode has slight differences depending upon the type of oscillator used.

a. 1-pin oscillator—RC or external

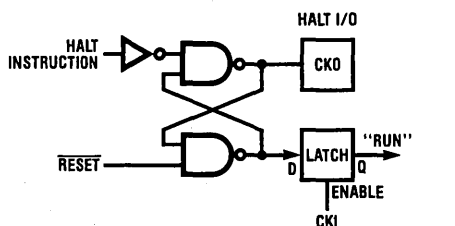
The HALT mode may be entered into by either program control (HALT instruction) or by forcing CKO to a logic "1" state.

The circuit may be awakened by one of two different methods:

- 1) Continue function. By forcing CKO to a logic "0", the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- 2) Restart. Forcing the $\overline{\text{RESET}}$ pin to a logic "0" will restart the chip regardless of HALT or CKO (see initialization).

b. 2-pin oscillator—crystal

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.



TL/DD/5015-7

Halt I/O Port

CKO Pin Options

In a crystal-controlled oscillator system, CKO is used as an output to the crystal network. CKO will be forced high during the execution of a HALT instruction, thus inhibiting the crystal network. If a 1-pin oscillator system is chosen (RC or external), CKO will be selected as HALT and is an I/O

flip-flop which is an indicator of the HALT status. An external signal can override this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and the CKO output will go high to keep the chip stopped. By forcing a low level to CKO, the chip will continue and CKO output will go low.

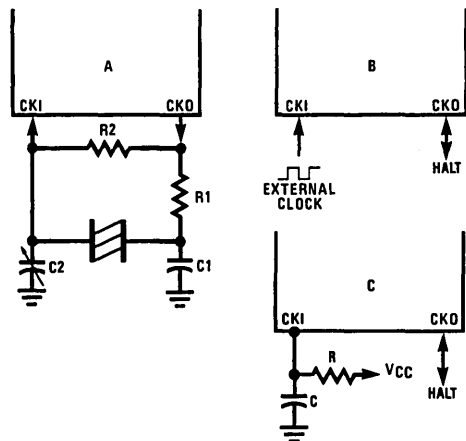
All features associated with the CKO I/O pin are available with the 24-pin package only.

OSCILLATOR OPTIONS

There are three options available that define the use of CKI and CKO.

- a. Crystal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optionally by 8 or 4).
- b. External Oscillator. CKI is configured as LSTTL-compatible input accepting an external clock signal. The external frequency is divided by 16 (optionally by 8 or 4) to give the instruction cycle time. CKO is the HALT I/O port.
- c. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle time equals the oscillation frequency divided by 4. CKO is the HALT I/O port.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.



TL/DD/5015-8

FIGURE 6. COP410C Oscillator

Crystal or Resonator

Crystal Value	Component Value				RC-Controlled Oscillator			
	R1	R2	C1 pF	C2 pF	R	C	Cycle Time	V _{CC}
32 kHz	220k	20M	30	5-36	15k	82 pF	4-9 μs	$\geq 4.5\text{V}$
455 kHz	5k	10M	80	40	30k	82 pF	8-16 μs	$\geq 4.5\text{V}$
2.096 MHz	2k	1M	30	6-36	47k	100 pF	16-32 μs	2.4 to 4.5
4.0 MHz	1k	1M	30	6-36	Note: $15\text{k} \leq R \leq 150\text{k}$, $50\text{ pf} \leq C \leq 150\text{ pF}$			

COP410C/COP411C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410C/411C instruction set.

TABLE II. COP410C/411C Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	2-bit Operand Field, 0-3 binary (RAM Register Select)
Br	Upper 2 bits of B (register address)	a	9-bit Operand Field, 0-511 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0-15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register	OPERATIONAL SYMBOLS	
G	4-bit Register to latch data for G I/O Port	+	Plus
L	8-bit TRI-STATE I/O Port	-	Minus
M	4-bit contents of RAM Memory pointed to by B Register	→	Replaces
PC	9-bit ROM Address Register (program counter)	↔	Is exchanged with
Q	8-bit Register to latch data for L I/O Port	=	Is equal to
SA	9-bit Subroutine Save Register A	\bar{A}	The one's complement of A
SB	9-bit Subroutine Save Register B	⊕	Exclusive-OR
SIO	4-bit Shift Register and Counter	:	Range of values
SK	Logic-Controlled Clock Output		

TABLE III. COP410C/411C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5-	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add immediate, Skip on Carry ($y \neq 0$)
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP410C/411C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC ₈ , A, M) → PC _{7:0}	None	Jump Indirect (Note 2)
JMP	a	6- -	0110 000 a ₈ a _{7:0}	a → PC	None	Jump
JP	a	-	1 a _{6:0} (pages 2,3 only) or 11 a _{5:0} (all other pages)	a → PC _{6:0} a → PC _{5:0}	None	Jump within Page (Note 1)
JSRP	a	-	10 a _{5:0}	PC + 1 → SA → SB 010 → PC _{8:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 2)
JSR	a	6- -	0110 100 a ₈ a _{7:0}	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 10011	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	0011 0011 0011 1000		None	Halt processor
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → Q _{7:4} RAM(B) → Q _{3:0}	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	Q _{7:4} → RAM(B) Q _{3:0} → A	None	Copy Q to RAM, A
LD	r	-5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	1011 1111	ROM(PC ₈ , A, M) → Q SA → SB	None	Load Q Indirect
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → RAM(B) ₀ 0 → RAM(B) ₁ 0 → RAM(B) ₂ 0 → RAM(B) ₃	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → RAM(B) ₀ 1 → RAM(B) ₁ 1 → RAM(B) ₂ 1 → RAM(B) ₃	None	Set RAM Bit
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	0010 0011 1011 1111	RAM(3,15) ↔ A	None	Exchange A with RAM (3,15)

Instruction Set (Continued)

TABLE III. COP410C/411C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	00 r 0111	RAM(B) ← A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	-4	00 r 0100	RAM(B) ← A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	-	00 r (d-1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6-	0011 0011 0010 y	y → EN	None	Load EN Immediate
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
		21	0010 0001			
SKGBZ		33	0011 0011	1st byte	G ₀ = 0	Skip if G Bit is Zero
	0	01	0000 0001			
	1	11	0001 0001			
	2	03	0000 0011			
		3	0010 0011	2nd byte	G ₃ = 0	
		13	0010 0011			
SKMBZ	0	01	0000 0001		RAM(B) ₀ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001			
	2	03	0000 0011			
	3	13	0001 0011			
					RAM(B) ₁ = 0	
					RAM(B) ₂ = 0	
					RAM(B) ₃ = 0	
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
		2E	0010 1110			
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP410C/411C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register). If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC₈, A, M. PC₈ is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC₈, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → PC_{7:4}, RAM(B) → PC_{3:0}, leaving PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

INSTRUCTION SET NOTES

- The first word of a COP410C/411C program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).
- The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw additional current. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP410C current drain.

$$I_c = I_q + (V \times 20 \times F_i) + (V \times 1280 \times F_i/D_v)$$

where I_c = chip current drain in microamps

I_q = quiescent leakage current (from curve)

F_i = CKI frequency in megahertz

V = chip V_{CC} in volts

D_v = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 4),

$$I_c = 10 + (5 \times 20 \times 0.4) + (5 \times 1280 \times 0.4/4)$$

$$I_c = 10 + 40 + 640 = 690 \mu A$$

I/O OPTIONS

COP410C/411C outputs have the following optional configurations, illustrated in *Figure 7*:

- Standard. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC} , compatible with CMOS and LSTTL.
- Low Current. This is the same configuration as (a) above except that the sourcing current is much less.
- Open Drain. An N-channel device to ground only, allowing external pull-up as required by the user's application.
- Standard TRI-STATE L Output. A CMOS output buffer similar to (a) which may be disabled by program control.
- Low-Current TRI-STATE L Output. This is the same as (d) above except that the sourcing current is much less.
- Open-Drain TRI-STATE L Output. This has the N-channel device to ground only.

The SI and RESET inputs are Hi-Z inputs (*Figure 7g*).

When using either the G or L I/O ports as inputs, a pull-up device is necessary. This can be an external device or the following alternative is available: Select the low-current output option. Now, by setting the output registers to a logic "1" level, the P-channel devices will act as the pull-up load. Note that when using the L ports in this fashion, the Q registers must be set to a logic "1" level and the L drivers *must be enabled* by an LEI instruction.

Functional Description (Continued)

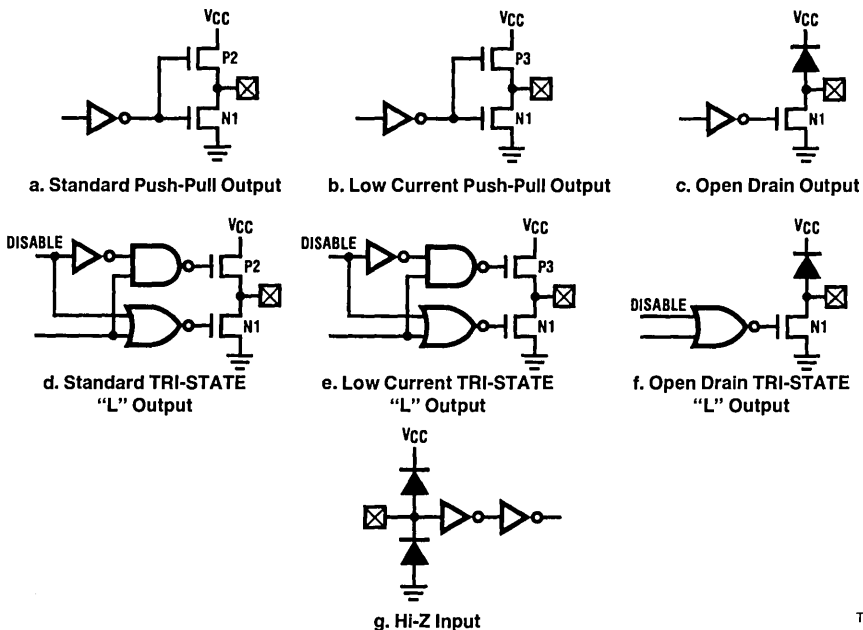


FIGURE 7. I/O Configurations

TL/DD/5015-9

Typical Performance Characteristics

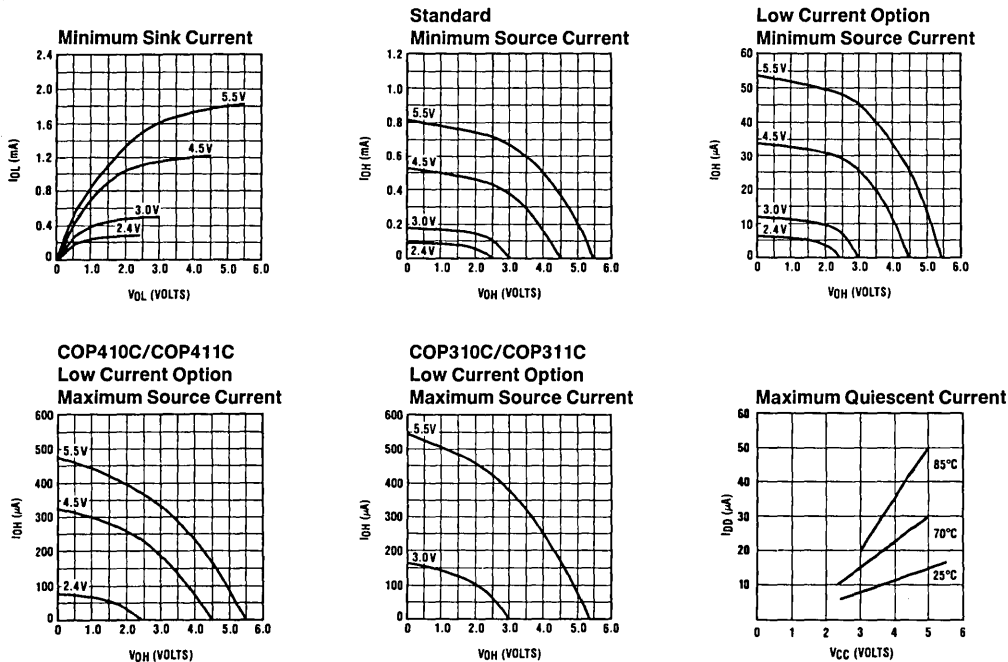


FIGURE 8

TL/DD/5015-10

All output drivers uses one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.

Option List

The COP410C/411C mask-programmable options are assigned numbers which correspond with the COP410C pins. The following is a list of COP410C options. When specifying a COP411 chip, options 20, 21, and 22 must be set to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

- Option 1: 0 = Ground Pin. No options available.
- Option 2: CKO I/O Port. (Determined by Option 3.)
= 0: No option.
(a. is crystal oscillator output for two pin oscillator.
b. is HALT I/O for one pin oscillator.)
- Option 3: CKI Input.
= 0: Crystal-controlled oscillator input ($\div 4$).
= 1: Single-pin RC-controlled oscillator ($\div 4$).
= 2: External oscillator input ($\div 4$).
= 3: Crystal oscillator input ($\div 8$).
= 4: External oscillator input ($\div 8$).
= 5: Crystal oscillator input ($\div 16$).
= 6: External oscillator input ($\div 16$).
- Option 4: RESET Input = 1: Hi-Z input. No option available.
- Option 5: L₇ Driver
= 0: Standard TRI-STATE push-pull output.
= 1: Low-current TRI-STATE push-pull output.
= 2: Open-drain TRI-STATE output.

- Option 6: L₆ Driver. (Same as Option 5.)
- Option 7: L₅ Driver. (Same as Option 5.)
- Option 8: L₄ Driver. (Same as Option 5.)
- Option 9: V_{CC} Pin = 0 no option.
- Option 10: L₃ Driver. (Same as Option 5.)
- Option 11: L₂ Driver. (Same as Option 5.)
- Option 12: L₁ Driver. (Same as Option 5.)
- Option 13: L₀ Driver. (Same as Option 5.)
- Option 14: SI Input.
No option available.
= 1: Hi-Z input.
- Option 15: SO Output.
= 0: Standard push-pull output.
= 1: Low-current push-pull output.
= 2: Open-drain output.
- Option 16: SK Driver. (Same as Option 15.)
- Option 17: G₀ I/O Port. (Same as Option 15.)
- Option 18: G₁ I/O Port. (Same as Option 15.)
- Option 19: G₂ I/O Port. (Same as Option 15.)
- Option 20: G₃ I/O Port. (Same as Option 15.)
- Option 21: D₃ Output. (Same as Option 15.)
- Option 22: D₂ Output. (Same as Option 15.)
- Option 23: D₁ Output. (Same as Option 15.)
- Option 24: D₀ Output. (Same as Option 15.)
- Option 25: Internal Initialization Logic.
= 0: Normal operation.
= 1: No internal initialization logic.
- Option 26: No option available.
- Option 27: COP Bonding
= 0: COP410C (24-pin device).
= 1: COP411C (20-pin device). See note.
= 2: COP410C and COP411C. See note.

Note: If opt. #27 = 1 or 2 then opt #20 must = 0.

Option Table

Please fill out a photocopy of the option table and send it along with your EPROM.

Option Table

Option 1 Value = <u> 0 </u>	is: Ground Pin	Option 15 Value = <u> </u>	is: SO Output
Option 2 Value = <u> 0 </u>	is: CKO Pin	Option 16 Value = <u> </u>	is: SK Driver
Option 3 Value = <u> </u>	is: CKI Input	Option 17 Value = <u> </u>	is: G ₀ I/O Port
Option 4 Value = <u> 1 </u>	is: RESET Input	Option 18 Value = <u> </u>	is: G ₁ I/O Port
Option 5 Value = <u> </u>	is: L ₇ Driver	Option 19 Value = <u> </u>	is: G ₂ I/O Port
Option 6 Value = <u> </u>	is: L ₆ Driver	Option 20 Value = <u> </u>	is: G ₃ I/O Port
Option 7 Value = <u> </u>	is: L ₅ Driver	Option 21 Value = <u> </u>	is: D ₃ Output
Option 8 Value = <u> </u>	is: L ₄ Driver	Option 22 Value = <u> </u>	is: D ₂ Output
Option 9 Value = <u> 0 </u>	is: V _{CC} Pin	Option 23 Value = <u> </u>	is: D ₁ Output
Option 10 Value = <u> </u>	is: L ₃ Driver	Option 24 Value = <u> </u>	is: D ₀ Output
Option 11 Value = <u> </u>	is: L ₂ Driver	Option 25 Value = <u> </u>	is: Internal
Option 12 Value = <u> </u>	is: L ₁ Driver		Initialization
Option 13 Value = <u> </u>	is: L ₀ Driver		Logic
Option 14 Value = <u> 1 </u>	is: SI Input	Option 26 Value = <u> 0 </u>	is: N/A
		Option 27 Value = <u> </u>	is: COP Bonding

COP410L/COP411L/COP310L/COP311L Single-Chip N-Channel Microcontrollers

General Description

The COP410L and COP411L Single-Chip N-Channel Microcontrollers are members of the COPSTM family, fabricated using N-channel, silicon gate MOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP411L is identical to the COP410L, but with 16 I/O lines instead of 19. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

The COP310L and COP311L are exact functional equivalents but extended temperature versions of COP410L and COP411L respectively.

The COP401L should be used for exact emulation.

Features

- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 19 I/O lines (COP410L)
- Two-level subroutine stack
- 16 μ s instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device
— COP310L/COP311L (–40°C to +85°C)

Block Diagram

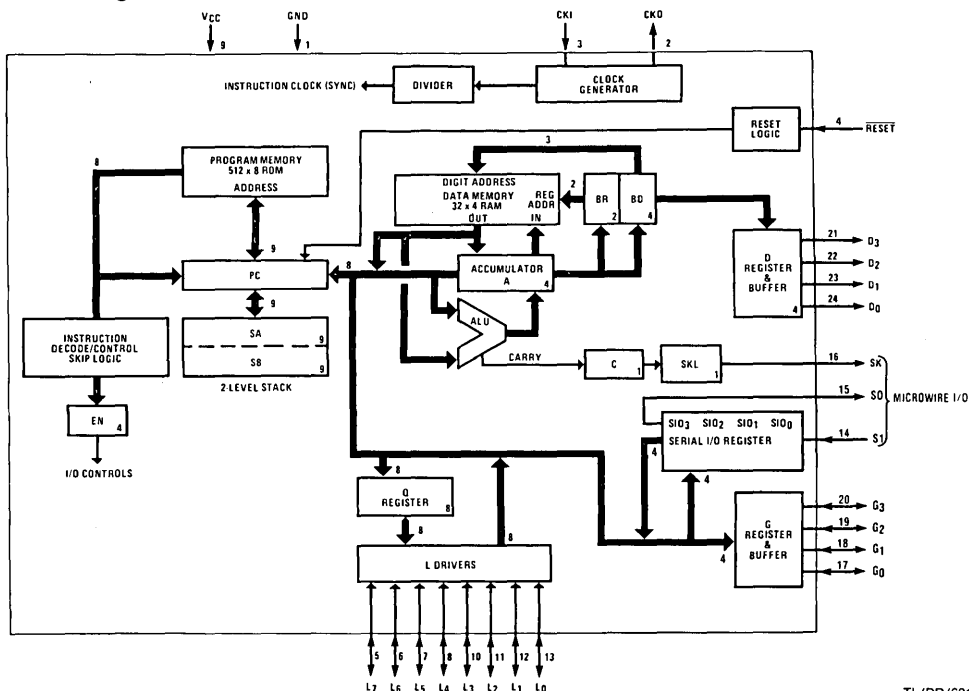


FIGURE 1. COP410L

TL/DD/6919-1

COP410L/COP411L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Power Dissipation
COP410L

0.75W at 25°C
0.4W at 70°C

COP411L

0.65W at 25°C
0.3W at 70°C

Total Source Current

120 mA

Total Sink Current

100 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	6.3	V
Power Supply Ripple (Notes 1, 4)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷ 8)				
Logic High (V _{IH})	V _{CC} = Max	3.0		V
Logic High (V _{IH})	V _{CC} = 5V ± 5%	2.0		V
Logic Low (V _{IL})		-0.3	0.4	V
Schmitt Trigger Input (÷ 4)				
Logic High (V _{IH})		0.7 V _{CC}		V
Logic Low (V _{IL})		-0.3	0.6	V
RESET Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	2.5	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V _{CC} = 5V ± 5%	-0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 4)			7	pF
Hi-Z Input Leakage		-1	+1	μA
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ± 10%			
Logic High (V _{OH})	I _{OH} = -25 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 3)				
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

Note 3: TRI-STATE® and LED configurations are excluded.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

COP410L/COP411L

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 6.3\text{V}$, $V_{OL} = 0.4\text{V}$	1.2		mA
	$V_{CC} = 4.5\text{V}$, $V_{OL} = 0.4\text{V}$	0.9		mA
L_0 - L_7 Outputs, G_0 - G_3 and LSTTL D_0 - D_3 Outputs (I_{OL})	$V_{CC} = 6.3\text{V}$, $V_{OL} = 0.4\text{V}$	0.4		mA
	$V_{CC} = 4.5\text{V}$, $V_{OL} = 0.4\text{V}$	0.4		mA
D_0 - D_3 Outputs with High Current Options (I_{OL})	$V_{CC} = 6.3\text{V}$, $V_{OL} = 1.0\text{V}$	11		mA
	$V_{CC} = 4.5\text{V}$, $V_{OL} = 1.0\text{V}$	7.5		mA
D_0 - D_3 Outputs with Very High Current Options (I_{OL})	$V_{CC} = 6.3\text{V}$, $V_{OL} = 1.0\text{V}$	22		mA
	$V_{CC} = 4.5\text{V}$, $V_{OL} = 1.0\text{V}$	15		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5\text{V}$, $V_{IH} = 3.5\text{V}$	2		mA
CKO	$V_{CC} = 4.5\text{V}$, $V_{OL} = 0.4\text{V}$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{CC} = 6.3\text{V}$, $V_{OH} = 2.0\text{V}$	-75	-480	μA
	$V_{CC} = 4.5\text{V}$, $V_{OH} = 2.0\text{V}$	-30	-250	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	$V_{CC} = 6.3\text{V}$, $V_{OH} = 2.4\text{V}$	-1.4		mA
	$V_{CC} = 4.5\text{V}$, $V_{OH} = 1.0\text{V}$	-1.2		mA
LED Configuration, L_0 - L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 6.0\text{V}$, $V_{OH} = 2.0\text{V}$	-1.5	-13	mA
LED Configuration, L_0 - L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 6.0\text{V}$, $V_{OH} = 2.0\text{V}$	-3.0	-25	mA
TRI-STATE Configuration, L_0 - L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 6.3\text{V}$, $V_{OH} = 3.2\text{V}$	-0.8		mA
	$V_{CC} = 4.5\text{V}$, $V_{OH} = 1.5\text{V}$	-0.9		mA
TRI-STATE Configuration, L_0 - L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 6.3\text{V}$, $V_{OH} = 3.2\text{V}$	-1.6		mA
	$V_{CC} = 4.5\text{V}$, $V_{OH} = 1.5\text{V}$	-1.8		mA
Input Load Source Current	$V_{CC} = 5.0\text{V}$, $V_{IL} = 0\text{V}$	-10	-140	μA
CKO Output				
RAM Power Supply Option Power Requirement	$V_R = 3.3\text{V}$		1.5	mA
TRI-STATE Output Leakage Current				
		-2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7 - L_4 , G Port			4	mA
L_3 - L_0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7 - L_4			60	mA
L_3 - L_0			60	mA
Each L Pin			25	mA
Any Other Pin			1.5	mA

COP310L/COP311L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	-40°C to +85°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Power Dissipation

COP310L	0.75W at 25°C 0.25W at 85°C
COP311L	0.65W at 25°C 0.20W at 85°C

Total Source Current 120 mA

Total Sink Current 100 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	5.5	V
Power Supply Ripple (Notes 1, 4)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Crystal Input				
Logic High (V _{IH})	V _{CC} = Max	3.0		V
Logic High (V _{IH})	V _{CC} = 5V ±5%	2.2		V
Logic Low (V _{IL})		-0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V _{IH})		0.7 V _{CC}		V
Logic Low (V _{IL})		-0.3	0.4	V
RESET Input Levels (Schmitt Trigger Input)				
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.2	2.5	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V _{CC} = 5V ±5%	-0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 4)			7	pF
Hi-Z Input Leakage		-2	+2	μA
Output Voltage Levels				
LSTTL Operation				
Logic High (V _{OH})	V _{CC} = 5V ±10%	2.7		V
Logic Low (V _{OL})	I _{OH} = -20 μA I _{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 3)				
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

Note 3: TRI-STATE and LED configurations are excluded.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

COP310L/COP311L**DC Electrical Characteristics** (Continued)-40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	V _{CC} = 5.5V, V _{OL} = 0.4V	1.0		mA
	V _{CC} = 4.5V, V _{OL} = 0.4V	0.8		mA
L ₀ -L ₇ Outputs, G ₀ -G ₃ and LSTTL D ₀ -D ₃ Outputs (I _{OL})	V _{CC} = 5.5V, V _{OL} = 0.4V	0.4		mA
	V _{CC} = 4.5V, V _{OL} = 0.4V	0.4		mA
D ₀ -D ₃ Outputs with High Current Options (I _{OL})	V _{CC} = 5.5V, V _{OL} = 1.0V	9		mA
	V _{CC} = 4.5V, V _{OL} = 1.0V	7		mA
D ₀ -D ₃ Outputs with Very High Current Options (I _{OL})	V _{CC} = 5.5V, V _{OL} = 1.0V	18		mA
	V _{CC} = 4.5V, V _{OL} = 1.0V	14		mA
CKI (Single-Pin RC Oscillator)	V _{CC} = 4.5V, V _{IH} = 3.5V	1.5		mA
CKO	V _{CC} = 4.5V, V _{OL} = 0.4V	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-55	-600	μA
	V _{CC} = 4.5V, V _{OH} = 2.0V	-28	-350	μA
Push-Pull Configuration SO and SK Outputs (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.1		mA
	V _{CC} = 4.5V, V _{OH} = 1.0V	-1.2		mA
LED Configuration, L ₀ -L ₇ Outputs, Low Current Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-0.7	-15	μA
LED Configuration, L ₀ -L ₇ Outputs, High Current Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.4	-30	μA
TRI-STATE Configuration, L ₀ -L ₇ Outputs, Low Current Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.7V	-0.6		mA
	V _{CC} = 4.5V, V _{OH} = 1.5V	-0.9		mA
TRI-STATE Configuration, L ₀ -L ₇ Outputs, High Current Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.7V	-1.2		mA
	V _{CC} = 4.5V, V _{OH} = 1.5V	-1.8		mA
Input Load Source Current	V _{CC} = 5.0V, V _{IL} = 0V	-10	-200	μA
CKO Output RAM Power Supply Option Power Requirement	V _R = 3.3V		2.0	mA
TRI-STATE Output Leakage Current		-5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L ₇ -L ₄ , G Port			4	mA
L ₃ -L ₀			4	mA
Any Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L ₇ -L ₄			60	mA
L ₃ -L ₀			60	mA
Each L Pin			25	mA
Any Other Pins			1.5	mA

AC Electrical Characteristics

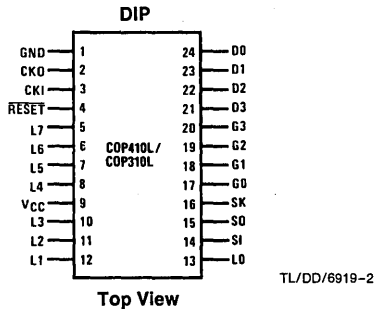
COP410L/411L: $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted

COP310L/311L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ unless otherwise noted

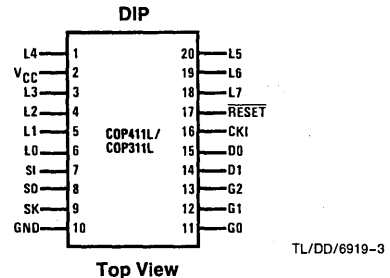
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time — t_C		16	40	μs
CKI				
Input Frequency — f_i	$\div 8$ Mode	0.2	0.5	MHz
	$\div 4$ Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time (Note 1)	$f_i = 0.5$ MHz		500	ns
Fall Time (Note 1)			200	ns
CKI Using RC ($\div 4$) (Note 1)	$R = 56\text{ k}\Omega \pm 5\%$ $C = 100\text{ pF} \pm 10\%$			
Instruction Cycle Time		16	28	μs
CKO as SYNC Input t_{SYNC}		400		ns
INPUTS				
G_3-G_0, L_7-L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: $C_L = 50\text{ pF}, R_L = 20\text{ k}\Omega, V_{\text{OUT}} = 1.5\text{V}$			
SO, SK Outputs $t_{\text{pd1}}, t_{\text{pd0}}$			4.0	μs
All Other Outputs $t_{\text{pd1}}, t_{\text{pd0}}$			5.6	μs

Note 1: This parameter is only sampled and not 100% tested.

Connection Diagrams



Top View
 Order Number COP310L-XXX/D or COP410L-XXX/D
 See NS Hermetic Package Number D24C
 (D Pkg.—for Prototypes Only)
 Order Number COP310L-XXX/N or COP410L-XXX/N
 See NS Molded Package Number N24A



Top View
 Order Number COP311L-XXX/D or COP411L-XXX/D
 See NS Hermetic Package Number D20A
 (D Pkg.—for Prototypes Only)
 Order Number COP311L-XXX/N or COP411L-XXX/N
 See NS Molded Package Number N20A

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L_7-L_0	8 bidirectional I/O ports with TRI-STATE	CKI	System oscillator input
G_3-G_0	4 bidirectional I/O ports (G_2-G_0 for COP411L)	CKO	System oscillator output (or RAM power supply or SYNC input) (COP410L only)
D_3-D_0	4 general purpose outputs (D_1-D_0 for COP411L)	RESET	System reset input
SI	Serial input (or counter input)	VCC	Power supply
SO	Serial output (or general purpose output)	GND	Ground
SK	Logic-controlled clock (or general purpose output)		

Timing Diagrams

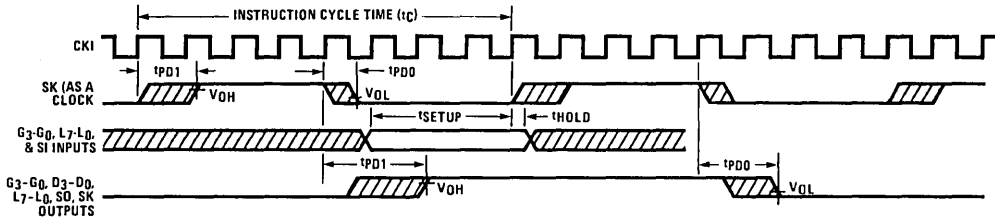


FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)

TL/DD/6919-4

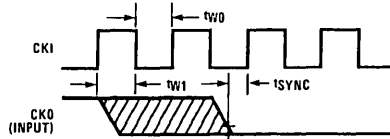


FIGURE 3a. Synchronization Timing

TL/DD/6919-5

Functional Description

A block diagram of the COP410L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP410L/COP411L also apply to the COP310L/COP311L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP410L/411L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it

may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs. The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).

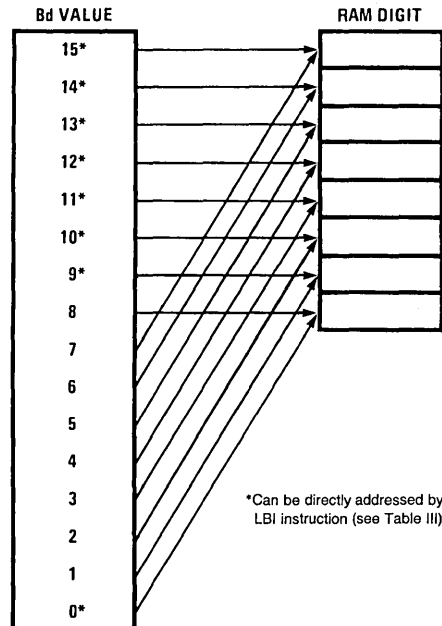


FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

TL/DD/6919-6

Functional Description (Continued)

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the B_d portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP410L/411L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃–EN₀).

1. The least significant bit of the enable register, EN₀, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, *decrementing* its value by one upon

each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.

2. EN₁ is not used. It has no effect on COP410L/COP411L operation.
3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN₃ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." Table I provides a summary of the modes associated with EN₃ and EN₀.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below (Figure 5). The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.

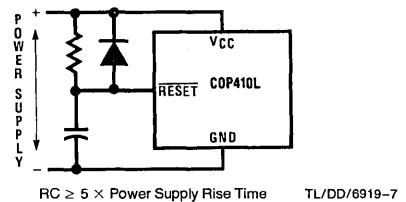


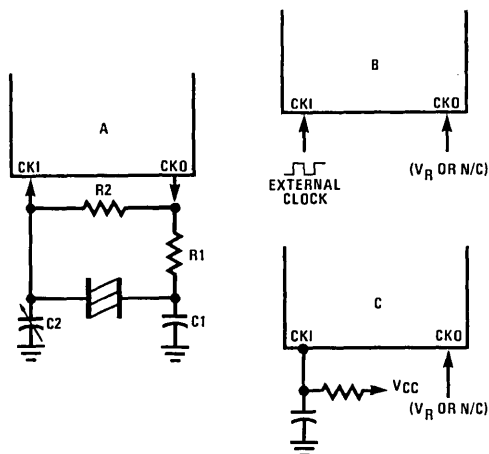
FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN₃ and EN₀

EN ₃	EN ₀	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/6919-8

Ceramic Resonator Oscillator

Resonator Value	Components Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (k Ω)	C (pF)	Instruction Cycle Time in μ s
51	100	19 \pm 15%
82	56	19 \pm 13%

Note: 200 k Ω \geq R \geq 25 k Ω . 360 pF \geq C \geq 50 pF. Does not include tolerances.

FIGURE 6. COP410L/411L Oscillator

OSCILLATOR

There are three basic clock oscillator configurations available as shown by Figure 6.

- Resonator Controlled Oscillator.** CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8. This is not available in the COP411L.
- External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 4 to give the instruction frequency time. CKO is now available to be used as the RAM power supply (V_R), or no connection.

Note: No CKO on COP411L.

- RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or no connection.

CKO PIN OPTIONS

In a resonator controlled oscillator system, CKO is used as an output to the resonator network. As an option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using no connection option is appropriate in applications where the COP410L system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

- \overline{RESET} must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before \overline{RESET} goes high on power-up.
- During normal operation, V_R must be within the operating range of the chip with $(V_{CC} - 1) \leq V_R \leq V_{CC}$.
- V_R must be $\geq 3.3V$ with V_{CC} off.

I/O OPTIONS

COP410L/411L inputs and outputs have the following optional configurations, illustrated in Figure 7:

- Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC} , compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.
 - Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.
 - Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.
 - Standard L**—same as a., but may be disabled. Available on L outputs only.
 - Open Drain L**—same as b., but may be disabled. Available on L outputs only.
 - LED Direct Drive**—an enhancement mode device to ground and to V_{CC} , meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.
- Note: Series current limiting resistors must be used if LEDs are driven directly and higher operating voltage option is selected.
- TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC} . These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

Functional Description (Continued)

h. An on-chip depletion load device to V_{CC} .

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP410L/411L system.

The SO, SK outputs can be configured as shown in a., b., or c. The D and G outputs can be configured as shown in a. or b. Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in d., e., f., or g.

An important point to remember if using configuration d. or f. with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 8*, device 2.) However, when the L port is used as input, the disabled depletion device CANNOT be relied on to source sufficient current to pull an input to a logic "1".

COP411L

If the COP410L is bonded as a 20-pin device, it becomes the COP411L, illustrated in *Figure 2*, COP410L/411L Connection Diagrams. Note that the COP411L does not contain D2, D3, G3, or CKO. Use of this option of course precludes use of D2, D3, G3, and CKO options. All other options are available for the COP411L.

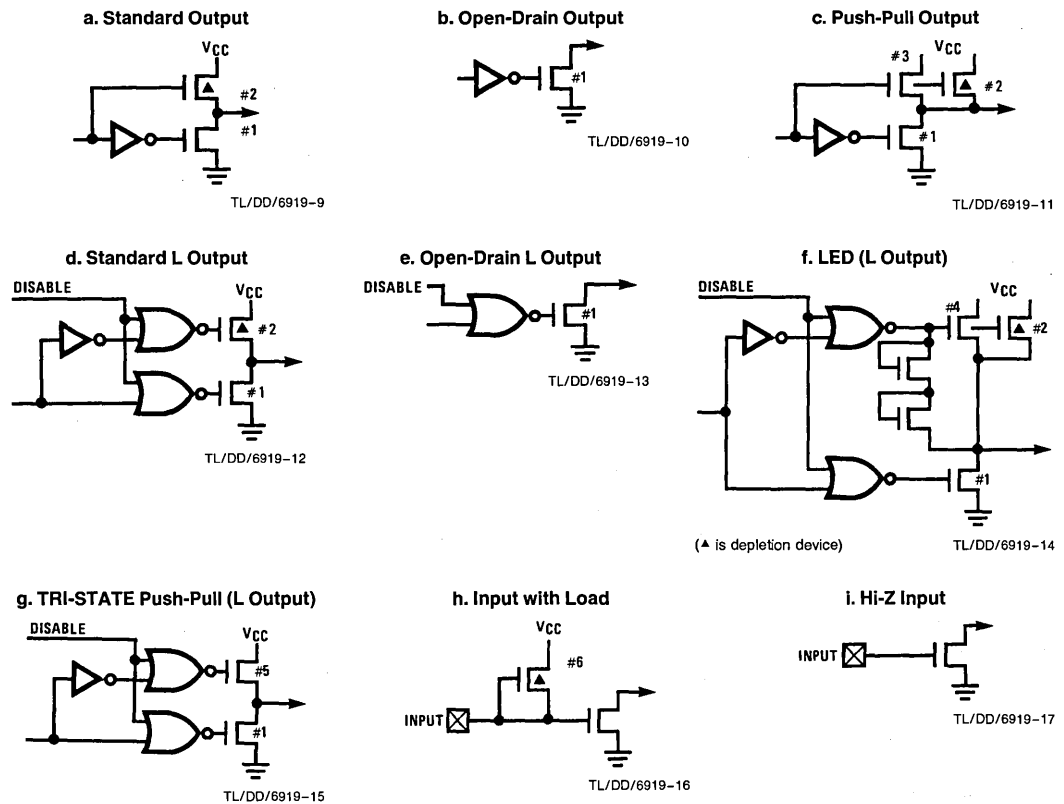


FIGURE 7. Input and Output Configurations

L-Bus Considerations

False states may be generated on L₀–L₇ during the execution of the CAMQ instruction. The L-ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. the following short program that illustrates this situation.

START:

```
CLRA           ;ENABLE THE Q
LEI 4         ;REGISTER TO L LINES
LBI TEST
STII 3
AISC 12
```

LOOP:

```
LBI TEST ;LOAD Q WITH X'C3
CAMQ
JP LOOP
```

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L₀, L₁, L₆, L₇, and logic lows on L₂–L₅ via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L₀, L₁, L₆, L₇, and positive glitches on L₂–L₅. Glitch durations are under 2 μ s, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics

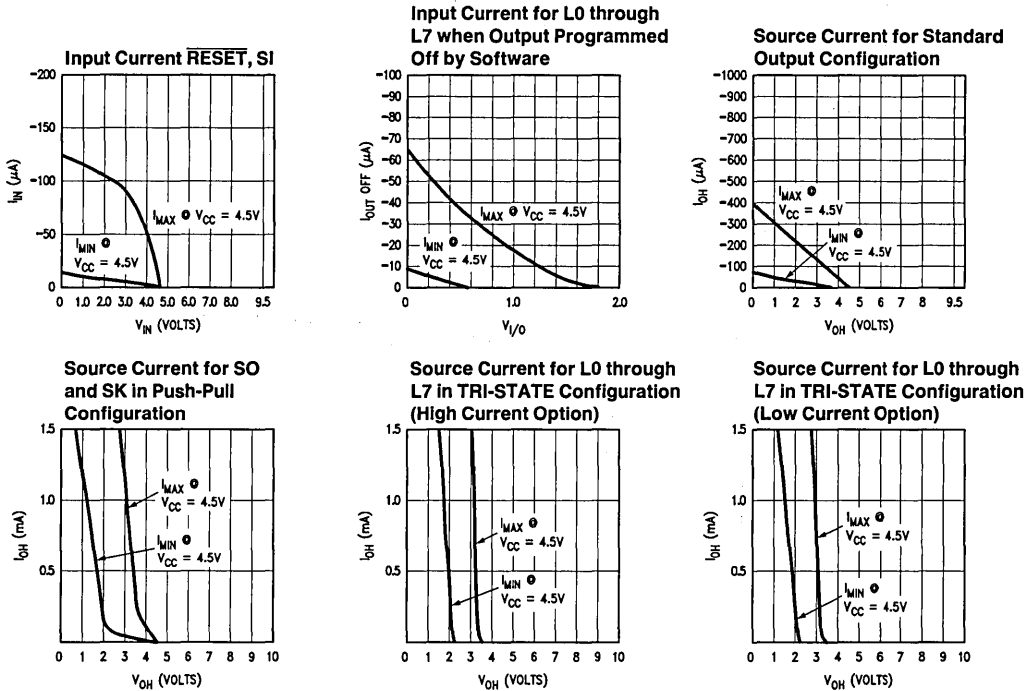


FIGURE 8a. COP410L/COP411L I/O DC Current Characteristics

TL/DD/6919-18

Typical Performance Characteristics (Continued)

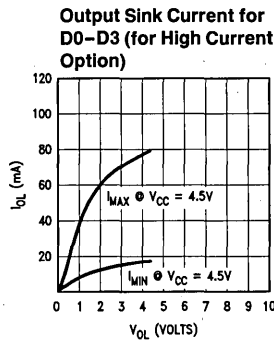
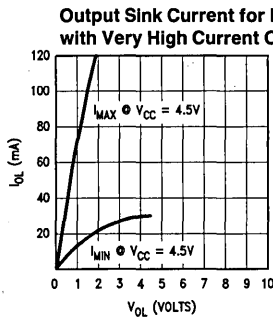
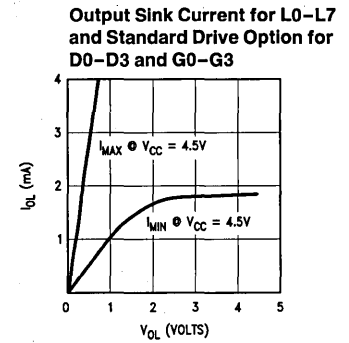
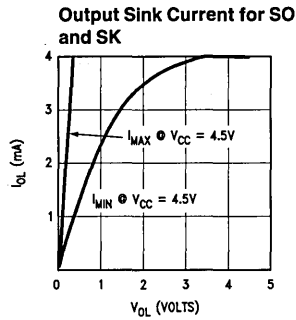
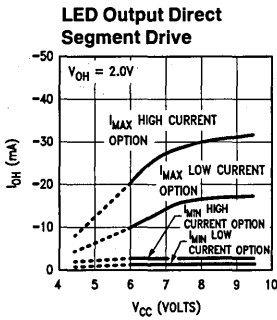
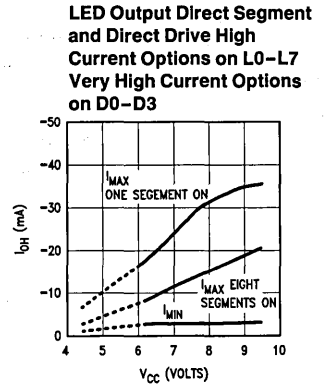
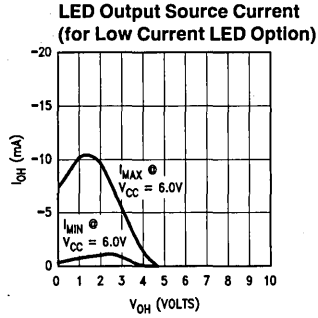
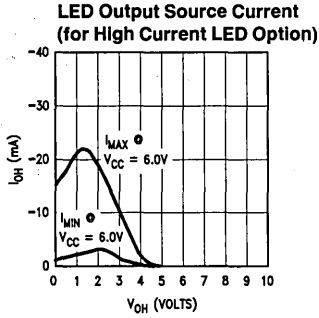
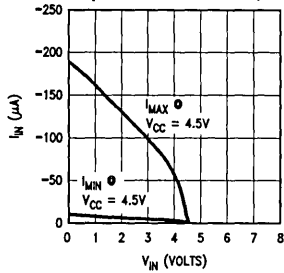


FIGURE 8a. COP410L/COP411L I/O DC Current Characteristics (Continued)

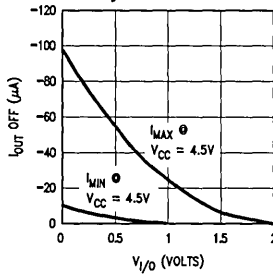
TL/DD/6919-19

Typical Performance Characteristics (Continued)

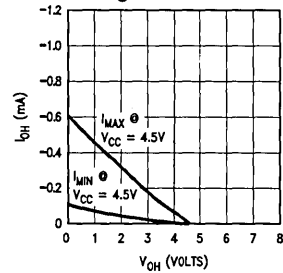
Input Current $\overline{\text{RESET}}$, SI



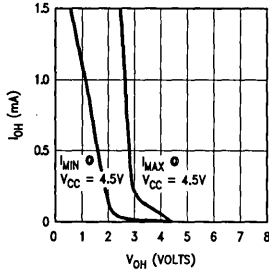
Input Current for L0-L7 when Output Programmed Off by Software



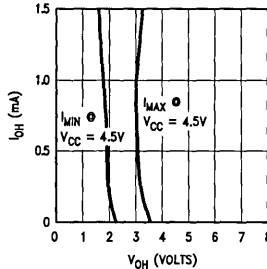
Source Current for Standard Output Configuration



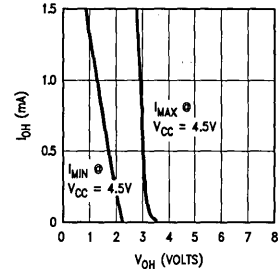
Source Current for SO and SK in Push-Pull Configuration



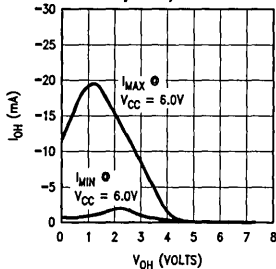
Source Current for L0-L7 in TRI-STATE Configuration (High Current Option)



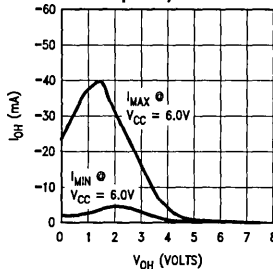
Source Current for L0-L7 in TRI-STATE Configuration (Low Current Option)



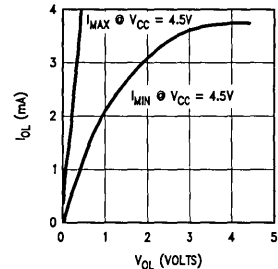
LED Output Source Current (for Low Current LED Option)



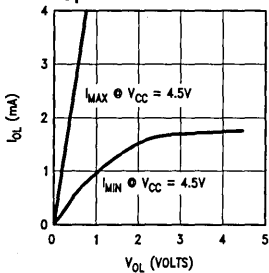
LED Output Source Current (for High Current LED Option)



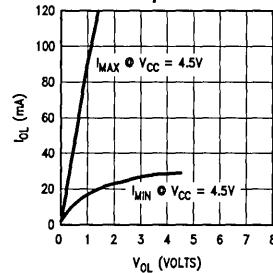
Output Sink Current for SO and SK



Output Sink Current for L0-L7 and Standard Drive Option for D0-D3 and G0-G3



Output Sink Current for D0-D3 with Very High Current Option



Output Sink Current for D0-D3 (for High Current Option)

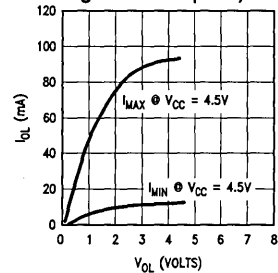


FIGURE 8b. COP310L/COP311L Input/Output Characteristics

TL/DD/6919-20

COP410L/411L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410L/411L instruction set.

TABLE II. COP410L/411L Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	2-bit Operand Field, 0–3 binary (RAM Register Select)
Br	Upper 2 bits of B (register address)	a	9-bit Operand Field, 0–511 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0–15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register		
G	4-bit Register to latch data for G I/O Port		
L	8-bit TRI-STATE I/O Port		
M	4-bit contents of RAM Memory pointed to by B Register		
PC	9-bit ROM Address Register (program counter)		
Q	8-bit Register to latch data for L I/O Port		
SA	9-bit Subroutine Save Register A		
SB	9-bit Subroutine Save Register B		
SIO	4-bit Shift Register and Counter		
SK	Logic-Controlled Clock Output		
		OPERATIONAL SYMBOLS	
		+	Plus
		–	Minus
		→	Replaces
		↔	Is exchanged with
		=	Is equal to
		\bar{A}	The one's complement of A
		⊕	Exclusive-OR
		:	Range of values

TABLE III. COP410L/411L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5–	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP410L/411L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	$\boxed{1111 \mid 1111}$	ROM (PC _{8,A,M}) → PC _{7:0}	None	Jump Indirect (Note 2)
JMP	a	6-- --	$\boxed{0110 \mid 000 \mid a_8}$ $\boxed{a_{7:0}}$	a → PC	None	Jump
JP	a	--	$\boxed{1 \mid a_{6:0}}$ (pages 2,3 only)	a → PC _{6:0}	None	Jump within Page (Note 3)
			or $\boxed{11 \mid a_{5:0}}$ (all other pages)	a → PC _{5:0}		
JSRP	a	--	$\boxed{10 \mid a_{5:0}}$	PC + 1 → SA → SB 010 → PC _{8:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 4)
JSR	a	6-- --	$\boxed{0110 \mid 100 \mid a_8}$ $\boxed{a_{7:0}}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	$\boxed{0100 \mid 1000}$	SB → SA → PC	None	Return from Subroutine
RETSK		49	$\boxed{0100 \mid 1001}$	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	$\boxed{0011 \mid 0011}$	A → Q _{7:4}	None	Copy A, RAM to Q
		3C	$\boxed{0011 \mid 1100}$	RAM(B) → Q _{3:0}		
LD	r	-5	$\boxed{00 \mid r \mid 0101}$	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	$\boxed{1011 \mid 1111}$	ROM(PC _{8,A,M}) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	$\boxed{0100 \mid 1100}$	0 → RAM(B) ₀	None	Reset RAM Bit
	1	45	$\boxed{0100 \mid 0101}$	0 → RAM(B) ₁		
	2	42	$\boxed{0100 \mid 0010}$	0 → RAM(B) ₂		
	3	43	$\boxed{0100 \mid 0011}$	0 → RAM(B) ₃		
SMB	0	4D	$\boxed{0100 \mid 1101}$	1 → RAM(B) ₀	None	Set RAM Bit
	1	47	$\boxed{0100 \mid 0111}$	1 → RAM(B) ₁		
	2	46	$\boxed{0100 \mid 0110}$	1 → RAM(B) ₂		
	3	4B	$\boxed{0100 \mid 1011}$	1 → RAM(B) ₃		
STII	y	7-	$\boxed{0111 \mid y}$	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	$\boxed{00 \mid r \mid 0110}$	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	$\boxed{0010 \mid 0011}$ $\boxed{1011 \mid 1111}$	RAM(3,15) ↔ A	None	Exchange A with RAM (3,15)
XDS	r	-7	$\boxed{00 \mid r \mid 0111}$	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	$\boxed{00 \mid r \mid 0100}$	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

Instruction Set (Continued)

TABLE III. COP410L/411L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	--	00 r (d-1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte	G ₀ = 0	Skip if G Bit is Zero
	0	01	0000 0001	} 2nd byte	G ₁ = 0	
	1	11	0001 0001		G ₂ = 0	
	2	03	0000 0011		G ₃ = 0	
	3	13	0001 0011			
SKMBZ		0 1 2 3	0000 0001 0001 0001 0000 0011 0001 0011		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 0011 0010 1110	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP410L/411L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC₈, A, M. PC₈ is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC₈, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → PC_{7,4}, RAM(B) → PC_{3,0}, leaving PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

- The first word of a COP410L/411L program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.
- The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or 7 will access data in the next group of 4 pages.

Option List

The COP410L/411L mask-programmable options are assigned numbers which correspond with the COP410L pins.

The following is a list of COP410L options. The LED Direct Drive option on the L Lines cannot be used if higher V_{CC} option is selected. When specifying a COP411L chip, Option 2 must be set to 3, Options 20, 21, and 22 to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Output (no option available for COP411L)

- = 0: Clock output to ceramic resonator
- = 1: Pin is RAM power supply (V_R) input
- = 3: No connection

Option 3: CKI Input

- = 0: Oscillator input divided by 8 (500 kHz max)
- = 1: Single-pin RC controlled oscillator divided by 4
- = 2: External Schmitt trigger level clock divided by 4

Option 4: RESET Input

- = 0: Load device to V_{CC}
- = 1: Hi-Z input

Option 5: L₇ Driver

- = 0: Standard output
- = 1: Open-drain output
- = 2: High current LED direct segment drive output
- = 3: High current TRI-STATE push-pull output
- = 4: Low-current LED direct segment drive output
- = 5: Low-current TRI-STATE push-pull output

Option 6: L₈ Driver

same as Option 5

Option 7: L₅ Driver

same as Option 5

Option 8: L₄ Driver

same as Option 5

Option 9: Operating voltage

- | | | |
|------|----------------|----------------|
| | COP41XL | COP31XL |
| = 0: | +4.5V to +6.3V | +4.5V to +5.5V |

Option 10: L₃ Driver

same as Option 5

Option 11: L₂ Driver

same as Option 5

Option 12: L₁ Driver

same as Option 5

Option 13: L₀ Driver

same as Option 5

Option 14: SI Input

- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 15: SO Driver

- = 0: Standard Output
- = 1: Open-drain output
- = 2: Push-pull output

Option 16: SK Driver

same as Option 15

Option List (Continued)

- Option 17: G₀ I/O Port
 = 0: Standard output
 = 1: Open-drain output
- Option 18: G₁ I/O Port
 same as Option 17
- Option 19: G₂ I/O Port
 same as Option 17
- Option 20: G₃ I/O Port (no option available for COP411L)
 same as Option 17
- Option 21: D₃ Output (no option available for COP411L)
 = 0: Very-high sink current standard output
 = 1: Very-high sink current open-drain output
 = 2: High sink current standard output
 = 3: High sink current open-drain output
 = 4: Standard LSTTL output (fanout = 1)
 = 5: Open-drain LSTTL output (fanout = 1)
- Option 22: D₂ Output (no option available for COP411L)
 same as Option 21
- Option 23: D₁ Output
 same as Option 21
- Option 24: D₀ Output
 same as Option 21

- Option 25: L Input Levels
 = 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
 = 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)
- Option 26: G Input Levels
 same as Option 25
- Option 27: SI Input Levels
 same as Option 25
- Option 28: COP Bonding
 = 0: COP410L (24-pin device)
 = 1: COP411L (20-pin device)
 = 2: Both 24- and 20-pin versions

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP410L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

- a. RAM and Internal Logic Test Mode (SI = 1)
- b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

Option Table

The following option information is to be sent to National along with the EPROM.

	Option Data	
OPTION 1	VALUE = <u>0</u>	IS: GROUND PIN
OPTION 2	VALUE = _____	IS: CKO PIN
OPTION 3	VALUE = _____	IS: CKI INPUT
OPTION 4	VALUE = _____	IS: RESET INPUT
OPTION 5	VALUE = _____	IS: L(7) DRIVER
OPTION 6	VALUE = _____	IS: L(6) DRIVER
OPTION 7	VALUE = _____	IS: L(5) DRIVER
OPTION 8	VALUE = _____	IS: L(4) DRIVER
OPTION 9	VALUE = <u>0</u>	IS: V _{CC} PIN
OPTION 10	VALUE = _____	IS: L(3) DRIVER
OPTION 11	VALUE = _____	IS: L(2) DRIVER
OPTION 12	VALUE = _____	IS: L(1) DRIVER
OPTION 13	VALUE = _____	IS: L(0) DRIVER
OPTION 14	VALUE = _____	IS: SI INPUT

	Option Data	
OPTION 15	VALUE = _____	IS: SO DRIVER
OPTION 16	VALUE = _____	IS: SK DRIVER
OPTION 17	VALUE = _____	IS: G ₀ I/O PORT
OPTION 18	VALUE = _____	IS: G ₁ I/O PORT
OPTION 19	VALUE = _____	IS: G ₂ I/O PORT
OPTION 20	VALUE = _____	IS: G ₃ I/O PORT
OPTION 21	VALUE = _____	IS: D ₃ OUTPUT
OPTION 22	VALUE = _____	IS: D ₂ OUTPUT
OPTION 23	VALUE = _____	IS: D ₁ OUTPUT
OPTION 24	VALUE = _____	IS: D ₀ OUTPUT
OPTION 25	VALUE = _____	IS: L INPUT LEVELS
OPTION 26	VALUE = _____	IS: G INPUT LEVELS
OPTION 27	VALUE = _____	IS: SI INPUT LEVELS
OPTION 28	VALUE = _____	IS: COPS BONDING

COP413L/COP313L Single Chip Microcontrollers

General Description

The COP413L and COP313L Single-Chip N-Channel Microcontrollers are members of the COPSM family, fabricated using N-channel, silicon gate MOS technology. These Control Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, 15 I/O lines with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Control Oriented Processor at a very low end-product cost.

The COP313L is an exact functional equivalent but extended temperature version of the COP413L.

The COP401L-R13 and COP410L-X13 should be used for exact emulation.

Features

- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-Level subroutine stack
- 16 μ s instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max.)
- Internal binary counter register with MICROWIRESM serial I/O capability
- General purpose outputs
- High noise immunity inputs ($V_{IL} = 1.2V$, $V_{IH} = 3.6V$)
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device COP313L ($-40^{\circ}C$ to $+85^{\circ}C$)

Block Diagram

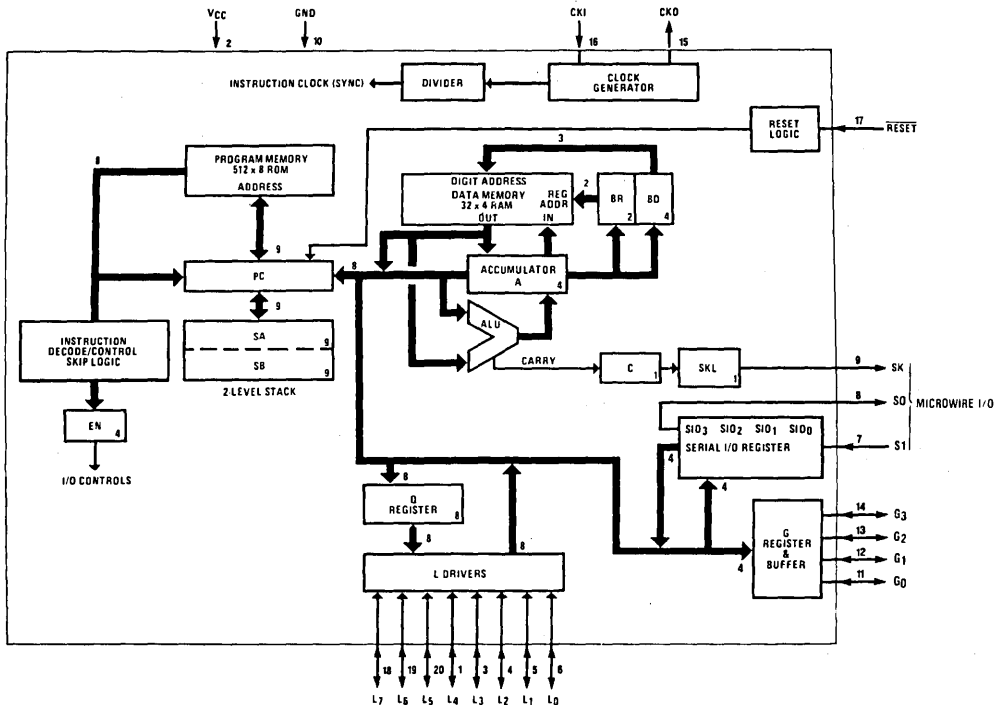


FIGURE 1

TL/DD/8371-1

COP413L Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.3 to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C

Power Dissipation COP413L

0.3 Watt at 70°C

Total Source Current 25 mA

Total Sink Current 25 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	6.3	V
Power Supply Ripple (Notes 1, 3)	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷ 8)				
Logic High (V _{IH})		3.0		V
Logic Low (V _{IL})			0.4	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V _{CC}		V
Logic Low			0.6	V
SO Input Level (Test Mode)	(Note 2)	2.5		V
SI Input Level				
Logic High	(TTL Level)	2.0		V
Logic Low			0.8	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance (Note 3)			7	pF
Reset Input Leakage		-1	+1	μA
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	V _{OL} = 0.4V	0.9		mA
L0-L7 Outputs, G0-G3	V _{OL} = 0.4V	0.4		mA
CKO (I _{OL})	V _{OL} = 0.4V	0.2		mA
Output Source Current				
L0-L7 and G0-G3	V _{OH} = 2.4V	-25		μA
SO and SK Outputs (I _{OH})	V _{OH} = 1.0V	-1.2		mA
Push-Pull	V _{OH} = 2.4V	-25		μA
SI Input Load Source Current	V _{IL} = 0V	-10	-140	μA
Total Sink Current Allowed				
L7-L4, G Port			4	mA
L3-L0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

Note 3: This parameter is only sampled and not 100% tested. Variation due to the device included.

COP313L Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.3 to +7V
Ambient Operating Temperature	-40°C to +85°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C

Power Dissipation COP313L	0.20 Watt at 85°C
Total Source Current	25 mA
Total Sink Current	25 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	5.5	V
Power Supply Ripple (Notes 1, 3)	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Logic High (V _{IH})		3.0		V
Logic Low (V _{IL})			0.3	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V _{CC}		V
Logic Low			0.4	V
SO Input (Test Mode)	(Note 2)	2.5		V
SI Input Level				
Logic High	(TTL Level)	2.2		V
Logic Low			0.6	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance (Note 3)			7	pF
Reset Input Leakage		-2	+2	μA
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	V _{OL} = 0.4V	0.8		mA
L0-L7 Outputs, G0-G3 (I _{OL})	V _{OL} = 0.4V	0.4		mA
CKO (I _{OL})	V _{OL} = 0.4V	0.2		mA
Output Source Current				
L0-L7 and G0-G3	V _{OH} = 2.4V	-23		μA
SO and SK Outputs (I _{OH})	V _{OH} = 1.0V	-1.0		mA
(Push-Pull)	V _{OH} = 2.4V	-23		μA
SI Input Load Source Current	V _{IL} = 0V	-10	-200	μA
Total Sink Current Allowed				
L7-L4, G Port			4	mA
L3-L0			4	mA
Any Other Pin			1.5	mA
Total Source Current Allowed				
Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

Note 3: This parameter is only sampled and not 100% tested. Variation due to the device included.

AC Electrical Characteristics

COP413L: $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$
 COP313L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time - t_c		16	40	μs
CKI				
Input Frequency - f_i	$\div 8$ Mode	0.2	0.5	MHz
Duty Cycle		30	60	%
Rise Time (Note 2)	$f_i = 0.5$ MHz		500	ns
Fall Time (Note 2)			200	ns
CKI Using RC ($\div 4$)	$R = 56\text{ k}\Omega \pm 5\%$ $C = 100\text{ pF} \pm 10\%$			
Instruction Cycle Time (Note 1)		16	28	μs
Inputs:				
G3-G0, L7-L0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: $C_L = 50\text{ pF}$, $R_L = 20\text{ k}\Omega$, $V_{\text{OUT}} = 1.5\text{V}$			
SO, SK Outputs			4.0	μs
t_{pd1} , t_{pd0}				
All Other Outputs			5.6	μs
t_{pd1} , t_{pd0}				

Note 1: Variation due to the device included.

Note 2: This parameter is only sampled and not 100% tested.

Connection Diagram

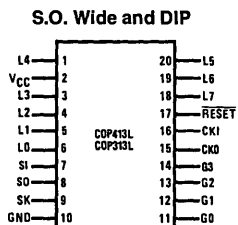


FIGURE 2

TL/DD/8371-2

Pin Descriptions

Pin	Description
L7-L0	8-bit bidirectional I/O port
G3-G0	4-bit bidirectional I/O port
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output or NC
RESET	System reset input
V _{CC}	Power Supply
GND	Ground

Order Number COP313L-XXX/D or COP413L-XXX/D
 See NS Hermetic Package Number D20A
 (D Pkg. for prototype only)

Order Number COP313L-XXX/WM or COP413L-XXX/WM
 See NS Surface Mount Package Number M20B

Order Number COP313L-XXX/N or COP413L-XXX/N
 See NS Molded Package Number N20A

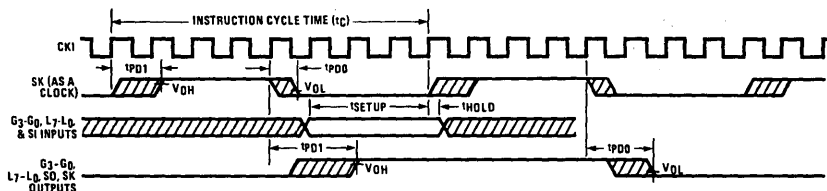


FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)

TL/DD/8371-3

Functional Description

A block diagram of the COP413L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP413L also apply to the COP313L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP413L instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction.

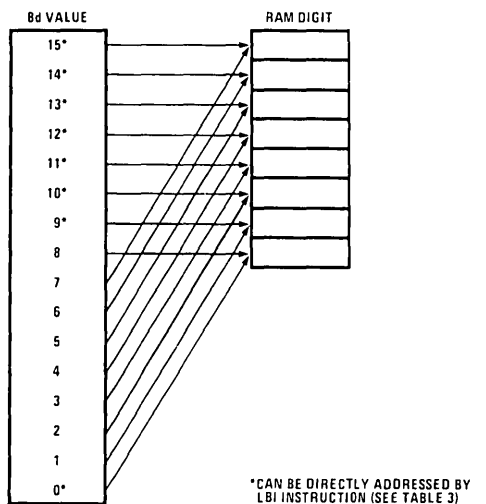
The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP413L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general purpose bidirectional I/O ports.



TL/DD/8371-4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃-EN₀).

1. The least significant bit of the enable register, EN₀ selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting with each instruction cycle time. The data present at SO goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.

2. EN₁ is not used. It has no effect on COP413L operation.

Functional Description (Continued)

TABLE I. Enable Register Modes - Bits EN₃ and EN₀

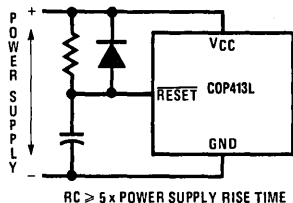
EN ₃	EN ₀	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high impedance input state.

4. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN₃ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN₃ and EN₀.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below (Figure 5). The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.

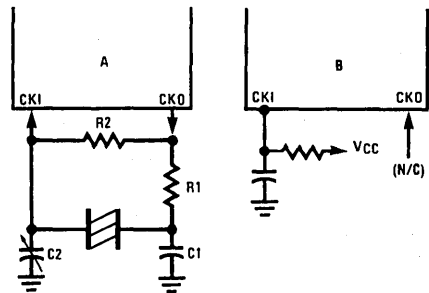

FIGURE 5. Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

OSCILLATOR

There are two basic clock oscillator configurations available as shown by Figure 6.

- Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.
- RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO becomes no connection.


FIGURE 6. COP413L Oscillator

Ceramic Resonator Oscillator

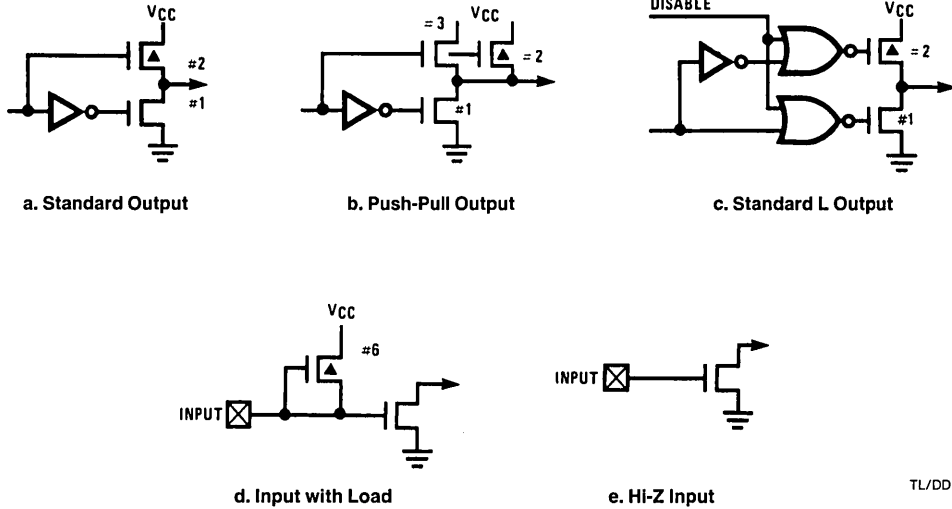
Resonator Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (k Ω)	C (pF)	Instruction Cycle Time (in μ s)
51	100	19 \pm 15%
82	56	19 \pm 13%

Note: 200 k Ω \geq R \geq 25 k Ω
220 pF \geq C \geq 50 pF

Functional Description (Continued)



TL/DD/8371-7

FIGURE 7. Input and Output Configurations

I/O CONFIGURATIONS

COP413L inputs and outputs have the following configurations, illustrated in *Figure 7*:

- a. G0–G3—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC} .
- b. SO, SK—an enhancement mode device to ground in conjunction with a depletion-mode device paralleled by an

enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

- c. L0–L7—same as a., but may be disabled.
- d. SI has on-chip depletion load device to V_{CC} .
- e. $\overline{\text{RESET}}$ has a Hi-Z input which must be driven to a “1” or “0” by external components.

Typical Performance Characteristics

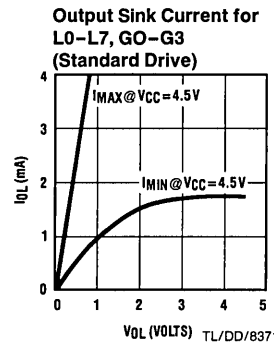
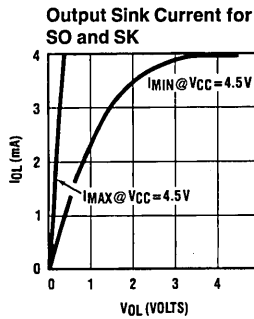
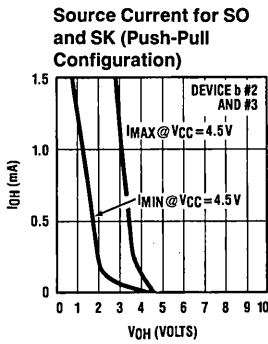
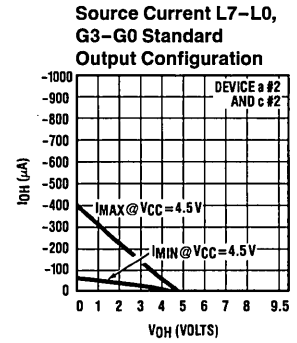
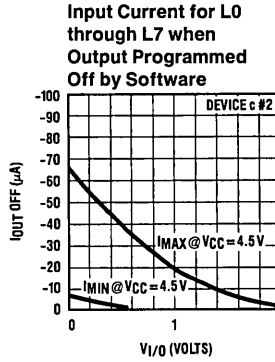
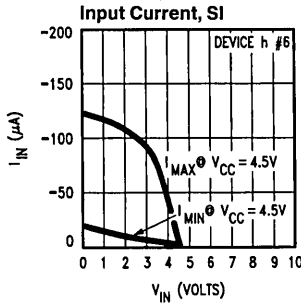


FIGURE 8a. COP413L I/O DC Current Characteristics

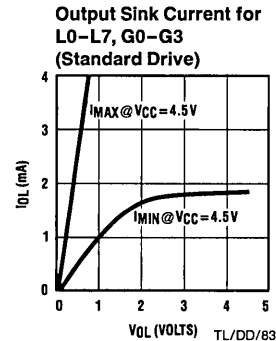
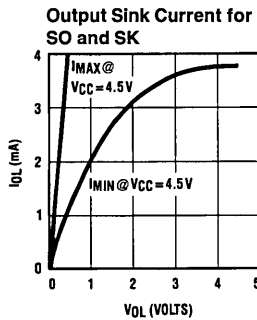
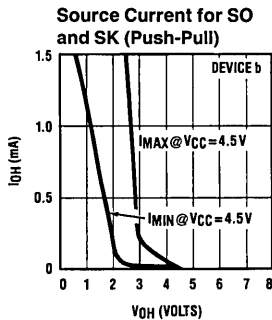
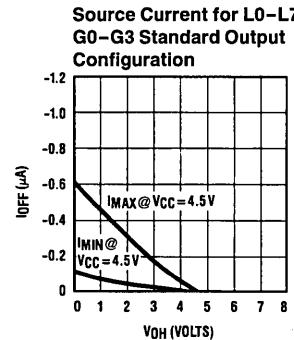
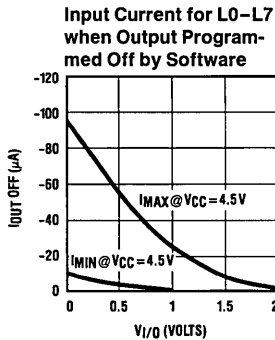
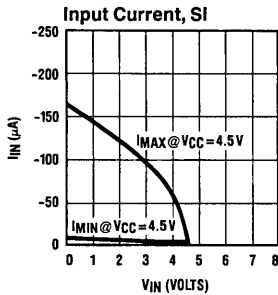


FIGURE 8b. COP313L I/O DC Current Characteristics

COP413L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table. Table III provides the mnemonic, oper-

and, machine code data flow, skip conditions and description associated with each instruction in the COP413L instruction set.

TABLE II. COP413L Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE® I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic Controlled Clock Output
Instruction Operand Symbols	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
Operational Symbols	
+	Plus
–	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
\bar{A}	The one's complement of A
⊕	Exclusive-OR
:	Range of values

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5-	0101 y	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	0000 0000	$0 \rightarrow A$	None	Clear A
COMP		40	0100 0000	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" $\rightarrow C$	None	Reset C
SC		22	0010 0010	"1" $\rightarrow C$	None	Set C
XOR		02	0000 0010	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	$\text{ROM}(\text{PC}_8, \text{A}, \text{M}) \rightarrow \text{PC}_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6-	0110 000 a ₈ a _{7:0}	$a \rightarrow \text{PC}$	None	Jump
JP	a	-	1 a _{6:0} (pages 2, 3 only) or 11 a _{5:0} (all other pages)	$a \rightarrow \text{PC}_{6:0}$ $a \rightarrow \text{PC}_{5:0}$	None	Jump within-Page (Note 3)
JSRP	a	-	10 a _{5:0}	$\text{PC} + 1 \rightarrow \text{SA} \rightarrow \text{SB}$ $010 \rightarrow \text{PC}_{8:6}$ $a \rightarrow \text{PC}_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6-	0110 100 a ₈ a _{7:0}	$\text{PC} + 1 \rightarrow \text{SA} \rightarrow \text{SB}$ $a \rightarrow \text{PC}$	None	Jump to Subroutine
RET		48	0100 1000	$\text{SB} \rightarrow \text{SA} \rightarrow \text{PC}$	None	Return from Subroutine
RETSK		49	0100 1001	$\text{SB} \rightarrow \text{SA} \rightarrow \text{PC}$	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	$A \rightarrow \text{Q}_{7:4}$	None	Copy A, RAM to Q
LD	r	-5	0011 1100 00 r 0101	$\text{RAM}(B) \rightarrow \text{Q}_{3:0}$ $\text{RAM}(B) \rightarrow A$ $\text{Br} \oplus r \rightarrow \text{Br}$	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011 1111	$\text{ROM}(\text{PC}_8, \text{A}, \text{M}) \rightarrow \text{Q}$ $\text{SA} \rightarrow \text{SB}$	None	Load Q Indirect (Note 2)
RMB	0	4C	0100 1100	$0 \rightarrow \text{RAM}(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	$0 \rightarrow \text{RAM}(B)_1$		
	2	42	0100 0010	$0 \rightarrow \text{RAM}(B)_2$		
	3	43	0100 0011	$0 \rightarrow \text{RAM}(B)_3$		
SMB	0	4D	0100 1101	$1 \rightarrow \text{RAM}(B)_0$	None	Set RAM Bit
	1	47	0100 0111	$1 \rightarrow \text{RAM}(B)_1$		
	2	46	0100 0110	$1 \rightarrow \text{RAM}(B)_2$		
	3	4B	0100 1011	$1 \rightarrow \text{RAM}(B)_3$		

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110	RAM(B) ↔ A Br @ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	0010 0011 1011 1111	RAM(3,15) ↔ A	None	Exchange A with RAM (3,15)
XDS	r	-7	00 r 0111	RAM(B) ↔ A Bd - 1 → Bd Br @ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	-4	00 r 0100	RAM(B) ↔ A Bd + 1 → Bd Br @ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	-	00 r (d-1) (d=0,9:15)	r,d → B	Skip until not a LBI	Load B immediate with r,d (Note 5)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		21	0010 0001	1st byte	}	Skip if G Bit is Zero
		33	0011 0011			
	0	01	0000 0001	G ₀ = 0		
	1	11	0001 0001	G ₁ = 0		
	2	03	0000 0011	G ₂ = 0		
	3	13	0001 0011	G ₃ = 0		
SKMBZ	0	01	0000 0001		RAM(B) ₀ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001		RAM(B) ₁ = 0	
	2	03	0000 0011		RAM(B) ₂ = 0	
	3	13	0001 0011		RAM(B) ₃ = 0	

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set (Continued)

Mnemonic	Operand	Machine		Data Flow	Skip Conditions	Description
		Hex Code	Language Code (Binary)			
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
		2E	0010 1110			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined) Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1* e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP413L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC₈, A, M. PC₈ is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC₈, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → PC_{7:4}, RAM (B)

→ PC_{3:0}, leaving PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

- The first word of a COP413L program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.
- The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or will access data in the next group of 4 pages.

Description of Selected Instructions (Continued)

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmable COP413L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

- a. RAM and internal Logic Test Mode (SI = 1)
- b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

Option List

The option selected must be sent in with the EPROM of ROM Code for a Mask order of 413L. Make xerox copy of the table, select the appropriate option, and send it in with the EPROM.

COP 413L/COP 313L

Option 1: Oscillator Selection

= 0 Ceramic Resonator or external input frequency divided by 8. CKO is oscillator output.

= 1 Single pin RC controlled oscillator divided by 4. CKO is no connection.

NOTE:

The following option information is to be sent to National along with the EPROM

Option 1: Value = _____ is: Oscillator Selection



COP413C/COP413CH/COP313C/COP313CH

Single-Chip CMOS Microcontrollers

General Description

The COP413C, COP413CH, COP313C, and COP313CH fully static, single-chip CMOS microcontrollers are members of the COPSTM family, fabricated using double-poly, silicongate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP413CH is identical to the COP413C except for operating voltage and frequency. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide a customized controller-oriented processor at a low end-product cost.

The COP313C/COP313CH is the extended temperature range version of the COP413C/COP413CH.

For emulation use the ROMless COP404C.

Features

- Lowest power dissipation (40 μ W typical)
- Low cost
- Power-saving HALT Mode
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-level subroutine stack
- DC to 4 μ s instruction time
- Single supply operation (3V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter with MICROWIRE™ compatible serial I/O
- Software/hardware compatible with other members of the COP400 family
- Extended temperature (-40°C to +85°C) devices available

Block Diagram

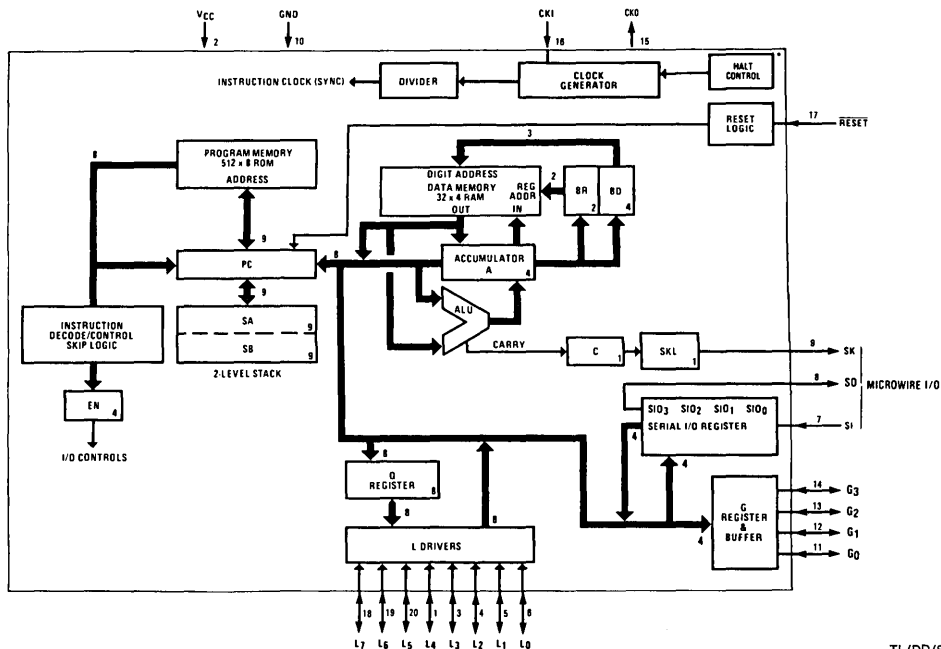


FIGURE 1. COP413C/413CH

TL/DD/8537-1

COP413C/COP413CH

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA

Operating Temperature Range 0°C to +70°C
Storage Temperature Range -65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Conditions	COP413C		COP413CH		Units
		Min	Max	Min	Max	
Operating Voltage		3.0	5.5	4.5	5.5	V
Power Supply Ripple (Notes 4, 5)			0.1 V _{CC}		0.1 V _{CC}	V
Supply Current (Note 1)	V _{CC} = 5.0V, t _c = Min V _{CC} = 3.0V, t _c = Min (t _c is inst. cycle)		500 300		2000	μA μA
HALT Mode Current (Note 2)	V _{CC} = 5.0V, F _I = 0 kHz V _{CC} = 3.0V, F _I = 0 kHz		30 10		30	μA μA
Input Voltage Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		0.9 V _{CC} 0.7 V _{CC}	0.1 V _{CC} 0.2 V _{CC}	0.9 V _{CC} 0.7 V _{CC}	0.1 V _{CC} 0.2 V _{CC}	V V V V
RESET, SI Input Leakage		-1	+1	-1	+1	μA
Input Capacitance (Notes 5, 6)			7		7	pF
Output Voltage Levels (SO, SK, L Port) Logic High Logic Low	I _{OH} = -10 μA I _{OL} = 10 μA	V _{CC} - 0.2	0.2	V _{CC} - 0.2	0.2	V V
Output Current Levels Sink (Note 3) Source (SO, SK, L Port) Source (G Port)	V _{CC} = Min, V _{OUT} = V _{CC} V _{CC} = Min, V _{OUT} = 0V V _{CC} = Min, V _{OUT} = 0V	0.2 -0.1 -8	-150	1.2 -0.5 -30	-330	mA mA μA
Allowable Sink/Source Current Per Pin (Note 3)			5		5	mA
TRI-STATE Leakage Current		-2	+2	-2	+2	μA

COP413C/COP413CH

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	COP413C		COP413CH		Units
		Min	Max	Min	Max	
Instruction Cycle Time		16	DC	4	DC	μs
Operating CKI Frequency	$\div 8$ Mode	DC	500	DC	2000	kHz
Instruction Cycle Time RC Oscillator $\div 4$	R = $30\text{k} \pm 5\%$, $V_{\text{CC}} = 5\text{V}$ C = $82\text{ pF} \pm 5\%$			8	16	μs
Instruction Cycle Time RC Oscillator $\div 4$ (Note 6)	R = $56\text{k} \pm 5\%$, $V_{\text{CC}} = 5\text{V}$ C = $100\text{ pF} \pm 5\%$	16	32	16	32	μs
Duty Cycle (Note 5)	$F_i = \text{Max freq ext clk}$	40	60	40	60	%
Rise Time (Note 5)	$F_i = \text{Max freq ext clk}$		60		60	ns
Fall Time (Note 5)	$F_i = \text{Max freq ext clk}$		40		40	ns
Inputs (See Figure 3) t_{SETUP}	G Inputs SI Input L Inputs	$t_c/4 + 2.8$ 1.2 6.8		$t_c/4 + 0.7$ 0.3 1.7		μs μs μs
t_{HOLD}		1.0		0.25		μs
Output Propagation Delay t_{PD1} , t_{PD0}	$V_{\text{OUT}} = 1.5$, $C_L = 100\text{ pF}$ $R_L = 5\text{k}$		4.0		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled to V_{CC} with 5k resistors.

Note 2: The Halt mode will stop CKI from oscillating.

Note 3: SO output sink current must be limited to keep V_{OL} less than $0.2 V_{\text{CC}}$ when part is running in order to prevent entering test mode.

Note 4: Voltage change must be less than 0.5V in a 1 ms period.

Note 5: This parameter is only sampled and not 100% tested.

Note 6: Variation due to the device included.

COP313C/COP313CH**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA

Total Allowable Sink Current	25 mA
Operating Temperature Range	-40°C to +85°C
Storage Temperature Range	-65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	COP313C		COP313CH		Units
		Min	Max	Min	Max	
Operating Voltage		3.0	5.5	4.5	5.5	V
Power Supply Ripple (Notes 4, 5)			0.1 V _{CC}		0.1 V _{CC}	V
Supply Current (Note 1)	V _{CC} = 5.0V, t _c = Min V _{CC} = 3.0V, t _c = Min (t _c is inst. cycle)		600 360		2500	μA μA
Halt Mode Current (Note 2)	V _{CC} = 5.0V, Fi = 0 kHz V _{CC} = 3.0V, Fi = 0 kHz		50 20		50	μA μA
Input Voltage Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		0.9 V _{CC} 0.7 V _{CC}	0.1 V _{CC}	0.9 V _{CC} 0.7 V _{CC}	0.1 V _{CC} 0.2 V _{CC}	V V V V
RESET, SI Input Leakage		-2	+2	-2	+2	μA
Input Capacitance (Notes 5, 6)			7		7	pF
Output Voltage Levels (SO, SK, L Port) Logic High Logic Low	I _{OH} = -10 μA I _{OL} = 10 μA	V _{CC} - 0.2	0.2	V _{CC} - 0.2	0.2	V V
Output Current Levels Sink (Note 3) Source (SO, SK, L Port) Source (G Port)	V _{CC} = Min, V _{OUT} = V _{CC} V _{CC} = Min, V _{OUT} = 0V V _{CC} = Min, V _{OUT} = 0V	0.2 -0.1 -8	-200	1.2 -0.5 -30	-440	mA mA μA
Allowable Sink/Source Current Per Pin (Note 3)			5		5	mA
TRI-STATE Leakage Current (Note 3)		-4	+4	-4	+4	μA

COP313C/COP313CH

AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	COP313C		COP313CH		Units
		Min	Max	Min	Max	
Instruction Cycle Time		16	DC	4	DC	μs
Operating CKI Frequency	$\div 8$ Mode	DC	500	DC	2000	kHz
Instruction Cycle Time RC Oscillator $\div 4$	R = $30\text{k} \pm 5\%$, $V_{CC} = 5\text{V}$ C = $82\text{pF} \pm 5\%$			8	16	μs
Instruction Cycle Time RC Oscillator $\div 4$ (Note 6)	R = $56\text{k} \pm 5\%$, $V_{CC} = 5\text{V}$ C = $100\text{pF} \pm 5\%$	16	32	16	32	μs
Duty Cycle (Note 5)	Fi = Max Freq Ext Clk	40	60	40	60	%
Rise Time (Note 5)	Fi = Max Freq Ext Clk		60		60	ns
Fall Time (Note 5)	Fi = Max Freq Ext Clk		40		40	ns
Inputs (See Figure 3)						
t_{SETUP}	G Inputs	$tc/4 + 2.8$		$tc/4 + 0.7$		μs
	SI Input	1.2		0.3		μs
	L Inputs	6.8		1.7		μs
t_{HOLD}		1.0		0.25		μs
Output Propagation Delay	$V_{\text{OUT}} = 1.5\text{V}$, $C_L = 100\text{pF}$ $R_L = 5\text{k}$					
t_{PD1} , t_{PD0}			4.0		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating.

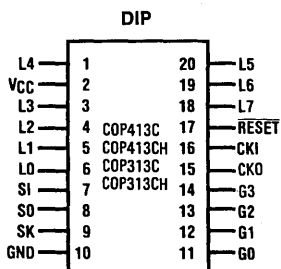
Note 3: SO output sink current must be limited to keep V_{OL} less than $0.2 V_{CC}$ when part is running in order to prevent entering test mode.

Note 4: Voltage change must be less than 0.5V in a 1 ms period.

Note 5: This parameter is only sampled and not 100% tested.

Note 6: Variation due to the device included.

Connection Diagram



Top View

TL/DD/8537-2

Pin Descriptions

Pin	Description
L7-L0	8-bit bidirectional I/O port with TRI-STATE
G3-G0	4-bit bidirectional I/O port
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	Crystal oscillator output, or NC
RESET	System reset input
V_{CC}	System power supply
GND	System Ground

FIGURE 2

Order Number COP313C-XXX/D, COP313CH-XXX/D,
COP413C-XXX/D or COP413CH-XXX/D
See NS Hermetic Package Number D20A

Order Number COP313C-XXX/N, COP313CH-XXX/N,
COP413C-XXX/N or COP413CH-XXX/N
See NS Molded Package Number N20A

Order Number COP313C-XXX/WM or
COP413C-XXX/WM
See NS Small Outline Package Number M20B

Timing Waveform

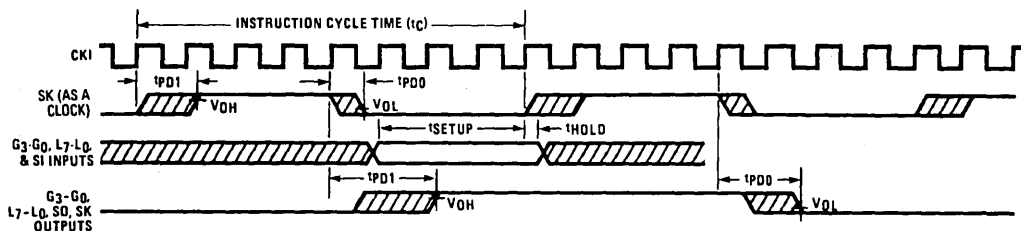


FIGURE 3. Input/Output Timing Diagrams (Divide-by-8 Mode)

TL/DD/8537-3

Development Support

The MOLE (Microcontroller On Line Emulator) is a low cost development system and real time emulator for COPS' products. They also include TMP, 8050 and the new 16 bit HPC microcontroller family. The MOLE provides effective support for the development of both software and hardware in the user's application.

The purpose of the MOLE is to provide a tool to write and assemble code, emulate code for the target microcontroller and assist in debugging of the system.

The MOLE can be connected to various hosts, IBM PC, STARPLEX™, Kaypro, Apple and Intel systems, via RS-232 port. This link facilitates the up loading/down loading of code, supports host assembly and mass storage.

The MOLE consists of three parts; brain, personality and optional host software.

The brain board is the computing engine of the system. It is a self-contained computer with its own firmware which provides for all system operation, emulation control, communication, from programming and diagnostic operation. It has three serial ports which can be connected to a terminal, host system, printer, modem or to other MOLE's in a multi-MOLE environment.

The personality board contains the necessary hardware and firmware needed to emulate the target microcontroller. The emulation cable which replaces the target controller attaches to this board. The software contains a cross assembler and a communications program for up loading and down loading code from the MOLE.

MOLE Ordering Information

P/N	Description
MOLE-BRAIN	MOLE Computer Board
MOLE-COPS-PB1	COPS Personality Board
MOLE-XXX-YYY	Optional Software

Where XXX = COPS

YYY = Host System, IBM, Apple,
KAY (Kaypro), CP/M

Functional Description

To ease reading of this description, only COP413C is referenced; however, all such references apply equally to COP413CH, COP313C, and COP313CH.

A block diagram of the COP413C is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP413C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8×4 -bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4*. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP413C is designed to ensure fully static operation of the device.

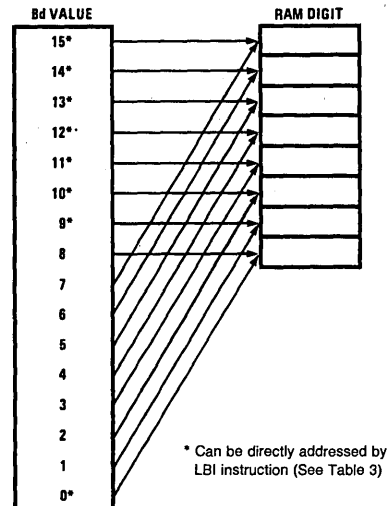
The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP413C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.



TL/DD/8537-4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP413C is MICROWIRE compatible.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3-EN0).

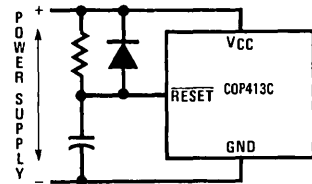
1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.
2. EN 1 is not used, it has no effect on the COP413C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift

register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

INITIALIZATION

The external RC network shown in *Figure 5* must be connected to the RESET pin. The RESET pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the RESET input, providing it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



$$RC > 5 \times \text{Power Supply Rise Time} \\ \text{and } RC > 100 \times \text{CKI Period}$$

TL/DD/8537-8

FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

HALT MODE

The COP413C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may be halted by the HALT instruction. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the RESET function.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw more than a square-wave input. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP413C current drain.

$$I_c = I_q + (V \times 20 \times F_i) + (V \times 1280 \times F_i / D_v)$$

where I_c = chip current drain in microamps

I_q = quiescent leakage current (from curve)

F_i = CKI frequency in megahertz

V = chip V_{CC} in volts

D_v = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 8),

$$I_c = 30 + (5 \times 20 \times 0.4) + (5 \times 1280 \times 0.4/8)$$

$$I_c = 30 + 40 + 320 = 390 \mu A$$

OSCILLATOR OPTIONS

There are two options available that define the use of CKI and CKO.

- Crystal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 8.
- RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle time equals the oscillation frequency divided by 4. CKO is NC.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.

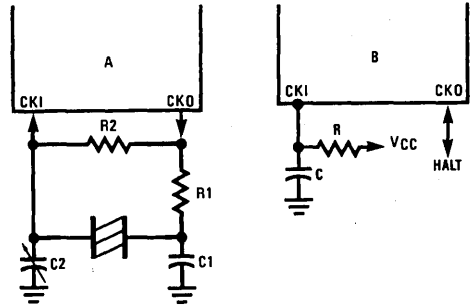


FIGURE 6. COP413C Oscillator

TL/DD/8537-6

Crystal or Resonator

RC-Controlled Oscillator

Crystal Value	Component Value				Cycle Time				V_{CC}
	R1	R2	C1 pF	C2 pF	R	C	Time		
32 kHz	220k	20M	30	5-36	15k	82 pF	4-9 μs	$\geq 4.5V$ COP413CH Only	
455 kHz	5k	10M	80	40	30k	82 pF	8-16 μs	$\geq 4.5V$ COP413CH Only	
2.000 MHz	2k	1M	30	6-36	47k	100 pF	16-32 μs	3.0 to 4.5V COP413C Only	
					56k	100 pF	16-32 μs		$\geq 4.5V$
Note: $15k \leq R \leq 150k$, $50 pF \leq C \leq 150 pF$									

Functional Description (Continued)

I/O CONFIGURATIONS

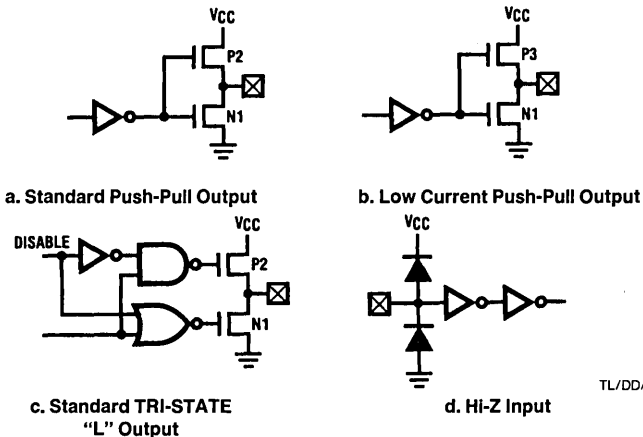
COP413C outputs have the following configurations, illustrated in Figure 7:

- a. Standard SO, SK Output. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.
- b. Low Current G Output. This is the same configuration as (a) above except that the sourcing current is much less.
- c. Standard TRI-STATE L Output. L output is a CMOS output buffer similar to (a) which may be disabled by program control.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (Figure 7d).

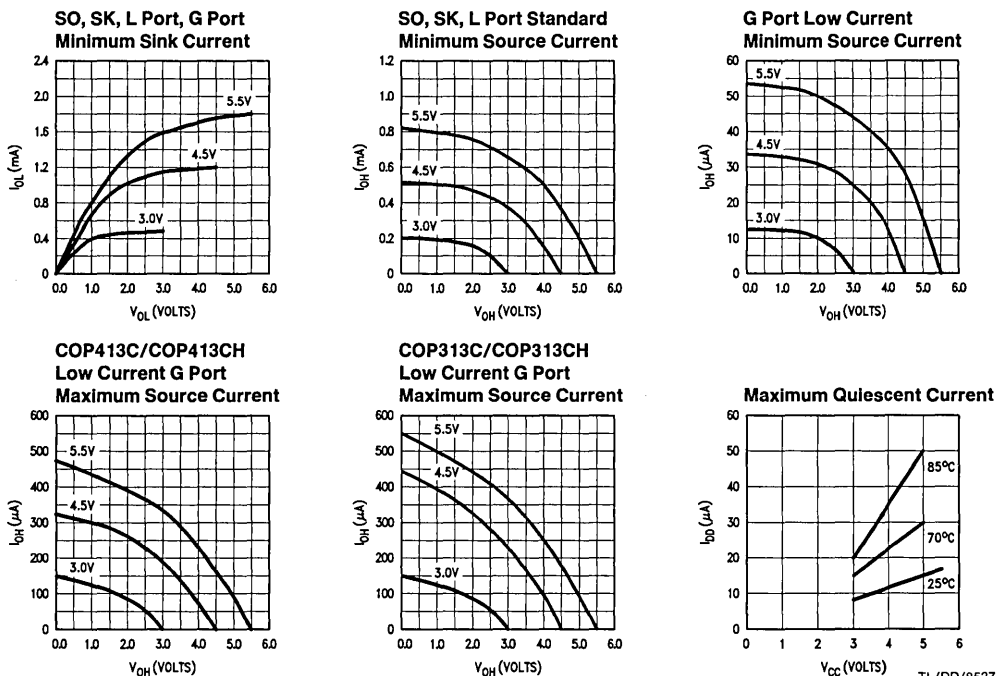
When using the G I/O port as an input, set the output register to a logic "1" level. The P-channel device will act as a pull-up load. When using the L I/O port as an input, disable the L drivers with the LEI instruction. The drivers are then in TRI-STATE mode and can be driven externally.

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in Figure 8 for each of these devices to allow the designer to effectively use these I/O configurations.



TL/DD/8537-7

FIGURE 7. I/O Configurations



TL/DD/8537-8

FIGURE 8

COP413C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP413C instruction set.

TABLE II. COP413C Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	2-bit Operand Field, 0–3 binary (RAM Register Select)
Br	Upper 2 bits of B (register address)	a	9-bit Operand Field, 0–511 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0–15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
EN	4-bit Enable Register	ROM(t)	Contents of ROM location addressed by t
G	4-bit Register to latch data for G I/O Port	OPERATIONAL SYMBOLS	
L	8-bit TRI-STATE I/O Port	+	Plus
M	4-bit contents of RAM Memory pointed to by B Register	–	Minus
PC	9-bit ROM Address Register (program counter)	→	Replaces
Q	8-bit Register to latch data for L I/O Port	↔	Is exchanged with
SA	9-bit Subroutine Save Register A	=	Is equal to
SB	9-bit Subroutine Save Register B	\bar{A}	The one's complement of A
SIO	4-bit Shift Register and Counter	⊕	Exclusive-OR
SK	Logic-Controlled Clock Output	:	Range of values

TABLE III. COP413C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5–	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add immediate, Skip on Carry ($y \neq 0$)
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

COP413C Instruction Set (Continued)

TABLE III. COP413C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC ₈ , A, M) → PC _{7:0}	None	Jump Indirect (Note 2)
JMP	a	6-	0110 000 a ₈ - a _{7:0}	a → PC	None	Jump
JP	a	-	1 a _{6:0} (pages 2, 3 only)	a → PC _{6:0}	None	Jump within Page (Note 1)
			11 a _{5:0} (all other pages)	a → PC _{5:0}		
JSRP	a	-	10 a _{5:0}	PC + 1 → SA → SB 010 → PC _{8:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 2)
JSR	a	6-	0110 100 a ₈ - a _{7:0}	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 10011	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33	0011 0011		None	Halt processor
		38	0011 1000			
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	A → Q _{7:4}	None	Copy A, RAM to Q
		3C	0011 1100	RAM(B) → Q _{3:0}		
CQMA		33	0011 0011	Q _{7:4} → RAM(B)	None	Copy Q to RAM, A
		2C	0010 1100	Q _{3:0} → A		
LD	r	-5	00 r 0101	RAM(B) → A Br @ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	1011 1111	ROM(PC ₈ , A, M) → Q SA → SB	None	Load Q Indirect
RMB	0	4C	0100 1100	0 → RAM(B) ₀	None	Reset RAM Bit
		45	0100 0101	0 → RAM(B) ₁		
		42	0100 0010	0 → RAM(B) ₂		
		43	0100 0011	0 → RAM(B) ₃		
SMB	0	4D	0100 1101	1 → RAM(B) ₀	None	Set RAM Bit
		47	0100 0111	1 → RAM(B) ₁		
		46	0100 0110	1 → RAM(B) ₂		
		4B	0100 1011	1 → RAM(B) ₃		
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110	RAM(B) ↔ A Br @ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23	0010 0011		None	Exchange A with RAM (3,15)
		BF	1011 1111			

COP413C Instruction Set (Continued)

TABLE III. COP413C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	00 r 0111	RAM(B) \leftrightarrow A Bd - 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	-4	00 r 0100	RAM(B) \leftrightarrow A Bd + 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A \rightarrow Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd \rightarrow A	None	Copy Bd to A
LBI	r,d	-	00 r (d - 1) (d = 0,9:15)	r,d \rightarrow B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6-	0011 0011 0010 y	y \rightarrow EN	None	Load EN Immediate
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
		21	0010 0001			
SKGBZ		33	0011 0011	1st byte	G ₀ = 0 G ₁ = 0 G ₂ = 0 G ₃ = 0	Skip if G Bit is Zero
	0	01	0000 0001			
	1	11	0001 0001			
	2	03	0000 0011			
SKMBZ		3	0010 0011	2nd byte	RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
	0	01	0000 0001			
	1	11	0001 0001			
	2	03	0000 0011			
	3	13	0001 0011			
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G \rightarrow A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	L _{7:4} \rightarrow RAM(B) L _{3:0} \rightarrow A	None	Input L Ports to RAM, A
		2E	0010 1110			
OMG		33	0011 0011	RAM(B) \rightarrow G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A \leftrightarrow SIO, C \rightarrow SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP413C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC₈, A, M. PC₈ is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC₈, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → PC_{7:4}, RAM(B) → PC_{3:0}, leaving PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

INSTRUCTION SET NOTES

- The first word of a COP413C program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).
- The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

COPS Programming Manual

For detailed information on writing COPS programs, the COPS Programming Manual 424410284-001 provides an in-depth discussion of the COPS architecture, instruction set and general techniques of COPS programming. This manual is written with the programmer in mind.

OPTION LIST—OSCILLATOR SELECTION

The oscillator option selected must be sent in with the EPROM of ROM Code for masking into the COP413C. Select the appropriate option, make a photocopy of the table and send it with the EPROM.

COP413C/COP313C

Option 1: Oscillator selection

- = 0 Ceramic Resonator input frequency divided by 8. CKO is oscillator output.
- = 1 Single pin RC controlled oscillator divided by 4. CKO is no connection.

Note: The following option information is to be sent to National along with the EPROM.

Option 1: Value = ____ is Oscillator Selected.



PRELIMINARY

COP414L/COP314L Single-Chip N-Channel Microcontrollers

General Description

The COP414L Single-Chip N-Channel Microcontrollers are members of the COPSTM family, fabricated using N-channel, silicon gate MOS technology. This Controller Oriented Processor is a complete microcomputer containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP414L is an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

The COP314L is an exact functional equivalent but extended temperature version of COP414L.

The COP414L can be emulated by the COP404C. The COP401L should be used for exact emulation.

Features

- Late waterfab programming of ROM and I/O for fast delivery of units
- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-level subroutine stack
- 16 μ s instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device
 - COP314L (–40°C to +85°C)
- Wider supply range (4.5V–6.3V) optionally available

Block Diagram

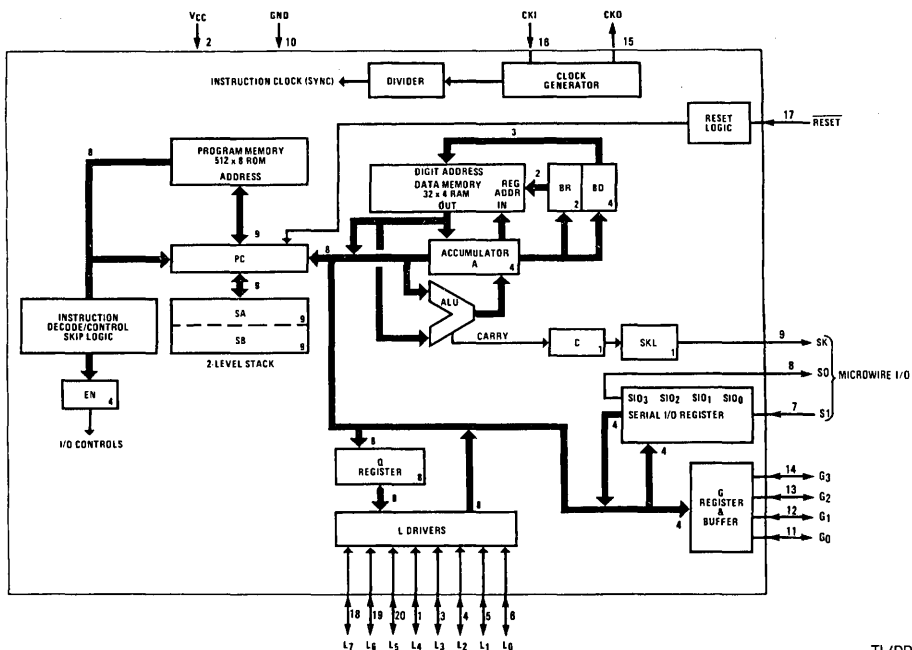


FIGURE 1. COP414L

TL/DD/8814-1

COP414L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Power Dissipation COP414L	0.65W at 25°C 0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	6.3	V
Power Supply Ripple (Note 1)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷ 8)				
Logic High (V _{IH})	V _{CC} = Max	3.0		V
Logic High (V _{IH})	V _{CC} = 5V ± 5%	2.0		V
Logic Low (V _{IL})		-0.3	0.4	V
Schmitt Trigger Input (÷ 4)				
Logic High (V _{IH})		0.7 V _{CC}		V
Logic Low (V _{IL})		-0.3	0.6	V
RESET Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	2.5	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V _{CC} = 5V ± 5%	-0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		-1	+1	μA
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ± 10%			
Logic High (V _{OH})	I _{OH} = -25 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 mA		0.4	V
CMOS Operation				
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.



COP414L**DC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 6.3\text{V}$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OL}} = 0.4\text{V}$	1.2		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.9		mA
L ₀ -L ₇ Outputs, G ₀ -G ₃ and LSTTL D ₀ -D ₃ Outputs (I_{OL})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.4		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.4		mA
CKI (Single-pin RC Oscillator)	$V_{\text{CC}} = 4.5, V_{\text{IH}} = 3.5\text{V}$	2		mA
CKO	$V_{\text{CC}} = 4.5, V_{\text{OL}} = 0.4\text{V}$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OH}} = 2.0\text{V}$	-75	-480	μA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 2.0\text{V}$	-30	-250	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OH}} = 2.4\text{V}$	-1.4		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 1.0\text{V}$	-1.2		mA
Input Load Source Current	$V_{\text{CC}} = 5.0\text{V}, V_{\text{IL}} = 0\text{V}$	-10	-140	μA
Open Drain Output Leakage		-2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L ₇ -L ₄ , G Port			4	mA
L ₃ -L ₀			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L ₇ -L ₄			60	mA
L ₃ -L ₀			60	mA
Each L Pin			25	mA
Any Other Pin			1.5	mA

COP314L**Absolute Maximum Ratings**

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	-40°C to +85°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Power Dissipation COP314L	0.65W at 25°C 0.20W at 85°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

COP314L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})		4.5	5.5	V
Power Supply Ripple (Note 1)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input ($\div 8$)				
Crystal Input				
Logic High (V_{IH})	$V_{CC} = \text{Max}$	3.0		V
Logic High (V_{IH})	$V_{CC} = 5\text{V} \pm 5\%$	2.2		V
Logic Low (V_{IL})		-0.3	0.3	V
Schmitt Trigger Input ($\div 4$)				
Logic High (V_{IH})		$0.7 V_{CC}$		V
Logic Low (V_{IL})		-0.3	0.4	V
RESET Input Levels	(Schmitt Trigger Input)			
Logic High		$0.7 V_{CC}$		V
Logic Low		-0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.2	2.5	V
All Other Inputs				
Logic High	$V_{CC} = \text{Max}$	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, $V_{CC} = 5\text{V} \pm 5\%$	-0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		-2	+2	μA
Output Voltage Levels				
LSTTL Operation				
Logic High (V_{OH})	$V_{CC} = 5\text{V} \pm 10\%$	2.7		V
Logic Low (V_{OL})	$I_{OH} = -20 \mu\text{A}$ $I_{OL} = 0.36 \text{ mA}$		0.4	V
CMOS Operation				
Logic High	$I_{OH} = -10 \mu\text{A}$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10 \mu\text{A}$		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

COP314L**DC Electrical Characteristics** (Continued)COP314L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{\text{CC}} = 5.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	1.0		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.8		mA
L_0 - L_7 Outputs, G_0 - G_3 and LSTTL, D_0 - D_3 Outputs (I_{OL})	$V_{\text{CC}} = 5.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.4		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.4		mA
CKI (Single-pin RC Oscillator)	$V_{\text{CC}} = 4.5\text{V}, V_{\text{IH}} = 3.5\text{V}$	1.5		mA
CKO	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{\text{CC}} = 5.5\text{V}, V_{\text{OH}} = 2.0\text{V}$	-55	-600	μA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 2.0\text{V}$	-28	-350	μA
Push-Pull Configuration	$V_{\text{CC}} = 5.5\text{V}, V_{\text{OH}} = 2.0\text{V}$	-1.1		mA
SO and SK Outputs (I_{OH})	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 1.0\text{V}$	-1.2		mA
Input Load Source Current	$V_{\text{CC}} = 5.0\text{V}, V_{\text{IL}} = 0\text{V}$	-10	-200	μA
Open Drain Output Leakage		-5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7 - L_4 , G Port			4	mA
L_3 - L_0			4	mA
Any Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7 - L_4			60	mA
L_3 - L_0			60	mA
Each L Pin			25	mA
Any Other Pins			1.5	mA

AC Electrical Characteristics

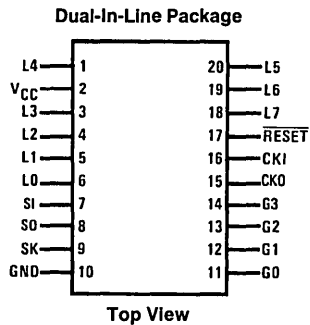
COP414L: $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 6.3\text{V}$ unless otherwise noted

COP314L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time — t_{C}		16	40	μs
CKI				
Input Frequency — f_{I}	$\div 8$ Mode	0.2	0.5	MHz
	$\div 4$ Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	$f_{\text{I}} = 0.5$ MHz		500	ns
Fall Time			200	ns
CKI Using RC ($\div 4$)	$R = 56\text{ k}\Omega \pm 5\%$ $C = 100\text{ pF} \pm 10\%$			
Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
Inputs				
G_3-G_0, L_7-L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: $C_L = 50\text{ pF}, R_L = 20\text{ k}\Omega, V_{\text{OUT}} = 1.5\text{V}$			
SO, SK Outputs			4.0	μs
$t_{\text{pd1}}, t_{\text{pd0}}$				
All Other Outputs			5.6	μs
$t_{\text{pd1}}, t_{\text{pd0}}$				

Note 1: Variation due to the device included.

Connection Diagram



TL/DD/8814-2

Order Number COP314L-XXX/D
or COP414L-XXX/D
See NS Hermetic Package D20A
(Prototyping Package Only)

Order Number COP314L-XXX/WM
or COP414L-XXX/WM
See NS Surface Mount Package M20B

Order Number COP314L-XXX/N
or COP414L-XXX/N
See NS Molded Package N20A

FIGURE 2

Pin Descriptions

Pin	Description
L_7-L_0	8 bidirectional I/O ports with TRI-STATE
G_3-G_0	4 bidirectional I/O ports
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)

Pin	Description
CKI	System oscillator input
CKO	System oscillator output
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams

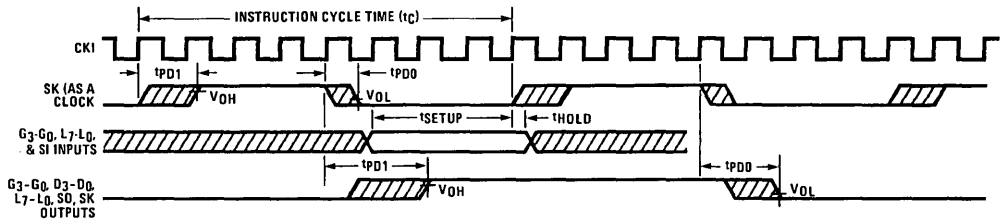


FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)

TL/DD/8814-3

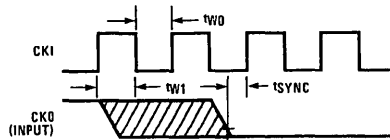


FIGURE 3a. Synchronization Timing

TL/DD/8814-4

Functional Description

A block diagram of the COP414L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP414L also apply to the COP314L, and COP214L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP414L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it

may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).

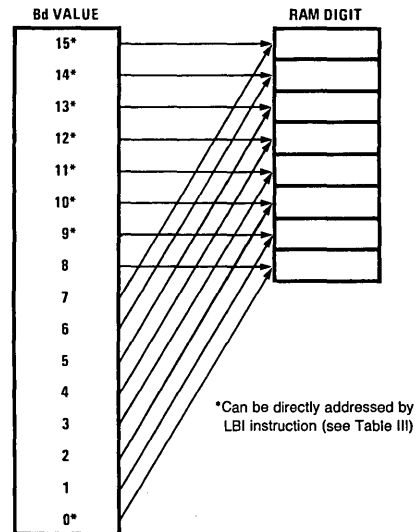


FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

TL/DD/8814-5

Functional Description (Continued)

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP414L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃-EN₀).

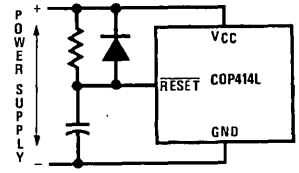
1. The least significant bit of the enable register, EN₀, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, *decrementing* its value by one upon

each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.

2. EN₁ is not used. It has no effect on COP414L operation.
3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN₃ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN₃ and EN₀.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below (Figure 5). The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.



RC ≥ 5 × Power Supply Rise Time TL/DD/8814-6

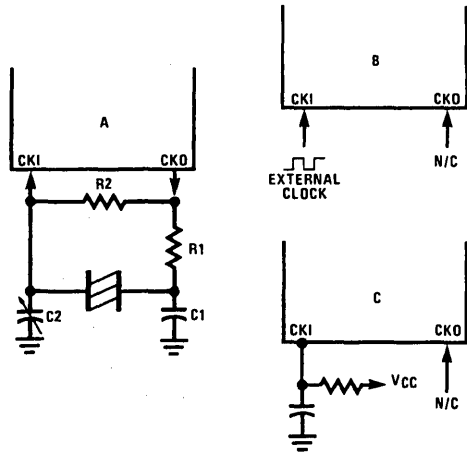
FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN₃ and EN₀

EN ₃	EN ₀	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/8814-7

Ceramic Resonator Oscillator

Resonator Value	Components Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (k Ω)	C (pF)	Instruction Cycle Time in μ s
51	100	19 \pm 15%
82	56	19 \pm 13%

Note: 200 k Ω \geq R \geq 25 k Ω . 360 pF \geq C \geq 50 pF. Does not include tolerances.

FIGURE 6. COP414L Oscillator

OSCILLATOR

There are four basic clock oscillator configurations available as shown by *Figure 6*.

- Resonator Controlled Oscillator.** CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.
- External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 4 to give the instruction frequency time. CKO is no connection.

- RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is no connection.

CKO PIN OPTIONS

In a resonator controlled oscillator system, CKO is used as an output to the resonator network. CKO is no connection for External or RC controlled oscillator.

I/O OPTIONS

COP414L inputs and outputs have the following optional configurations, illustrated in *Figure 7*:

- Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC} , compatible with LSTTL and CMOS input requirements. Available on SO, SK and all D and G outputs.
- Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK and all D and G outputs.
- Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.
- Standard L**—same as a., but may be disabled. Available on L outputs only.
- Open Drain L**—same as b., but may be disabled. Available on L outputs only.
- An on-chip depletion load device to V_{CC} .
- A Hi-Z input which must be driven to a "1" or "0" by external components.

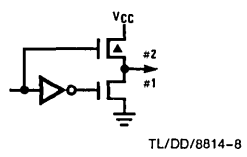
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1-6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP414L system.

The SO, SK outputs can be configured as shown in a., b., or c. The G outputs can be configured as shown in a. or b. Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in d., or e.

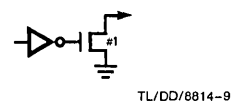
An important point to remember if using configuration d. with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 8*, device 2.) However, when the L port is used as input, the disabled depletion device CAN NOT be relied on to source sufficient current to pull an input to a logic "1".

Functional Description (Continued)

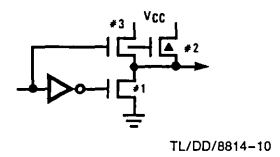
a. Standard Output



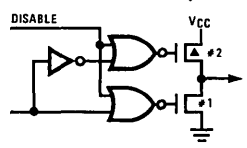
b. Open-Drain Output



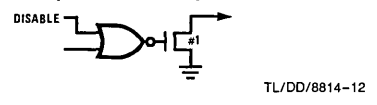
c. Push-Pull Output



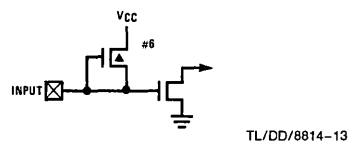
d. Standard L Output



e. Open-Drain L Output



f. Input with Load



g. Hi-Z Input

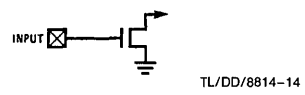
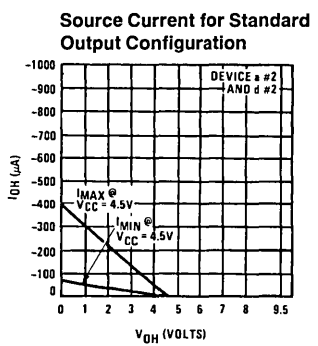
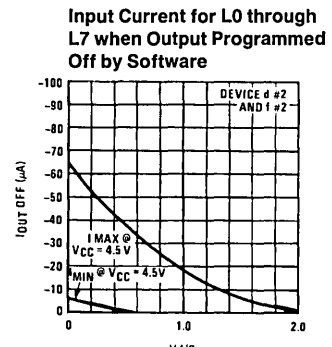
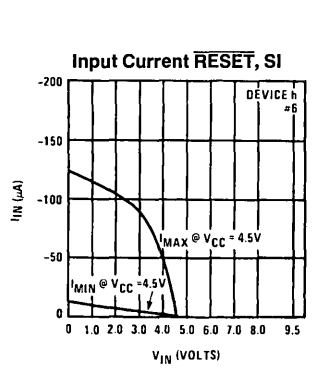
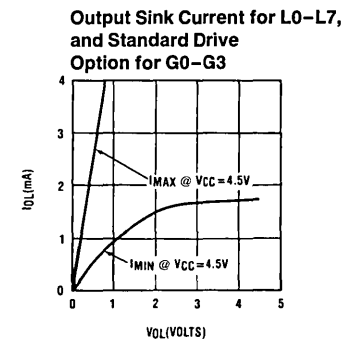
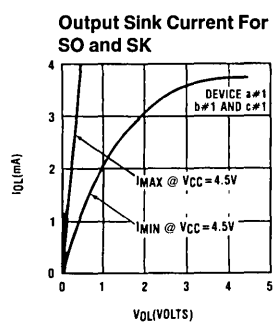
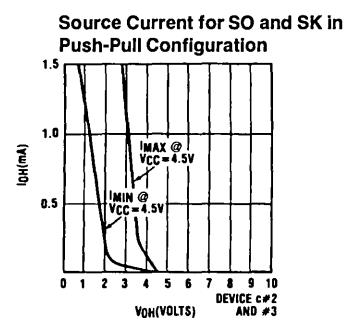


FIGURE 7. Input and Output Configurations

Typical Performance Curves



TL/DD/8814-15



TL/DD/8814-16

FIGURE 8a. COP414 I/O DC Current Characteristics

Typical Performance Curves (Continued)

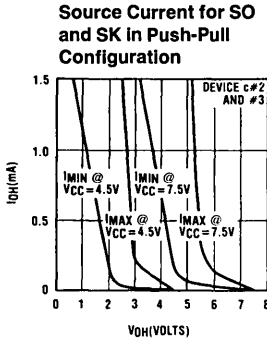
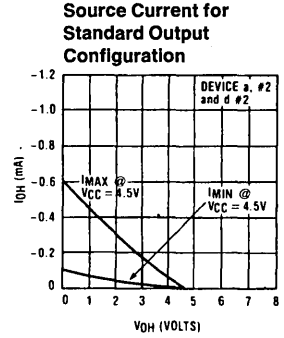
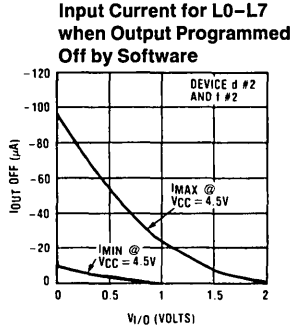
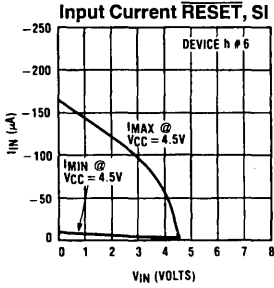


FIGURE 8b. COP314L Input/Output Characteristics

TL/DD/8814-17

COP414L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP414L instruction set.

TABLE II. COP414L Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	2-bit Operand Field, 0–3 binary (RAM Register Select)
Br	Upper 2 bits of B (register address)	a	9-bit Operand Field, 0–511 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0–15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register	OPERATIONAL SYMBOLS	
G	4-bit Register to latch data for G I/O Port	+	Plus
L	8-bit TRI-STATE I/O Port	–	Minus
M	4-bit contents of RAM Memory pointed to by B Register	→	Replaces
PC	9-bit ROM Address Register (program counter)	↔	Is exchanged with
Q	8-bit Register to latch data for L I/O Port	=	Is equal to
SA	9-bit Subroutine Save Register A	\bar{A}	The one's complement of A
SB	9-bit Subroutine Save Register B	⊕	Exclusive-OR
SIO	4-bit Shift Register and Counter	:	Range of values
SK	Logic-Controlled Clock Output		

TABLE III. COP414L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	[0011 0000]	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	[0011 0001]	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5–	[0101 y]	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	[0000 0000]	$0 \rightarrow A$	None	Clear A
COMP		40	[0100 0000]	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	[0100 0100]	None	None	No Operation
RC		32	[0011 0010]	"0" $\rightarrow C$	None	Reset C
SC		22	[0010 0010]	"1" $\rightarrow C$	None	Set C
XOR		02	[0000 0010]	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

COP414L Instruction Set (Continued)

TABLE III. COP414L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	$\boxed{1111 1111}$	ROM (PC ₈ ,A,M) PC _{7:0}	None	Jump Indirect (Note 2)
JMP	a	6--	$\boxed{0110 000 a_8 a_{7:0}}$	a → PC	None	Jump
JP	a	--	$\boxed{1 a_{6:0}}$ (pages 2, 3 only) or	a → PC _{6:0}	None	Jump within Page (Note 3)
			$\boxed{11 a_{5:0}}$ (all other pages)	a → PC _{5:0}		
JSRP	a	--	$\boxed{10 a_{5:0}}$	PC + 1 → SA → SB 010 → PC _{8:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 4)
JSR	a	6--	$\boxed{0110 100 a_8 a_{7:0}}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	$\boxed{0100 1000}$	SB → SA → PC	None	Return from Subroutine
RETSK		49	$\boxed{0100 1001}$	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	$\boxed{0011 0011}$ $\boxed{0011 1100}$	A → Q _{7:4} RAM(B) → Q _{3:0}	None	Copy A, RAM to Q
LD	r	-5	$\boxed{00 r 0101}$	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	$\boxed{1011 1111}$	ROM(PC ₈ , A, M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	$\boxed{0100 1100}$	0 → RAM(B) ₀	None	Reset RAM Bit
	1	45	$\boxed{0100 0101}$	0 → RAM(B) ₁		
	2	42	$\boxed{0100 0010}$	0 → RAM(B) ₂		
	3	43	$\boxed{0100 0011}$	0 → RAM(B) ₃		
SMB	0	4D	$\boxed{0100 1101}$	1 → RAM(B) ₀	None	Set RAM Bit
	1	47	$\boxed{0100 0111}$	1 → RAM(B) ₁		
	2	46	$\boxed{0100 0110}$	1 → RAM(B) ₂		
	3	4B	$\boxed{0100 1011}$	1 → RAM(B) ₃		
STII	y	7-	$\boxed{0111 y }$	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	$\boxed{00 r 0110}$	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3, 15	23 BF	$\boxed{0010 0011}$ $\boxed{1011 1111}$	RAM(3,15) ↔ A	None	Exchange A with RAM (3,15)
XDS	r	-7	$\boxed{00 r 0111}$	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	$\boxed{00 r 0100}$	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r

COP414L Instruction Set (Continued)

TABLE III. COP414L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	[0101 0000]	A → Bd	None	Copy A to Bd
CBA		4E	[0100 1110]	Bd → A	None	Copy Bd to A
LBI	r,d	--	[00 r (d-1)] (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6-	[0011 0011] [0010 y]	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	[0010 0000]		C = "1"	Skip if C is True
SKE		21	[0010 0001]		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	[0011 0011] [0010 0001]		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	[0011 0011]	1st byte 2nd byte	G ₀ = 0	Skip if G Bit is Zero
	0	01	[0000 0001]		G ₁ = 0	
	1	11	[0001 0001]		G ₂ = 0	
	2	03	[0000 0011]		G ₃ = 0	
	3	13	[0001 0011]			
SKMBZ		01 11 2 03 3 13	[0000 0001] [0001 0001] [0000 0011] [0001 0011]		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	[0011 0011] [0010 1010]	G → A	None	Input G Ports to A
INL		33 2E	[0011 0011] [0010 1110]	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
OBD		33 3E	[0011 0011] [0011 1110]	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	[0011 0011] [0011 1010]	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	[0100 1111]	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Option List

The COP414L mask-programmable options are assigned numbers which correspond with the COP414L pins.

The following is a list of COP414L options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: L₄ Driver

- = 0: Standard output
- = 1: Open-drain output

Option 2: V_{CC} Pin

- = 0: Standard V_{CC}

Option 3: L₃ Driver

same as Option 1

Option 4: L₂ Driver

same as Option 1

Option 5: L₁ Driver

same as Option 1

Option 6: L₀ Driver

same as Option 1

Option 7: SI Input

- = 0: load device to V_{CC}
- = 1: Hi-Z Output

Option 8: SO Driver

- = 0: Standard output
- = 1: Open-drain output
- = 2: Push-pull output

Option 9: SK Driver

same as Option 8

Option 10:

- = 0: Ground Pin—no options available

Option 11: G₀ I/O Port

- = 0: Standard output
- = 1: Open-drain output

Option 12: G₁ I/O Port

same as Option 11

Option 13: G₂ I/O Port

same as Option 11

Option 14: G₃ I/O Port

same as Option 11

Option 15: CKO Output

- = 0: Clock output to ceramic resonator/crystal
- = 1: No connection

Option 16: CKI Input

- = 0: Oscillator input divided by 8 (500 kHz max)
- = 1: Single pin RC controlled oscillator divided by 4
- = 2: External Schmitt trigger level clock divided by 4

Option 17: RESET Input

- = 0: Load device to V_{CC}
- = 1: Hi-Z Input

Option 18: L₇ Driver

same as Option 1

Option 19: L₆ Driver

same as Option 1

Option 20: L₅ Driver

same as Option 1

Option 21: L Input Levels

- = 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
- = 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 22: G Input Levels

same as Option 21

Option 23: SI Input Levels

same as Option 21

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP414L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

- a. RAM and Internal Logic Test Mode (SI = 1)
- b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing tests only.

COP414L Option List

Please fill out the Option List and send it with the EPROM.

Option Data

- OPTION 1 VALUE = _____ IS: L₄ DRIVER
- OPTION 2 VALUE = _____ IS: V_{CC} PIN
- OPTION 3 VALUE = _____ IS: L₃ DRIVER
- OPTION 4 VALUE = _____ IS: L₂ DRIVER
- OPTION 5 VALUE = _____ IS: L₁ DRIVER
- OPTION 6 VALUE = _____ IS: L₀ DRIVER
- OPTION 7 VALUE = _____ IS: SI INPUT
- OPTION 8 VALUE = _____ IS: SO DRIVER
- OPTION 9 VALUE = _____ IS: SK DRIVER
- OPTION 10 VALUE = 0 IS: GROUND PIN
- OPTION 11 VALUE = 0 IS: G₀ I/O PORT
- OPTION 12 VALUE = _____ IS: G₁ I/O PORT
- OPTION 13 VALUE = _____ IS: G₂ I/O PORT
- OPTION 14 VALUE = _____ IS: G₃ I/O PORT
- OPTION 15 VALUE = _____ IS: CKO OUTPUT
- OPTION 16 VALUE = _____ IS: CKI INPUT
- OPTION 17 VALUE = _____ IS: RESET INPUT
- OPTION 18 VALUE = _____ IS: L₇ DRIVER
- OPTION 19 VALUE = _____ IS: L₆ DRIVER
- OPTION 20 VALUE = _____ IS: L₅ DRIVER
- OPTION 21 VALUE = _____ IS: L INPUT LEVELS
- OPTION 22 VALUE = _____ IS: G INPUT LEVELS
- OPTION 23 VALUE = _____ IS: SI INPUT LEVELS

COP420/COP421/COP422 and COP320/COP321/COP322 Single-Chip N-Channel Microcontrollers

General Description

The COP420, COP421, COP422, COP320, COP321 and COP322 Single-Chip N-Channel Microcontrollers are members of the COPSTM family, fabricated using N-channel, silicon gate MOS technology. They are complete microcomputers containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP421 is identical to the COP420, except with 19 I/O lines instead of 23; the COP422 has 15 I/O lines. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

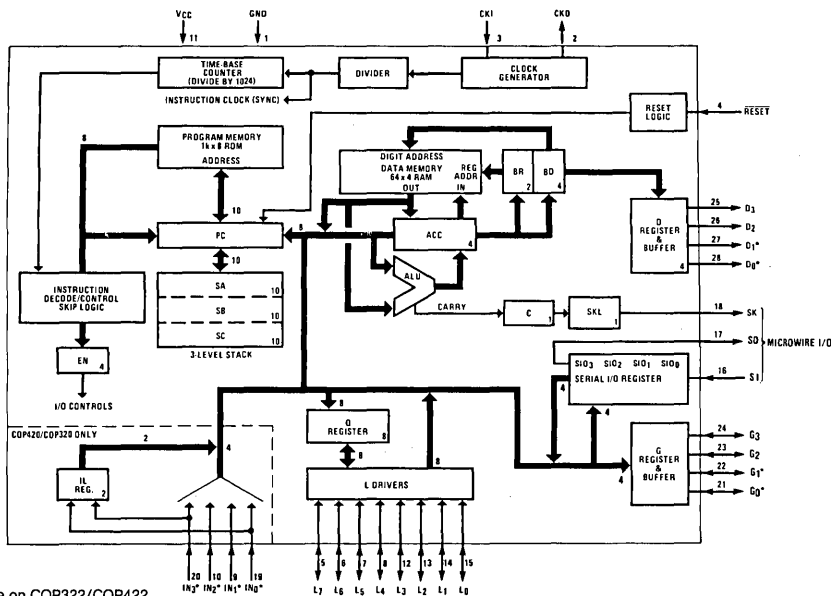
The COP320 is the extended temperature range version of the COP420 (likewise the COP321 and COP322 are the extended temperature range versions of the COP421/COP422). The COP320/321/322 are exact functional equivalents of the COP420/421/422.

Features

- Low cost
- Powerful instruction set
- 1k x 8 ROM, 64 x 4 RAM
- 23 I/O lines (COP420, COP320)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 4.0 μ s instruction time
- Single supply operation
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O capacity
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- LED direct drive outputs
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device COP320/COP321/COP322 (-40°C to +85°C)
- COP420P emulator available

COP420/COP421/COP422/COP320/COP321/COP322

Block Diagram



*Not available on COP322/COP422.

FIGURE 1

TL/DD/6921-1

COP420/COP421/COP422 and COP320/COP321/COP322**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin	-0.3V to +7V
Operating Temperature Range	
COP420/COP421/COP422	0°C to 70°C
COP320/COP321/COP322	-40°C to +85°C
Storage Temperature Range	-65°C to +150°C
Total Sink Current	75 mA
Total Source Current	95 mA

Package Power Dissipation	750 mW at 25°C
24 and 28 pin	400 mW at 70°C
250 mW at 85°C	
Package Power Dissipation	650 mW at 25°C
20 pin	300 mW at 70°C
200 mW at 85°C	

Lead Temperature (soldering, 10 sec.) 300°C

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP420/COP421/COP422**DC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	6.3	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	Outputs Open		38	mA
Supply Current	Outputs Open, $V_{CC} = 5\text{V}$, $T_A = 25^{\circ}\text{C}$		30	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High	$V_{CC} = \text{Max.}$	3.0		V
Logic High	$V_{CC} = 5\text{V} \pm 5\%$	2.0		V
Logic Low		-0.3	0.4	V
TTL Input	$V_{CC} = 5\text{V} \pm 5\%$			
Logic High		2.0		V
Logic Low		-0.3	0.8	V
Schmitt Trigger Inputs				
RESET, CKI (+4)				
Logic High		$0.7 V_{CC}$		V
Logic Low		-0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	3.0	V
All Other Inputs				
Logic High	$V_{CC} = \text{Max.}$	3.0		V
Logic High	$V_{CC} = 5\text{V} \pm 5\%$	2.0		V
Logic Low		-0.3	0.8	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		-0.3	1.2	V
Input Load Source Current	$V_{CC} = 5\text{V}$			
CKO		-4	-800	μA
All Others		-100	-800	μA
Input Capacitance (Note 3)			7	pF
Hi-Z Input Leakage		-1	+1	μA
Output Voltage Levels				
Standard Outputs				
TTL Operation	$V_{CC} = 5\text{V} \pm 10\%$			
Logic High	$I_{OH} = -100 \mu\text{A}$	2.4		V
Logic Low	$I_{OL} = 1.6 \text{ mA}$	-0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	$I_{OH} = -10 \mu\text{A}$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10 \mu\text{A}$		0.2	V

Note 1: TRI-STATE and LED configurations are excluded.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

Note 3: This parameter is only sampled and not 100% tested. Variation due to the device included.

COP420/COP421/COP422**DC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
LED Direct Drive Output	$V_{CC} = 6\text{V}$			
Logic High	$V_{OH} = 2.0\text{V}$	2.5	14	mA
CKI Sink Current (R/C Option)	$V_{IN} = 3.5\text{V}$	2		mA
CKO (RAM Supply Current)	$V_R = 3.3\text{V}$		3	mA
TRI-STATE or Open Drain Leakage Current	$V_{CC} = 5\text{V}$	-2.5	+2.5	μA
Output Current Levels				
Output Sink Current (I_{OL})	$V_{CC} = 4.5\text{V}, V_{OL} = 0.4\text{V}$	+1.6		mA
Output Source Current (I_{OH})				
Standard Configuration				
All Outputs	$V_{CC} = 6.3\text{V}, V_{OH} = 3.0\text{V}$	-200	-900	μA
	$V_{CC} = 4.5\text{V}, V_{OH} = 2.0\text{V}$	-100	-500	μA
Push-Pull Configuration				
SO, SK Outputs	$V_{CC} = 6.3\text{V}, V_{OH} = 3.0\text{V}$	-1.0		mA
	$V_{CC} = 4.5\text{V}, V_{OH} = 2.0\text{V}$	-0.4		mA
TRI-STATE Configuration				
L ₀ -L ₇ Outputs	$V_{CC} = 6.3\text{V}, V_{OH} = 3.2\text{V}$	-0.8		mA
	$V_{CC} = 4.5\text{V}, V_{OH} = 1.5\text{V}$	-0.9		mA
LED Configuration				
L ₀ -L ₇ Outputs	$V_{CC} = 6.3\text{V}, V_{OH} = 3.0\text{V}$	-1.0		mA
	$V_{CC} = 4.5\text{V}, V_{OH} = 2.0\text{V}$	-0.5		mA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			-15	mA
Per Pin (All Others)			-1.5	mA

COP320/COP321/COP322

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	5.5	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	$T_A = -40^{\circ}\text{C}$, Outputs Open		40	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input	$V_{CC} = \text{Max}$			
Logic High		2.2		V
Logic Low		-0.3	0.3	V
TTL Input	$V_{CC} = 5\text{V} \pm 5\%$			
Logic High		2.2		V
Logic Low		-0.3	0.6	V
Schmitt Trigger Inputs				
RESET, CKI ($\div 4$)				
Logic High		$0.7 V_{CC}$		V
Logic Low		-0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.0	3.0	V
All Other Inputs				
Logic High	$V_{CC} = \text{Max}$	3.0		V
Logic High	$V_{CC} = 5\text{V} \pm 5\%$	2.2		V
Logic Low		-0.3	0.6	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		-0.3	1.2	V
Input Load Source Current	$V_{CC} = 5\text{V}$			
CKO		-4	-800	μA
All Others		-100	-800	μA
Input Capacitance (Note 3)			7	pF
Hi-Z Input Leakage		-2	+2	μA
Output Voltage Levels				
Standard Outputs				
TTL Operation	$V_{CC} = 5\text{V} \pm 10\%$			
Logic High	$I_{OH} = -75 \mu\text{A}$	2.4		V
Logic Low	$I_{OL} = 1.6 \text{mA}$	-0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	$I_{OH} = -10 \mu\text{A}$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10 \mu\text{A}$	-0.3	0.2	V
Output Current Levels				
LED Direct Drive Output	$V_{CC} = 5\text{V}$ (Note 4)			
Logic High	$V_{OH} = 2.0\text{V}$	1.0	12	mA
CKI Sink Current (R/C Option)	$V_{IN} = 3.5\text{V}$	2		mA
CKO (RAM Supply Current)	$V_R = 3.3\text{V}$		4	mA
TRI-STATE or Open Drain Leakage Current		-5	+5	μA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			-15	mA
Per Pin (All Others)			-1.5	mA

Note 1: TRI-STATE and LED configurations are excluded.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

Note 3: This parameter is only sampled and not 100% tested. Variation due to the device included.

AC Electrical Characteristics

COP420/COP421/COP422 $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted

COP320/COP321/COP322 $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	$\div 16$ mode	1.6	4.0	MHz
	$\div 8$ mode	0.8	2.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time (Note 3)	Freq. = 4 MHz		60	ns
Fall Time (Note 3)	Freq. = 4 MHz		40	ns
CKI Using RC (Figure 8c)				
Frequency	$\div 4$ mode $R = 15\text{ k}\Omega \pm 5\%$, $C = 100\text{ pF}$	0.5	1.0	MHz
Instruction Cycle Time (Note 2)		4	8	μs
Inputs:				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
Output Propagation Delay	Test Conditions: $R_L = 5\text{ k}\Omega$, $C_L = 50\text{ pF}$, $V_{\text{OUT}} = 1.5\text{V}$	300		ns
SO and SK				
t_{pd1}			1.0	μs
t_{pd0}			1.0	μs
CKO				
t_{pd1}			0.25	μs
t_{pd0}			0.25	μs
All Other Outputs				
t_{pd1}			1.4	μs
t_{pd0}			1.4	μs

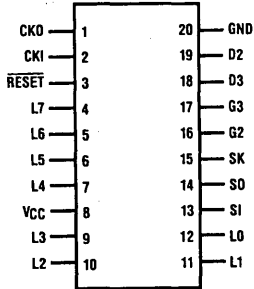
Note 1: Duty cycle = $t_{W1}/(t_{W1} + t_{W0})$.

Note 2: Variation due to the device included.

Note 3: This parameter is only sampled and not 100% tested.

Connection Diagrams

**COP422, COP322
DIP**

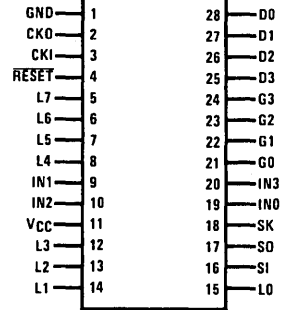


Top View

Order Number COP322-XXX/N
or COP422-XXX/N
See NS Molded Package N20A
Order Number COP322-XXX/D
or COP422-XXX/D
See NS Hermetic Package D20A
(Prototyping Pkg. Only)

TL/DD/6921-4

**COP420, COP320
DIP and SO Wide**

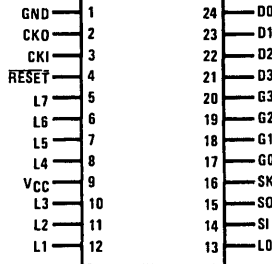


Top View

Order Number COP320-XXX/N
or COP420-XXX/N
See NS Molded Package N28B
Order Number COP320-XXX/D
or COP420-XXX/D
See NS Hermetic Package D28C
(Prototyping Pkg. Only)

TL/DD/6921-2

**COP421, COP321
DIP and SO Wide**



Top View

Order Number COP321-XXX/N or COP421-XXX/N
See NS Molded Package N24A
Order Number COP321-XXX/D or COP421-XXX/D
See NS Hermetic Package D24C
(Prototyping Pkg. Only)
Order Number COP321-XXX/WM or COP421-XXX/WM
See NS Surface Mount Package M24B

TL/DD/6921-3

Pin Descriptions

Pin	Description	Pin	Description
L7-L0	8 bidirectional I/O ports with TRI-STATE	SK	Logic-controlled clock (or general purpose output)
G3-G0	4 bidirectional I/O ports	CKI	System oscillator input
D3-D0	4 general purpose outputs	CKO	System oscillator output (or general purpose input or RAM power supply)
IN3-IN0	4 general purpose inputs (COP420/320 only)	RESET	System reset input
SI	Serial input (or counter input)	V _{CC}	Power supply
SO	Serial output (or general purpose output)	GD	Ground

Timing Diagrams

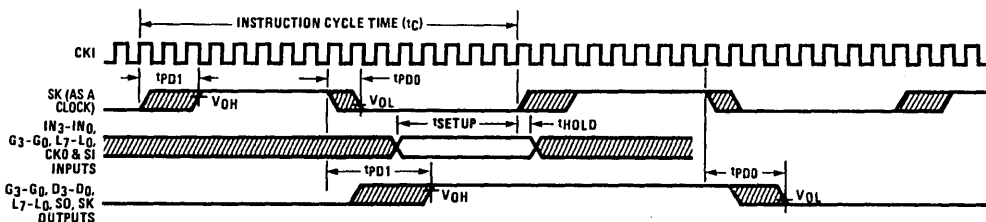
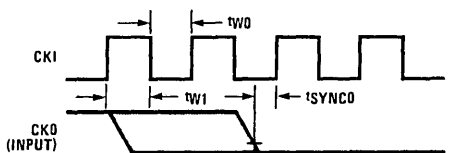


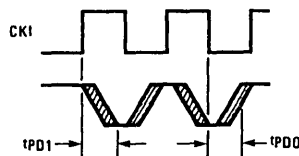
FIGURE 3. Input/Output Timing Diagrams (Crystal Divide by 16 Mode)

TL/DD/6921-5



TL/DD/6921-6

FIGURE 3A. Synchronization Timing



TL/DD/6921-7

FIGURE 3B. CKO Output Timing

Functional Description COP420/COP421/COP422, COP320/COP321/COP322

For ease of reading this description, only COP420 and/or COP421 are referenced; however, all such references apply equally to the COP422, COP322, COP320 and/or COP321, respectively.

A block diagram of the COP420 is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024 byte ROM. As can be seen by an examination of the COP420/421 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 10-bit binary count value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit **B register** whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit **A register** (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load the input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

A **4-bit adder** performs the arithmetic and logic functions of the COP420/421, storing its results in A. It also outputs a carry bit to the 1-bit **C register**, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four **general-purpose inputs, IN₃–IN₀**, are provided; IN₁, IN₂ and IN₃ may be selected, by a mask-programmable option, as Read Strobe, Chip Select and Write Strobe inputs, respectively, for use in MICROBUS applications.

The **D register** provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The **G register** contents are outputs to 4 general-purpose bidirectional I/O ports. G₀ may be mask-programmed as an output for MICROBUS applications.

The **Q register** is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The **8 L drivers**, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The **SIO register** functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers. For example of additional parallel output capacity see **Application #2**.

The XAS instruction copies C into the **SKL latch**. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The **EN register** is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃–EN₀).

1. The least significant bit of the enable register, EN₀ selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0" occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting let each instruction cycle time. The data present at DI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With the EN₁ set the IN₁ input is enabled as an interrupt input. Immediately following an interrupt, EN₁ is reset to disable further interrupts.
3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN enables SO as the output of the SIO shift register outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides summary of the modes associated with EN₃ and EN₁.

OSCILLATOR

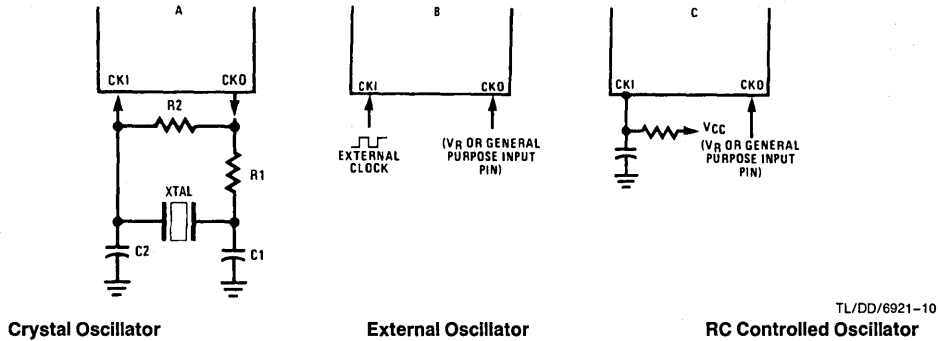
There are three basic clock oscillator configurations available as shown by *Figure 8*.

- a. Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optional by 8).
- b. External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 16 (optional by 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R) of as a general purpose input.
- c. RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available for non-timing functions.

Enable Register Modes—Bits EN₃ and EN₀

EN ₃	EN ₀	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)



TL/DD/6921-10

Crystal Oscillator

Crystal Value	Component Values			
	R1(Ω)	R2(Ω)	C1(pF)	C2(pF)
4 MHz	4.7k	1M	22	22
3.58 MHz	3.3k	1M	22	27
2.09 MHz	8.2k	1M	47	33

RC Controlled Oscillator

R(kΩ)	C(pF)	Instruction Cycle Time (μs)
12	100	5 ± 20%
6.8	220	5.3 ± 23%
8.2	300	8 ± 29%
22	100	8.6 ± 16%

Note: 50 kΩ ≥ R ≥ 5 kΩ
360 pF ≥ C ≥ 50 pF

FIGURE 8. COP420/421/COP320/321 Oscillator

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (VR), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP420/421 system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION (NOT AVAILABLE ON COP422)

Selecting CKO as the RAM power supply (VR) allows the user to shut off the chip power supply (VCC) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. RESET must go low before VCC goes below spec during power off; VCC must be within spec before RESET goes high on power up.
2. VR must be within the operating range of the chip, and equal to VCC ± 1V during normal operation.
3. VR must be ≥ 3.3V with VCC off.

INTERRUPT

The following features are associated with the IN1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC

- + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.
- b. An interrupt will be acknowledged only after the following conditions are met:
 1. EN1 has been set.
 2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN1 input.
 3. A currently executing instruction has been completed.
 4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.
- c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

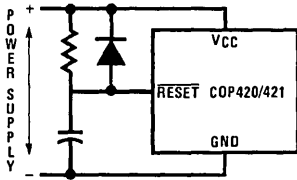
- d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.
- e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic, internal to the COP420/421, will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC} .

Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/6921-13

FIGURE 7. Power-Up Clear Circuit

I/O OPTIONS

COP420/421 outputs have the following optional configurations, illustrated in *Figure 9a*:

- a. **Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC} , compatible with TTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.
- b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.
- c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.
- d. **Standard L**—same as a., but may be disabled. Available on L outputs only.
- e. **Open Drain L**—same as b., but may be disabled. Available on L outputs only.

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

- f. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC} , meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display.
- g. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC} . These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers.

COP420/COP421 inputs have the following optional configurations:

- h. An on-chip depletion load device to V_{CC} .
- i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 9b* for each

of these devices to allow the designer to effectively use these I/O configurations in designing a COP420/421 system.

The SO, SK outputs can be configured as shown in *a.*, *b.*, or *c.* The D and G outputs can be configured as shown in *a.* or *b.* Note that when inputting data to the G ports, the G outputs should be set to "1." The L outputs can be configured as in *d.*, *e.*, *f.* or *g.*

An important point to remember if using configuration *d.* or *f.* with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 9b*, device 2); however, when the L lines are used as input, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to logic "1".

COP421

If the COP420 is bonded as a 24-pin device, it becomes the COP421, illustrated in *Figure 2*, COP420/421 Connection Diagrams. Note that the COP421 does not contain the four general purpose IN inputs (IN_3 – IN_0). Use of this option precludes, of course, use of the IN options and interrupt feature. All other options are available for the COP421.

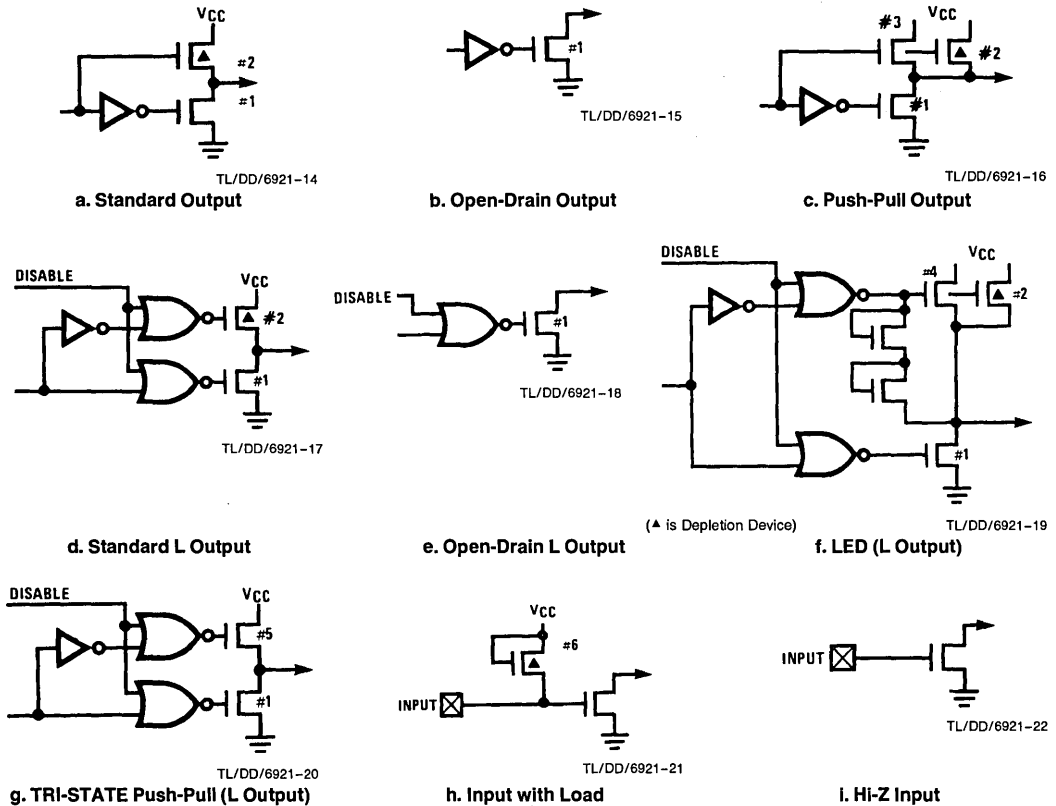


FIGURE 9a. Input/Output Configurations

L-Bus Considerations

False states may be generated on L₀-L₇ during the execution of the CAMQ instruction. The L-ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. The following short program illustrates this situation.

START:

```
CLRA      ;ENABLE THE Q
LEI  4    ;REGISTER TO L LINES
LBI  TEST
STII  3
AISC  12
```

LOOP:

```
LBI  TEST ;LOAD Q WITH X'C3
CAMQ
JP   LOOP
```

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L₀, L₁, L₆, L₇, and logic lows on L₂-L₅ via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L₀, L₁, L₆, L₇, and positive glitches on L₂-L₅. Glitch durations are under 2 microseconds, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics

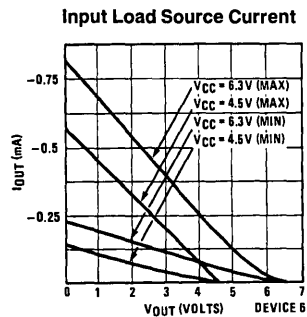
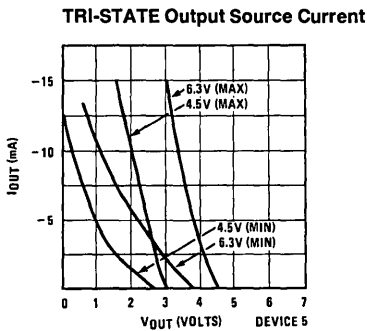
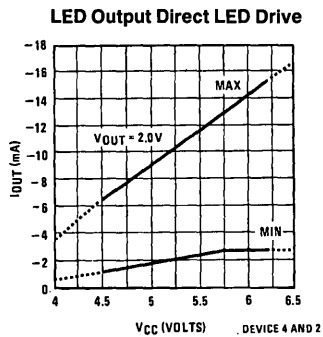
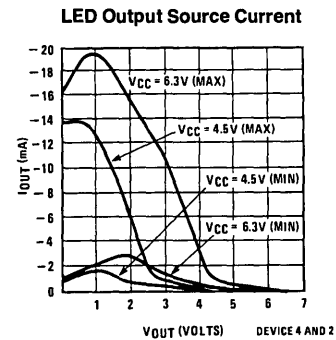
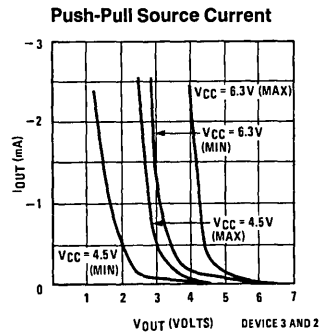
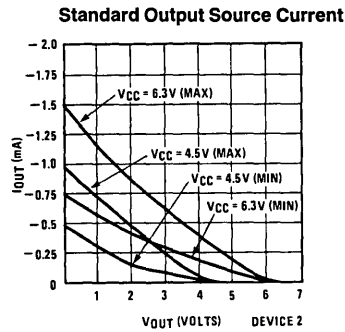
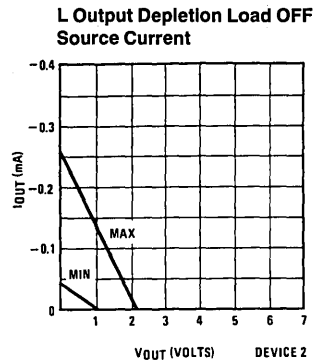
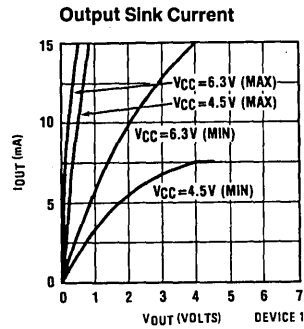


FIGURE 9b. COP420/COP421 Input/Output Characteristics

Typical Performance Characteristics (Continued)

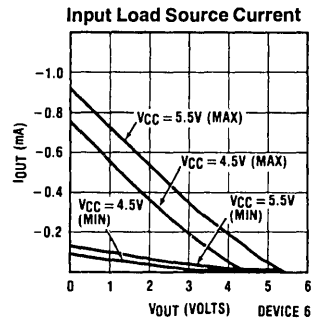
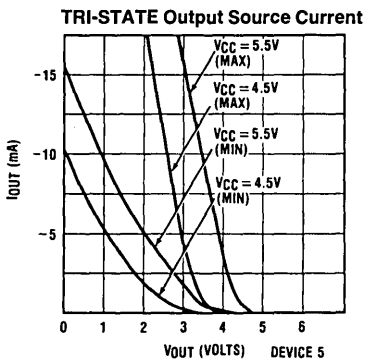
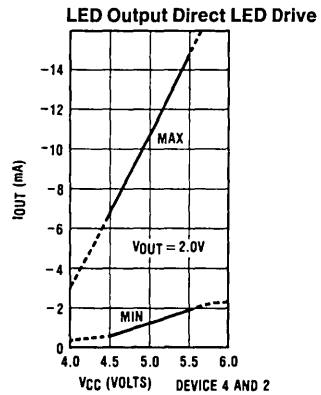
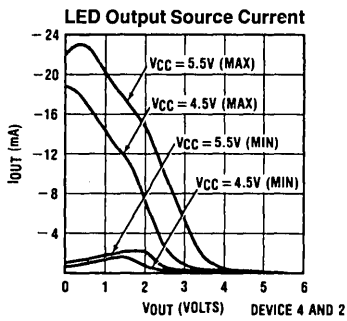
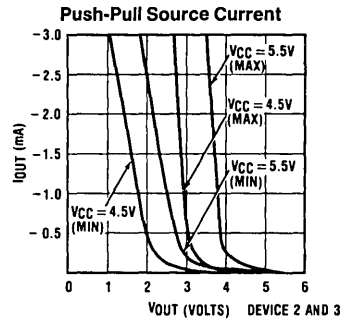
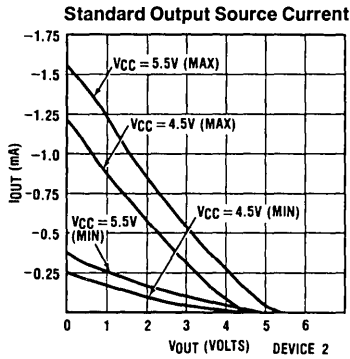
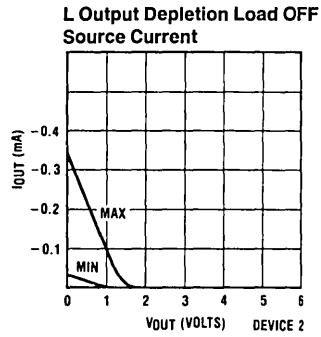
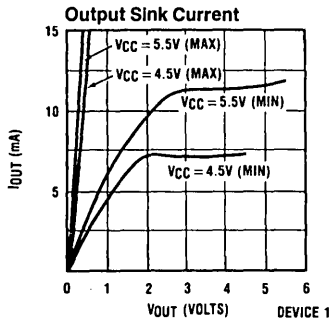


FIGURE 9c. COP320/COP321 Input/Output Characteristics

TL/DD/6921-24

Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP420/COP421/COP422 instruction set.

TABLE I. COP420/421/422/320/321/322 Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	2-bit Operand Field, 0-3 binary (RAM Register Select)
Br	Upper 2 bits of B (register address)	a	10-bit Operand Field, 0-1023 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0-15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register		
G	4-bit Register to latch data for G I/O Port		
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs		
IN	4-bit Input Port		
L	8-bit TRI-STATE I/O Port		
M	4-bit contents of RAM Memory pointed to by B Register		
PC	8-bit ROM Address Register (program counter)		
Q	8-bit Register to latch data for L I/O Port		
SA	10-bit Subroutine Save Register A		
SB	10-bit Subroutine Save Register B		
SC	10 Subroutine Save Register A		
SIO	4-bit Shift Register and Counter		
SK	Logic-Controlled Clock Output		
		OPERATIONAL SYMBOLS	
		+	Plus
		-	Minus
		→	Replaces
		↔	Is exchanged with
		=	Is equal to
		\bar{A}	The one's complement of A
		⊕	Exclusive-OR
		:	Range of values

TABLE II. COP420/421/422/320/321/322 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	$A + C + \text{RAM}(B) \rightarrow A$ $\text{Carry} \rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
ADT		4A	0100 1010	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5-	0101 y	$A + y \rightarrow A$	Carry	Add immediate, Skip on Carry ($y \neq 0$)
CASC		10	0001 0000	$\bar{A} + \text{RAM}(B) + C \rightarrow A$ $\text{Carry} \rightarrow C$	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 0000	$0 \rightarrow A$	None	Clear A
COMP		40	0100 0000	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" $\rightarrow C$	None	Reset C
SC		22	0010 0010	"1" $\rightarrow C$	None	Set C
XOR		02	0000 0010	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description						
TRANSFER OF CONTROL INSTRUCTIONS												
JID		FF	<table border="1"><tr><td>1111</td><td>1111</td></tr></table>	1111	1111	ROM(PC ₈ , A, M) → PC _{7:0}	None	Jump Indirect (Note 3)				
1111	1111											
JMP	a	6--	<table border="1"><tr><td>0110</td><td>00</td><td>a₈</td></tr><tr><td colspan="3">a_{7:0}</td></tr></table>	0110	00	a ₈	a _{7:0}			a → PC	None	Jump
0110	00	a ₈										
a _{7:0}												
JP	a	--	<table border="1"><tr><td>1</td><td>a_{6:0}</td></tr></table> (pages 2,3 only)	1	a _{6:0}	a → PC _{6:0}	None	Jump within Page (Note 4)				
		1	a _{6:0}									
--	<table border="1"><tr><td>11</td><td>a_{5:0}</td></tr></table> (all other pages)	11	a _{5:0}	a → PC _{5:0}								
11	a _{5:0}											
JSRP	a	--	<table border="1"><tr><td>10</td><td>a_{5:0}</td></tr></table>	10	a _{5:0}	PC + 1 → SA → SB → SC 010 → PC _{8:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 5)				
10	a _{5:0}											
JSR	a	6--	<table border="1"><tr><td>0110</td><td>10</td><td>a_{9:8}</td></tr><tr><td colspan="3">a_{7:0}</td></tr></table>	0110	10	a _{9:8}	a _{7:0}			PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
0110	10	a _{9:8}										
a _{7:0}												
RET		48	<table border="1"><tr><td>0100</td><td>1000</td></tr></table>	0100	1000	SC → SB → SA → PC	None	Return from Subroutine				
0100	1000											
RETSK		49	<table border="1"><tr><td>0100</td><td>1001</td></tr></table>	0100	1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip				
0100	1001											
MEMORY REFERENCE INSTRUCTIONS												
CAMQ		33	<table border="1"><tr><td>0011</td><td>0011</td></tr></table>	0011	0011	A → Q _{7:4}	None	Copy A, RAM to Q				
		0011	0011									
3C	<table border="1"><tr><td>0011</td><td>1100</td></tr></table>	0011	1100	RAM(B) → Q _{3:0}								
0011	1100											
CQMA		33	<table border="1"><tr><td>0011</td><td>0011</td></tr></table>	0011	0011	Q _{7:4} → RAM(B)	None	Copy Q to RAM, A				
		0011	0011									
2C	<table border="1"><tr><td>0010</td><td>1100</td></tr></table>	0010	1100	Q _{3:0} → A								
0010	1100											
LD	r	-5	<table border="1"><tr><td>00</td><td>r</td><td>0101</td></tr></table>	00	r	0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r			
00	r	0101										
LDD	r,d	23	<table border="1"><tr><td>0010</td><td>0011</td></tr></table>	0010	0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d				
		0010	0011									
--	<table border="1"><tr><td>00</td><td>r</td><td>d</td></tr></table>	00	r	d								
00	r	d										
LQID		BF	<table border="1"><tr><td>1011</td><td>1111</td></tr></table>	1011	1111	ROM(PC _{9:8} , A, M) → Q SB → SC	None	Load Q Indirect (Note 3)				
1011	1111											
RMB	0	4C	<table border="1"><tr><td>0100</td><td>1100</td></tr></table>	0100	1100	0 → RAM(B) ₀	None	Reset RAM Bit				
		0100	1100									
		45	<table border="1"><tr><td>0100</td><td>0101</td></tr></table>	0100	0101	0 → RAM(B) ₁						
		0100	0101									
42	<table border="1"><tr><td>0100</td><td>0010</td></tr></table>	0100	0010	0 → RAM(B) ₂								
0100	0010											
43	<table border="1"><tr><td>0100</td><td>0011</td></tr></table>	0100	0011	0 → RAM(B) ₃								
0100	0011											
SMB	0	4D	<table border="1"><tr><td>0100</td><td>1101</td></tr></table>	0100	1101	1 → RAM(B) ₀	None	Set RAM Bit				
		0100	1101									
		47	<table border="1"><tr><td>0100</td><td>1101</td></tr></table>	0100	1101	1 → RAM(B) ₁						
		0100	1101									
46	<table border="1"><tr><td>0100</td><td>0110</td></tr></table>	0100	0110	1 → RAM(B) ₂								
0100	0110											
4B	<table border="1"><tr><td>0100</td><td>1011</td></tr></table>	0100	1011	1 → RAM(B) ₃								
0100	1011											
STII	y	7-	<table border="1"><tr><td>0111</td><td>y</td></tr></table>	0111	y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd				
0111	y											
X	r	-6	<table border="1"><tr><td>00</td><td>r</td><td>0110</td></tr></table>	00	r	0110	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r			
00	r	0110										
XAD	r,d	23	<table border="1"><tr><td>0010</td><td>0011</td></tr></table>	0010	0011	RAM(r,d) ↔ A	None	Exchange A with RAM pointed to directly by r,d				
		0010	0011									
--	<table border="1"><tr><td>10</td><td>r</td><td>d</td></tr></table>	10	r	d								
10	r	d										

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	<u>00</u> r <u>0111</u>	RAM(B) \leftrightarrow A Bd - 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	<u>00</u> r <u>0100</u>	RAM(B) \leftrightarrow A Bd + 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	<u>0101</u> <u>0000</u>	A \rightarrow Bd	None	Copy A to Bd
CBA		4E	<u>0100</u> <u>1110</u>	Bd \rightarrow A	None	Copy Bd to A
LBI	r,d	--	<u>00</u> r (d-1) (d = 0,9:15) or <u>0011</u> <u>0011</u> -- <u>10</u> r d (any d)	r,d \rightarrow B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6--	<u>0011</u> <u>0011</u> <u>0010</u> y	y \rightarrow EN	None	Load EN Immediate (Note 7)
XABR		12	<u>0001</u> <u>0010</u>	A \leftrightarrow Br (0,0 \rightarrow A ₃ ,A ₂)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	<u>0010</u> <u>0000</u>		C = "1"	Skip if C is True
SKE		21	<u>0010</u> <u>0001</u>		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	<u>0011</u> <u>0011</u> <u>0010</u> <u>0001</u>		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ	0 1 2 3	33 01 11 03 13	<u>0011</u> <u>0011</u> <u>0000</u> <u>0001</u> <u>0001</u> <u>0001</u> <u>0000</u> <u>0011</u> <u>0010</u> <u>0011</u>	1st byte 2nd byte	G ₀ = 0 G ₁ = 0 G ₂ = 0 G ₃ = 0	Skip if G Bit is Zero
SKMBZ	0 1 2 3	01 11 03 13	<u>0000</u> <u>0001</u> <u>0001</u> <u>0001</u> <u>0000</u> <u>0011</u> <u>0001</u> <u>0011</u>		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
SKT		41	<u>0100</u> <u>0001</u>		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	<u>0011</u> <u>0011</u>	G → A	None	Input G Ports to A
		2A	<u>0010</u> <u>1010</u>			
ININ		33	<u>0011</u> <u>0011</u>	IN → A	None	Input IN Inputs to A (Note 2)
		28	<u>0010</u> <u>1000</u>			
INIL		33	<u>0011</u> <u>0011</u>	IL ₃ , CKO, "0", IL ₀ → A	None	Input IL Latches to A (Note 3)
		29	<u>0010</u> <u>1001</u>			
INL		33	<u>0011</u> <u>0011</u>	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
		2E	<u>0010</u> <u>1110</u>			
OBD		33	<u>0011</u> <u>0011</u>	Bd → D	None	Output Bd to D Outputs
		3E	<u>0011</u> <u>1110</u>			
OGI	y	33	<u>0011</u> <u>0011</u>	y → G	None	Output to G Ports Immediate
		5-	<u>0101</u> <u>y</u>			
OMG		33	<u>0011</u> <u>0011</u>	RAM(B) → G	None	Output RAM to G Ports
		3A	<u>0011</u> <u>1010</u>			
XAS		4F	<u>0100</u> <u>1111</u>	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit register.

Note 2: The ININ instruction is not available on the COP421/COP321 and COP422/COP322 since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP420/421 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If

SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, PC_{9:8}, A, M. PC₉ and PC₈ are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

Description of Selected Instructions (Continued)

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see Figure 10) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A_3 and A_0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A_2 . If CKO has not been so programmed, a "1" will be placed in A_2 . A "0" is always placed in A_1 upon the execution of an INIL. The general purpose inputs IN_3 - IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset.

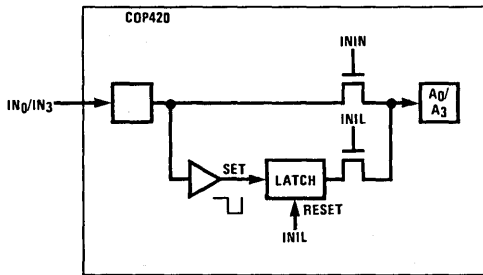


FIGURE 10

TL/DD/6921-25

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9 , PC_8 , A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack ($PC + 1 \rightarrow SA \rightarrow SB \rightarrow SC$) and replaces the least significant 8 bits of PC as follows: $A \rightarrow PC_{7,4}$, $RAM(B) \rightarrow PC_{3,0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" ($SC \rightarrow SB \rightarrow SA \rightarrow PC$), restoring the saved value of PC to continue sequential program execu-

tion. Since LQID pushes $SB \rightarrow SC$, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the content of SB are placed in SC ($SB \rightarrow SC$). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP420/421 to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 131 kHz (crystal frequency $\div 16$) and the binary counter output pulse frequency will be 128 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 128 ticks.

INSTRUCTION SET NOTES

- The first word of a COP420/421 program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed except JID and LQID. LQID and JID take two cycle times if executed and one if skipped.
- The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11 or 15 will access data in the next group of four pages.

Option List

The COP420/421/422 mask-programmable options are assigned numbers which correspond with the COP420 pins.

The following is a list of COP420 options. When specifying a COP421 or COP422 chip, Options 9, 10, 19, 20 and 29 must all be set to zero. When specifying a COP422 chip, Options 21, 22, 27 and 28 must also be zero, and Option 2 must not be a 1. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground—no options available

Option 2: CKO Pin

- = 0: clock generator output to crystal
0 not available if option 3 = 4 or 5
- = 1: Pin is RAM power supply (V_R) input
(Not available on COP422/COP322)
- = 2: general purpose input with load device
- = 4: general purpose Hi Z input

Option 3: CKI Input

- = 0: crystal input divided by 16
- = 1: crystal input divided by 8
- = 2: TTL external clock input divided by 16
- = 3: TTL external clock input divided by 8
- = 4: single-pin RC controlled oscillator ($\div 4$)
- = 5: External Schmitt trigger clock input ($\div 4$)

Option 4: RESET Pin

- = 0: load devices to V_{CC}
- = 1: Hi-Z input

Option 5: L₇ Driver

- = 0: Standard output (*Figure 9D*)
- = 1: Open-Drain output (E)
- = 2: LED direct drive output (F)
- = 3: TRI-STATE push-pull output (G)

Option 6: L₆ Driver

same as Option 5

Option 7: L₅ Driver

same as Option 5

Option 8: L₄ Driver

same as Option 5

Option 9: IN₁ Input

- = 0: load devices to V_{CC} (H)
- = 1: Hi-Z input (I)

Option 10: IN₂ Input

same as Option 9

Option 11 = 0: V_{CC} Pin—no options available

Option 12: L₃ Driver

same as Option 5

Option 13: L₂ Driver

same as Option 5

Option 14: L₁ Driver

same as Option 5

Option 15: L₀ Driver

same as Option 5

Option 16: SI Input
same as Option 9

Option 17: SO Driver

- = 0: standard output (A)
- = 1: open-drain output (B)
- = 2: push-pull output (C)

Option 18: SK Driver

same as Option 17

Option 19: IN₀ Input

same as Option 9

Option 20: IN₃ Input

same as Option 9

Option 21: G₀ I/O Port

- = 0: Standard output (A)
- = 1: Open-Drain output (B)

Option 22: G₁ I/O Port

same as Option 21

Option 23: G₂ I/O Port

same as Option 21

Option 24: G₃ I/O Port

same as Option 21

Option 25: D₃ Output

- = 0: Standard output (A)
- = 1: Open-Drain output (B)

Option 26: D₂ Output

same as Option 25

Option 27: D₁ Output

same as Option 25

Option 28: D₀ Output

same as Option 25

Option 29: COP Function

= 0: normal operation

Option 30: COP Bonding

- = 0: COP420 (28-pin device)
- = 1: COP421 (24-pin device)
- = 2: 28- and 24-pin device
- = 3: COP422 (20-pin device)
- = 4: 28- and 20-pin device
- = 5: 24- and 20-pin device
- = 6: 28-, 24- and 20-pin device

Option 31: In Input Levels

- = 0: normal input levels
- = 1: Higher voltage input levels
("0" = 1.2V, "1" = 3.6V)

Option 32: G Input Levels

same as Option 31

Option 33: L Input Levels

same as Option 31

Option 34: CKO Input Levels

same as Option 31

Option 35: SI Input Levels

same as Option 31

Option List (Continued)

COP OPTION LIST

The following option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE =	0	IS: GROUND PIN
OPTION 2 VALUE =		IS: CKO PIN
OPTION 3 VALUE =		IS: CKI INPUT
OPTION 4 VALUE =		IS: RESET INPUT
OPTION 5 VALUE =		IS: L ₇ DRIVER
OPTION 6 VALUE =		IS: L ₆ DRIVER
OPTION 7 VALUE =		IS: L ₅ DRIVER
OPTION 8 VALUE =		IS: L ₄ DRIVER
OPTION 9 VALUE =		IS: IN ₁ INPUT
OPTION 10 VALUE =		IS: IN ₂ INPUT
OPTION 11 VALUE =		IS: VCG PIN
OPTION 12 VALUE =		IS: L ₃ DRIVER
OPTION 13 VALUE =		IS: L ₂ DRIVER
OPTION 14 VALUE =		IS: L ₁ DRIVER
OPTION 15 VALUE =		IS: L ₀ DRIVER
OPTION 16 VALUE =		IS: SI INPUT
OPTION 17 VALUE =		IS: SO DRIVER
OPTION 18 VALUE =		IS: SK DRIVER
OPTION 19 VALUE =		IS: IN ₀ INPUT
OPTION 20 VALUE =		IS: IN ₃ INPUT
OPTION 21 VALUE =		IS: G ₀ I/O PORT
OPTION 22 VALUE =		IS: G ₁ I/O PORT
OPTION 23 VALUE =		IS: G ₂ I/O PORT
OPTION 24 VALUE =		IS: G ₃ I/O PORT
OPTION 25 VALUE =		IS: D ₃ OUTPUT
OPTION 26 VALUE =		IS: D ₂ OUTPUT
OPTION 27 VALUE =		IS: D ₁ OUTPUT
OPTION 28 VALUE =		IS: D ₀ OUTPUT
OPTION 29 VALUE =	0	IS: COP FUNCTION
OPTION 30 VALUE =		IS: COP BONDING
OPTION 31 VALUE =		IS: IN INPUT LEVELS
OPTION 32 VALUE =		IS: G INPUT LEVELS
OPTION 33 VALUE =		IS: L INPUT LEVELS
OPTION 34 VALUE =		IS: CKO INPUT LEVELS
OPTION 35 VALUE =		IS: SI INPUT LEVELS

TEST MODE (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP420. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

- RAM and Internal Logic Test Mode (SI = 1)
- ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATION #1: COP420 General Controller

Figure 8 shows an interconnect diagram for a COP420 used as a general controller. Operation of the system is as follows:

- The L₇-L₀ outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display.

- The D₃-D₀ outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
- The IN₃-IN₀ inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
- CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a V_R RAM power supply pin. RAM data integrity is thereby assured when the main power supply is shut down (see RAM Keep-Alive option description).
- SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
- The 4 bidirectional G I/O ports (G₃-G₀) are available for use as required by the user's application.

APPLICATION #2: MUSICAL ORGAN AND MUSIC BOX

Software is available on Dial-A-Helper.

Play Mode: Twenty-five musical keys and 25 LEDs are provided to denote F to F with half notes in between. All the keys and LEDs are directly detected and driven by the microprocessor. Depression of the key will give the corresponding musical note and light up the corresponding LED.

Clear: Memory is provided to store a played tune. Depression of the CLEAR key erases the memory and the microprocessor is ready to store new musical notes. A maximum of 28 notes can be stored where each note can be of one to eight musical beats. (Two bytes of memory are required to store one musical note. Any note longer than eight musical beats will require additional memory space for storage.)

Playback: Depression of this button will playback the tune stored in the memory since last "clear."

Preprogrammed Tunes: There are ten preprogrammed tunes (each has an average of 55 notes) masked in the chip. Any tune can be recalled by depressing the "Tune Button" followed by the corresponding "Sharp Key."

Learn Mode: This mode is for the player to learn the ten preprogrammed tunes. By pressing the "Learn Button" followed by the corresponding "Sharp Key," the LEDs will be lighted up one by one to indicate the notes of the selected tune. The LED will remain "on" until the player presses the correct musical key; the LED for the next note will then be lighted up.

Pause: In addition to the 25 musical keys, there is a special pause key. The depression of this key generates a blank note to the memory.

Note: In the Learn Mode when playing "Oh Susanna," the pause key must be used.

Tempo: This is a control input to the musical beat time oscillator for varying the speed of the musical tunes.

Vibrato: This is a switch control to vary the frequency vibration of the note.

Tunes Listing: The following is a listing of the ten preprogrammed tunes: 1) Jingle Bells, 2) Twinkle, Twinkle Little Star, 3) Happy Birthday, 4) Yankee Doodle, 5) Silent Night, 6) This Old Man, 7) London Bridge Is Falling Down, 8) Auld Lang Syne, 9) Oh Susanna, 10) Clementine.

Typical Applications

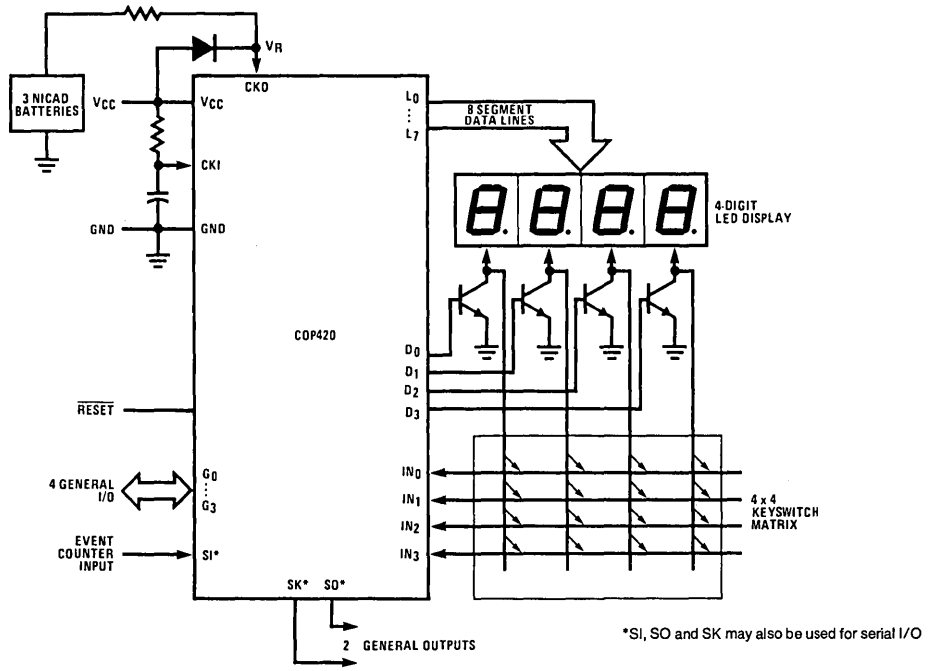
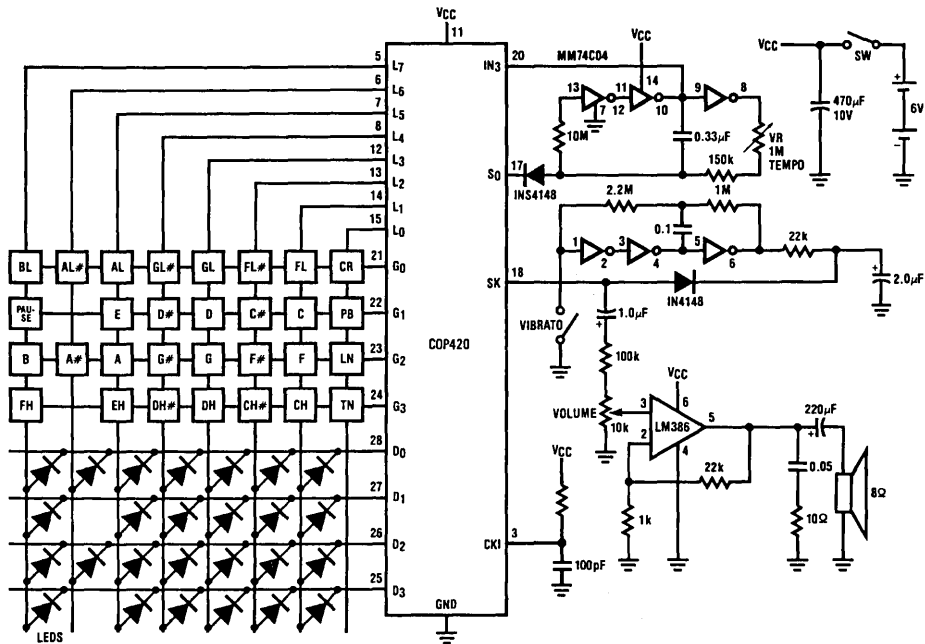


FIGURE 11. COP420 Keyboard Display Interface

TL/DD/6921-26

Circuit Diagram of COP420 Musical Organ

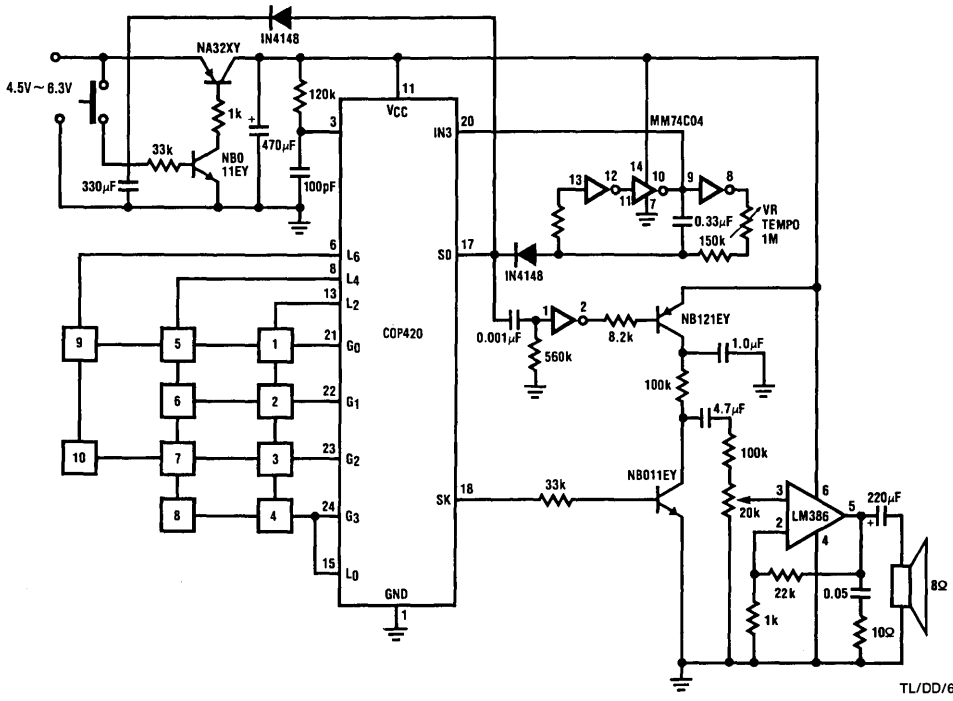


TL/DD/6921-27



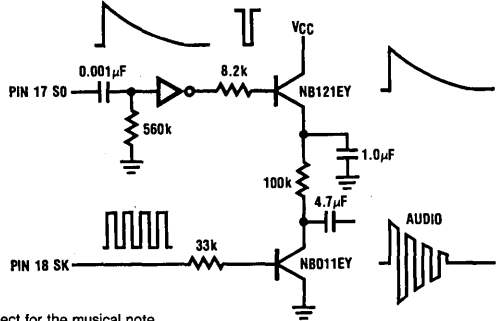
Typical Applications (Continued)

Music Box Application with Direct Key Access



TL/DD/6921-28

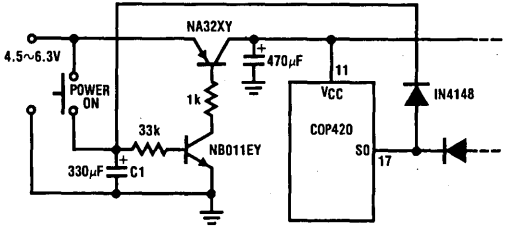
Bell Sound Circuit



This additional circuit provides tinkling effect for the musical note.

TL/DD/6921-29

Auto Power Shut-Off Circuit



This circuit automatically turns off the musical organ if none of the keys are pressed within approximately 30 seconds.

TL/DD/6921-30

COP420L/COP421L/COP422L/COP320L/COP321L/ COP322L Single-Chip N-Channel Microcontrollers

General Description

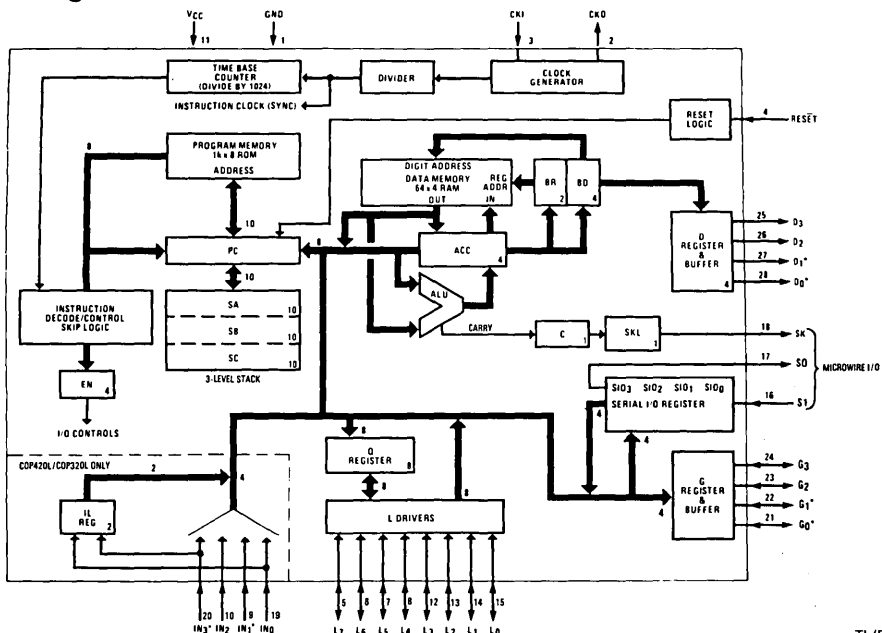
The COP420L, COP421L, COP422L, COP320L, COP321L, and COP322L Single-Chip N-Channel Microcontrollers are members of the COPSTM family, fabricated using N-channel, silicon gate MOS technology. These controller oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP421L and COP422L are identical to the COP420L, but with 19 and 15 I/O lines, respectively, instead of 23. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

The COP320L, COP321L, and COP322L are exact functional equivalents, but extended temperature range versions, of the COP420L, COP421L, and COP422L respectively.

Features

- Low cost
- Powerful instruction set
- 1k x 8 ROM, 64 x 4 RAM
- 23 I/O lines (COP420L)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 16 μ s instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (9 mA max)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device—
COP320L/COP321L/COP322L (–40°C to +85°C)

Block Diagram



*Not available on COP422L/COP322L

FIGURE 1

TL/DD/8825-1

COP420L/COP421L/COP422L**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Power Dissipation		
COP420L/COP421L	0.75W at 25°C	0.4W at 70°C
COP422L	0.65W at 25°C	0.3W at 70°C

Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	6.3	V
Power Supply Ripple (Notes 1, 4)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		9	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷32, ÷16, ÷8)				
Logic High (V _{IH})	V _{CC} = Max	3.0		V
Logic High (V _{IH})	V _{CC} = 5V ±5%	2.0		V
Logic Low (V _{IL})		-0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V _{IH})		0.7 V _{CC}		V
Logic Low (V _{IL})		-0.3	0.6	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
SO Input Level (Test Mode)	(Note 3)	2.0	2.5	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	with TTL Trip Level Options	2.0		V
Logic Low	Selected, V _{CC} = 5V ±5%	-0.3	0.8	V
Logic High	with High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 4)			7	pF
Hi-Z Input Leakage		-1	+1	μA
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ±10%			
Logic High (V _{OH})	I _{OH} = -25 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 ma		0.4	V
CMOS Operation (Note 2)	V _{CC} = 4.5V			
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.8V for normal operation.

Note 4: This parameter is only sampled and not 100% tested.

COP420L/COP421L/COP422L**DC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 6.3\text{V}$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OL}} = 0.4\text{V}$	1.2		mA
	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OL}} = 0.4\text{V}$	0.9		mA
L_0 – L_7 Outputs and Standard	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OL}} = 0.4\text{V}$	0.4		mA
G_0 – G_3 , D_0 – D_3 Outputs (I_{OL})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OL}} = 0.4\text{V}$	0.4		mA
G_0 – G_3 and D_0 – D_3 Outputs with	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OL}} = 1.0\text{V}$	11		mA
High Current Options (I_{OL})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OL}} = 1.0\text{V}$	7.5		mA
G_0 – G_3 and D_0 – D_3 Outputs with	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OL}} = 1.0\text{V}$	22		mA
Very High Current Options (I_{OL})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OL}} = 1.0\text{V}$	15		mA
CKI (Single-Pin RC Oscillator)	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{IH}} = 3.5\text{V}$	2		mA
CKO	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OL}} = 0.4\text{V}$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OH}} = 2.0\text{V}$	–75	–480	μA
All Outputs (I_{OH})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OH}} = 2.0\text{V}$	–30	–250	μA
Push-Pull Configuration	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OH}} = 2.4\text{V}$	–1.4		mA
SO and SK Outputs (I_{OH})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OH}} = 1.0\text{V}$	–1.2		mA
LED Configuration, L_0 – L_7				
Outputs, Low Current	$V_{\text{CC}} = 6.0\text{V}$, $V_{\text{OH}} = 2.0\text{V}$	–1.5	–13	mA
Driver Option (I_{OH})				
LED Configuration, L_0 – L_7				
Outputs, High Current	$V_{\text{CC}} = 6.0\text{V}$, $V_{\text{OH}} = 2.0\text{V}$	–3.0	–25	mA
Driver Option (I_{OH})				
TRI-STATE Configuration,	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OH}} = 3.2\text{V}$	–0.8		mA
L_0 – L_7 Outputs, Low	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OH}} = 1.5\text{V}$	–0.9		mA
Current Driver Option (I_{OH})				
TRI-STATE Configuration,	$V_{\text{CC}} = 6.3\text{V}$, $V_{\text{OH}} = 3.2\text{V}$	–1.6		mA
L_0 – L_7 Outputs, High	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OH}} = 1.5\text{V}$	–1.8		mA
Current Driver Option (I_{OH})				
Input Load Source Current	$V_{\text{CC}} = 5.0\text{V}$	–10	–140	μA
TRI-STATE Output Leakage Current		–2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7 – L_4			4	mA
L_3 – L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7 – L_4			60	mA
L_3 – L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

COP320L/COP321L/COP322L

Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	-0.5V to +10V	Total Source Current	120 mA
Ambient Operating Temperature	-40°C to +85°C	Total Sink Current	120 mA
Ambient Storage Temperature	-65°C to +150°C	<i>Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.</i>	
Lead Temperature (Soldering, 10 sec.)	300°C		
Power Dissipation			
COP320L/COP321L	0.75W at 25°C 0.4W at 70°C 0.25W at 85°C		
COP322L	0.65W at 25°C 0.20W at 70°C		

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	5.5	V
Power Supply Ripple (Notes 1, 4)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		11	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V _{IH})	V _{CC} = Max	3.0		V
Logic High (V _{IH})	V _{CC} = 5V ±5%	2.2		V
Logic Low (V _{IL})		-0.3	0.3	V
Schmitt Trigger Input				
Logic High (V _{IH})		0.7 V _{CC}		V
Logic Low (V _{IL})		-0.3	0.4	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.4	V
SO Input Level (Test Mode)	(Note 3)	2.2	2.5	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	with TTL Trip Level Options	2.2		V
Logic Low	Selected, V _{CC} = 5V ±5%	-0.3	0.6	V
Logic High	with High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 4)			7	pF
Hi-Z Input Leakage		-2	+2	μA
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ±10%			
Logic High (V _{OH})	I _{OH} = -20 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)	V _{CC} = 4.5V			
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

Note 4: This parameter is only sampled and not 100% tested.

COP320L/COP321L/COP322L**DC Electrical Characteristics**-40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	V _{CC} = 5.5V, V _{OL} = 0.4V	1.0		mA
	V _{CC} = 4.5V, V _{OL} = 0.4V	0.8		mA
L ₀ -L ₇ Outputs and Standard	V _{CC} = 5.5V, V _{OL} = 0.4V	0.4		mA
G ₀ -G ₃ and D ₀ -D ₃ Outputs (I _{OL})	V _{CC} = 4.5V, V _{OL} = 0.4V	0.4		mA
G ₀ -G ₃ and D ₀ -D ₃ Outputs with	V _{CC} = 5.5V, V _{OL} = 1.0V	9		mA
High Current Options (I _{OL})	V _{CC} = 4.5V, V _{OL} = 1.0V	7		mA
G ₀ -G ₃ and D ₀ -D ₃ Outputs with	V _{CC} = 5.5V, V _{OL} = 1.0V	18		mA
Very High Current Options (I _{OL})	V _{CC} = 4.5V, V _{OL} = 1.0V	14		mA
CKI (Single-Pin RC Oscillator)	V _{CC} = 4.5V, V _{IH} = 3.5V	2		mA
CKO	V _{CC} = 4.5V, V _{OL} = 0.4V	0.2		mA
Output Source Current				
Standard Configuration,				
All Outputs (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-55	-600	μA
	V _{CC} = 4.5V, V _{OH} = 2.0V	-28	-350	μA
Push-Pull Configuration				
SO and SK Outputs (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.1		mA
	V _{CC} = 4.5V, V _{OH} = 1.0V	-1.2		mA
LED Configuration, L₀-L₇				
Outputs, Low Current	V _{CC} = 6.0V, V _{OH} = 2.0V	-1.4	-17	mA
Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-0.7	-15	mA
LED Configuration, L₀-L₇				
Outputs, High Current	V _{CC} = 6.0V, V _{OH} = 2.0V	-2.7	-34	mA
Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.4	-30	mA
TRI-STATE Configuration,				
L ₀ -L ₇ Outputs, Low	V _{CC} = 5.5V, V _{OH} = 2.7V	-0.6		mA
Current Driver Option (I _{OH})	V _{CC} = 4.5V, V _{OH} = 1.5V	-0.9		mA
TRI-STATE Configuration,				
L ₀ -L ₇ Outputs, High	V _{CC} = 5.5V, V _{OH} = 2.7V	-1.2		mA
Current Driver Option (I _{OH})	V _{CC} = 4.5V, V _{OH} = 1.5V	-1.8		mA
Input Load Source Current	V _{CC} = 5.0V	-10	-200	μA
TRI-STATE Output Leakage Current		-5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L ₇ -L ₄			4	mA
L ₃ -L ₀			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L ₇ -L ₄			60	mA
L ₃ -L ₀			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics

COP420L/COP421L/COP422L: $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted

COP320L/COP321L/COP322L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time— t_C		16	40	μs
CKI				
Input Frequency— f_i	$\div 32$ Mode	0.8	2.0	MHz
	$\div 16$ Mode	0.4	1.0	MHz
	$\div 8$ Mode	0.2	0.5	MHz
	$\div 4$ Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time (Note 2)	$f_i = 2\text{ MHz}$		120	ns
Fall Time (Note 2)			80	ns
CKI Using RC ($\div 4$)	$R = 56\text{ k}\Omega \pm 5\%$ $C = 100\text{ pF} \pm 10\%$	16	28	μs
Instruction Cycle Time (Note 1)				
CKO as SYNC Input		400		ns
t_{SYNC}				
INPUTS:				
$\text{IN}_3\text{-IN}_0, \text{G}_3\text{-G}_0, \text{L}_7\text{-L}_0$				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: $C_L = 50\text{ pF}, R_L = 20\text{ k}\Omega, V_{\text{OUT}} = 1.5\text{V}$			
SO, SK Outputs			4.0	μs
$t_{\text{pd1}}, t_{\text{pd0}}$				
All Other Outputs			5.6	μs
$t_{\text{pd1}}, t_{\text{pd0}}$				

Note 1: Variation due to the device included.

Note 2: This parameter is only sampled and not 100% tested.

Timing Diagrams

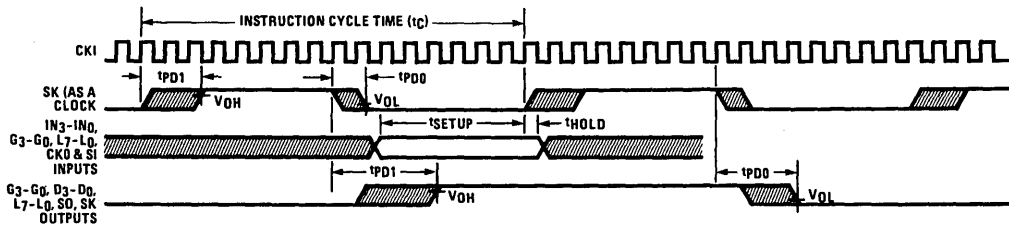


FIGURE 3. Input/Output Timing Diagrams (Crystal Divide-by-16 Mode)

TL/DD/8825-5

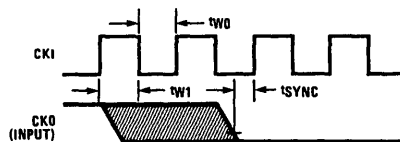
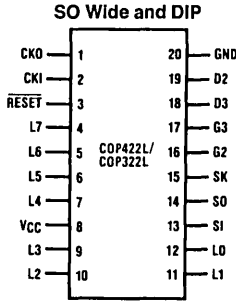


FIGURE 3a. Synchronization Timing

TL/DD/8825-6

Connection Diagrams



TL/DD/8825-4

Top View

Order Number COP422L-XXX/N
or COP322L-XXX/N

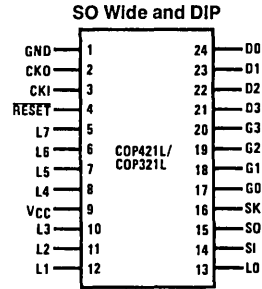
See NS Molded Package Number N24A

Order Number COP322L-XXX/D
or COP422L-XXX/D

See NS Hermetic Package Number D20A
(Prototyping Package Only)

Order Number COP322L-XXX/WM
or COP422L-XXX/WM

See NS Surface Mount Package Number M20B



TL/DD/8825-3

Top View

Order Number COP421L-XXX/N
or COP321L-XXX/N

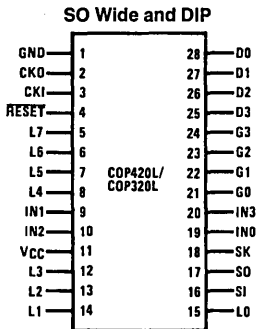
See NS Molded Package Number N20A

Order Number COP321L-XXX/D
or COP421L-XXX/D

See NS Hermetic Package Number D24C
(Prototyping Package Only)

Order Number COP321L-XXX/WM
or COP421L-XXX/WM

See NS Surface Mount Package
Number M24B



TL/DD/8825-2

Top View

Order Number COP420L-XXX/N
or COP320L-XXX/N

See NS Molded Package Number N28B

Order Number COP320L-XXX/D
or COP420L-XXX/D

See NS Hermetic Package Number D28C

FIGURE 2

Pin Descriptions

- L₇-L₀ 8 bidirectional I/O ports with TRI-STATE
- G₃-G₀ 4 bidirectional I/O ports
- D₃-D₀ 4 general purpose outputs
- IN₃-IN₀ 4 general purpose inputs (COP420L only)
- SI Serial input (or counter input)
- SO Serial output (or general purpose output)
- SK Logic-controlled clock (or general purpose output)
- CKI System oscillator input
- CKO System oscillator output or general purpose input
- RESET System reset input
- V_{CC} Power supply
- GND Ground

Functional Description

For ease of reading this description, only COP420L and/or COP421L are referenced; however, all such references apply also to COP320L, COP321L, COP322L, or COP422L.

A block diagram of the COP420L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024 byte ROM. As can be seen by an examination of the COP420L/421L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 10-bit binary count value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions is based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP420/421L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or

can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four general-purpose inputs, IN_3 – IN_0 , are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers. For example of additional parallel output capacity see Application #2.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3 – EN_0).

1. The least significant bit of the enable register, EN_0 , selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3 . With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables

Functional Description (Continued)

the L drivers, placing the L I/O ports in a high-impedance input state.

4. EN_3 , in conjunction with EN_0 , affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3 . With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted

data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides a summary of the modes associated with EN_3 and EN_0 .

Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address ($PC + 1$) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level ($PC + 1 \rightarrow SA \rightarrow SB \rightarrow SC$). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.
- b. An interrupt will be acknowledged only after the following conditions are met:
 1. EN_1 has been set.
 2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
 3. A currently executing instruction has been completed.
 4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).
- c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at address 0FF. At the end of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be

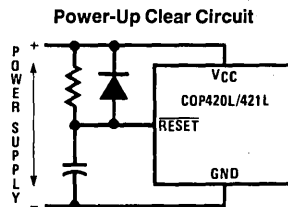
nested within the interrupt servicing routine since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

- d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.
- e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below. The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC} . Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/8825-7

$RC \geq 5 \times$ Power Supply Rise Time

Functional Description (Continued)

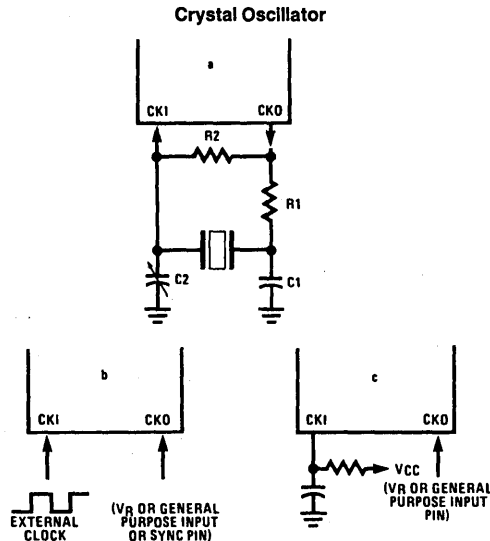
OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 4*.

- Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32 (optional by 16 or 8).
- External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 32 (optional by 16 or 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R) or as a general purpose input.
- RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or as a general purpose input.

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP420L/421L system timing configuration does not require use of the CKO pin.



TL/DD/8825-8

Crystal Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220
2.097 MHz	1k	1M	30	6-36

RC Controlled Oscillator

R (k Ω)	C (pF)	Instruction Cycle Time (μ s)
51	100	19 \pm 15%
82	56	19 \pm 13%

Note: $200k \geq R \geq 25k$
 $360 \text{ pF} \geq C \geq 50 \text{ pF}$

FIGURE 4. COP420L/421L Oscillator

Functional Description (Continued)

I/O OPTIONS

COP420L/421L outputs have the following optional configurations, illustrated in *Figure 5*:

- a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC} , compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.
- b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.
- c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.
- d. **Standard L**—same as a., but may be disabled. Available on L outputs only.
- e. **Open Drain L**—same as b., but may be disabled. Available on L outputs only.
- f. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC} , meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.
- g. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC} . These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

COP420L/COP421L inputs have the following optional configurations:

- h. An on-chip depletion load device to V_{CC} .
- i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP420L/421L system.

The SO, SK outputs can be configured as shown in a., b., or c. The D and G outputs can be configured as shown in a. or b. Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in d., e., f. or g.

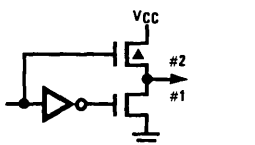
An important point to remember if using configuration d. or f. with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L lines are used as inputs, the disabled depletion device *cannot* be relied on to source sufficient current to pull an input to a logic 1.

COP421L

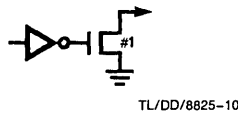
If the COP420L is bonded as a 24-pin device, it becomes the COP421L, illustrated in *Figure 2*, COP420L/421L Connection Diagrams. Note that the COP421L does not contain the four general purpose IN inputs (IN_3 – IN_0). Use of this option precludes, of course, use of the IN options and the interrupt feature. All other options are available for the COP421L.

COP422L

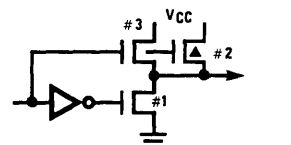
If the COP421L is bonded as a 20-pin device, it becomes the COP422L, as illustrated in *Figure 2*. Note that the COP422L contains all the COP421L pins except D_0 , D_1 , G_0 , and G_1 .



a. Standard Output



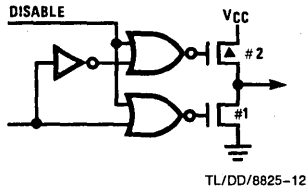
b. Open-Drain Output



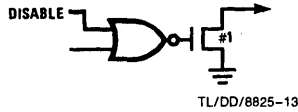
c. Push-Pull Output

FIGURE 5. Output Configurations

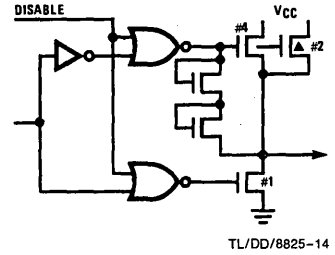
Functional Description (Continued)



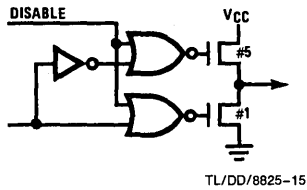
d. Standard L Output



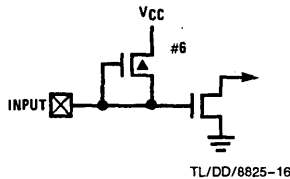
e. Open-Drain L Output



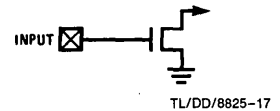
f. LED (L Output)
(▲ is Depletion Device)



g. TRI-STATE Push-Pull (L Output)



h. Input with Load



i. HI-Z Input

FIGURE 5. Output Configurations (Continued)

L-Bus Considerations

False states may be generated on L₀-L₇ during the execution of the CAMQ instruction. The L-ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. the following short program that illustrates this situation.

```

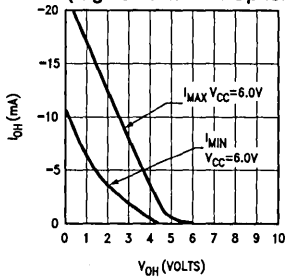
START:
    CLRA      ;ENABLE THE Q
    LEI 4    ;REGISTER TO L LINES
    LBI TEST
    STII 3
    AISC 12

LOOP:
    LBI TEST ;LOAD Q WITH X'C3
    CAMQ
    JP LOOP
    
```

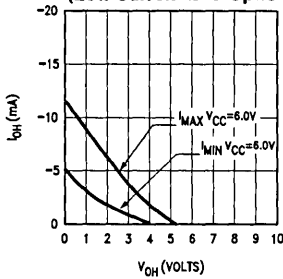
In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L₀, L₁, L₆, L₇, and logic lows on L₂-L₅ via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L₀, L₁, L₆, L₇, and positive glitches on L₂-L₅. Glitch durations are under 2 μs, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics

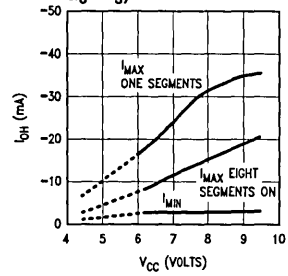
LED Output Source Current (High Current LED Option)



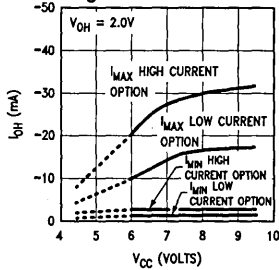
LED Output Source Current (Low Current LED Option)



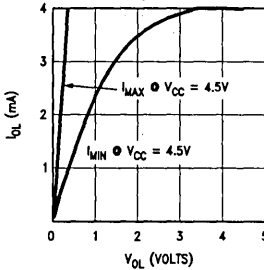
LED Output Direct Segment and Digit Drive (High Current Options on L₀-L₇; Very High Current Options on D₀-D₃ and G₀-G₃)



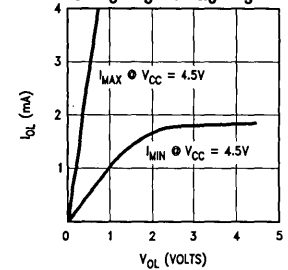
LED Output Direct Segment Drive



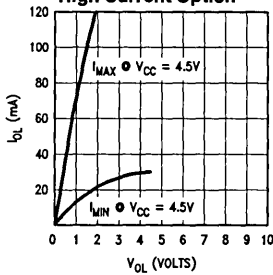
Output Sink Current for SO and SK



Output Sink Current for L₀-L₇ and Standard Drive Option for D₀-D₃ and G₀-G₃



Output Sink Current for G₀-G₃ and D₀-D₃ with Very High Current Option



Output Sink Current for G₀-G₃ and D₀-D₃ (High Current Option)

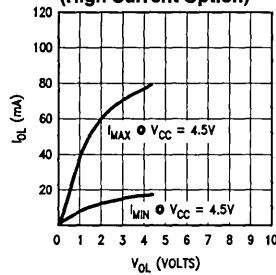
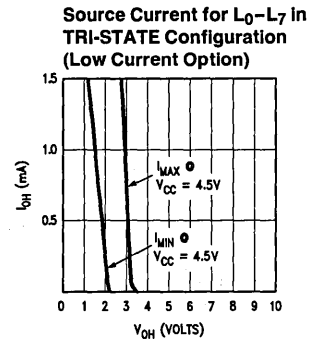
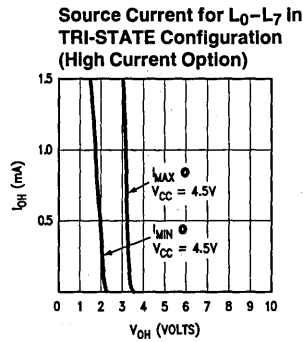
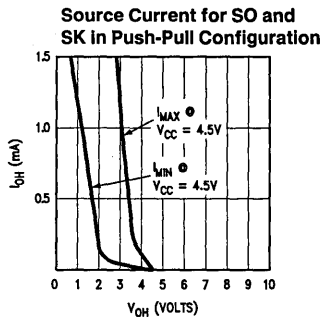
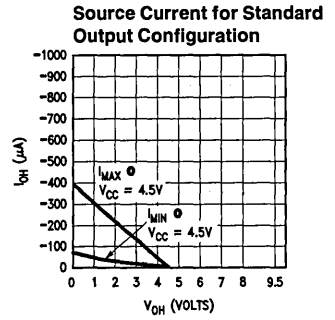
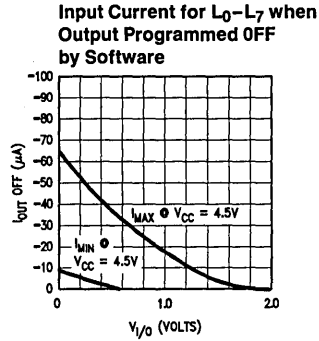
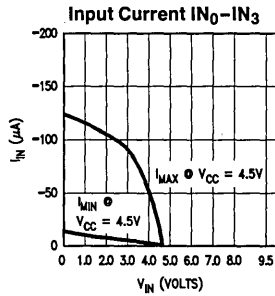


FIGURE 6. COP420L/COP421L/COP422L Input/Output Characteristics

TL/DD/8825-19

Typical Performance Characteristics (Continued)



TL/DD/8825-18

Typical Performance Characteristics (Continued)

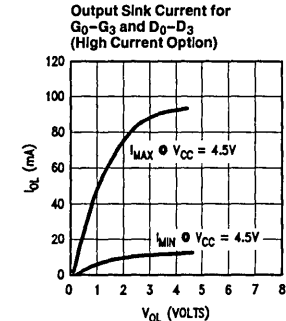
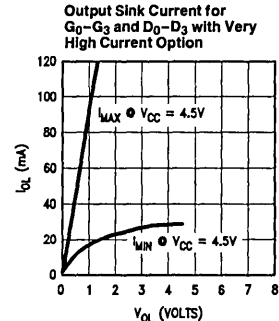
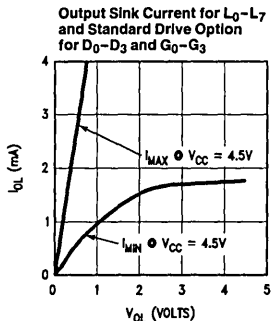
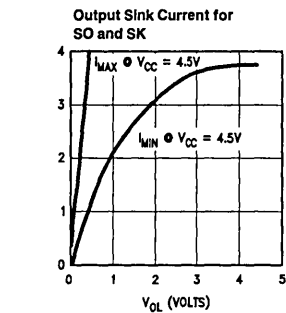
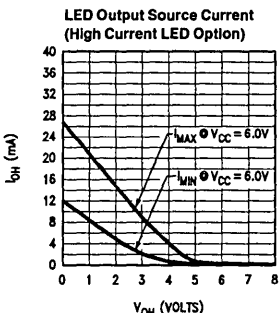
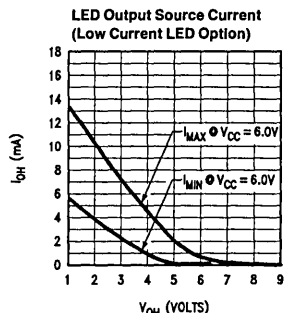
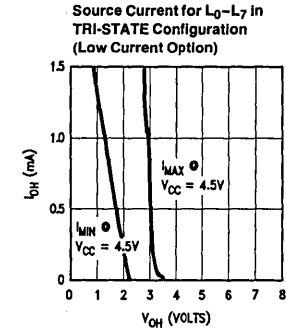
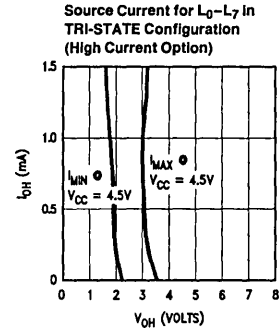
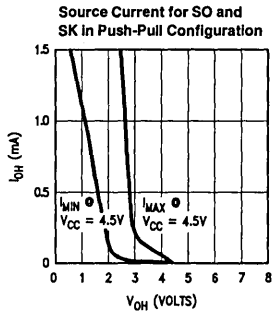
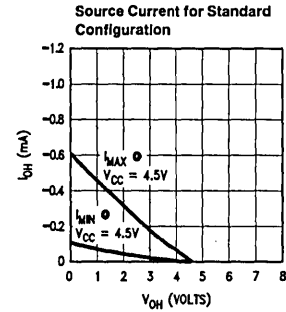
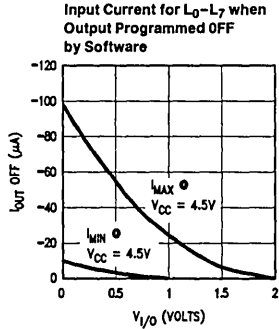
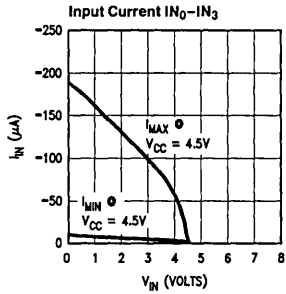


FIGURE 7. COP320L/DOP321L/COP322L Input/Output Characteristics

TL/DD/8825-20

COP420L/COP421L Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410L/411L instruction set.

TABLE I. COP420L/421L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit Latches associated with the IN ₃ or IN ₀ inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	10-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	10-bit Operand Field, 0–1023 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t

OPERATIONAL SYMBOLS

+	Plus
–	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
\bar{A}	The ones complement of A
⊕	Exclusive-OR
:	Range of values

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
ADT		4A	<u>0100</u> <u>1010</u>	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5-	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CASC		10	<u>0001</u> <u>0000</u>	$\bar{A} + \text{RAM}(B) + C \rightarrow A$ Carry $\rightarrow C$	Carry	Compliment and Add with Carry, Skip on Carry
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	Ones complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	<u>1111</u> <u>1111</u>	ROM ($PC_{9:8}, A, M$) $\rightarrow PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6- --	<u>0110</u> <u>00</u> <u>a_{9:8}</u> <u>a_{7:0}</u>	$a \rightarrow PC$	None	Jump
JP	a	--	<u>1</u> <u>a_{6:0}</u> (pages 2,3 only) or <u>11</u> <u>a_{5:0}</u> (all other pages)	$a \rightarrow PC_{6:0}$ $a \rightarrow PC_{5:0}$	None	Jump within Page (Note 4)
JSRP	a	--	<u>10</u> <u>a_{5:0}</u>	$PC + 1 \rightarrow SA \rightarrow SB \rightarrow SC$ $0010 \rightarrow PC_{9:6}$ $a \rightarrow PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6- --	<u>0110</u> <u>10</u> <u>a_{9:8}</u> <u>a_{7:0}</u>	$PC + 1 \rightarrow SA \rightarrow SB \rightarrow SC$ $a \rightarrow PC$	None	Jump to Subroutine
RET		48	<u>0100</u> <u>1000</u>	$SC \rightarrow SB \rightarrow SA \rightarrow PC$	None	Return from Subroutine
RETSK		49	<u>0100</u> <u>1001</u>	$SC \rightarrow SB \rightarrow SA \rightarrow PC$	Always Skip on Return	Return from Subroutine then Skip

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	<u>0011</u> <u>0011</u>	A → Q _{7:4}	None	Copy A, RAM to Q
		3C	<u>0011</u> <u>1100</u>	RAM(B) → Q _{3:0}		
CQMA		33	<u>0011</u> <u>0011</u>	Q _{7:4} → RAM(B)	None	Copy Q to RAM, A
		2C	<u>0010</u> <u>1100</u>	Q _{3:0} → A		
LD	r	-5	<u>00</u> r <u>0101</u>	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23	<u>0010</u> <u>0011</u>	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		--	<u>00</u> r d			
LQID		BF	<u>1011</u> <u>1111</u>	ROM(PC _{9:8} ,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	<u>0100</u> <u>1100</u>	0 → RAM(B) ₀	None	Reset RAM Bit
	1	45	<u>0100</u> <u>0101</u>	0 → RAM(B) ₁		
	2	42	<u>0100</u> <u>0010</u>	0 → RAM(B) ₂		
	3	43	<u>0100</u> <u>0011</u>	0 → RAM(B) ₃		
SMB	0	4D	<u>0100</u> <u>1101</u>	1 → RAM(B) ₀	None	Set RAM Bit
	1	47	<u>0100</u> <u>1101</u>	1 → RAM(B) ₁		
	2	46	<u>0100</u> <u>0110</u>	1 → RAM(B) ₂		
	3	4B	<u>0100</u> <u>1011</u>	1 → RAM(B) ₃		
STII	y	7-	<u>0111</u> y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	<u>00</u> r <u>0110</u>	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	<u>0010</u> <u>0011</u>	RAM(r,d) ↔ A	None	Exchange A with RAM pointed to directly by (r,d)
		--	<u>10</u> r d			
XDS	r	-7	<u>00</u> r <u>0111</u>	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	<u>00</u> r <u>0100</u>	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	<u>0101</u> <u>0000</u>	A → Bd	None	Copy A to Bd
CBA		4E	<u>0100</u> <u>1110</u>	Bd → A	None	Copy Bd to A
LBI	r,d	--	<u>00</u> <u>r</u> <u>(d-1)</u> (d=0,9;15) or <u>0011</u> <u>0011</u> -- <u>10</u> <u>r</u> <u>d</u> (any d)	r,d → B	Skip until not an LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6-	<u>0011</u> <u>0011</u> <u>0110</u> <u>y</u>	y → EN	None	Load EN Immediate (Note 7)
XABR		12	<u>0001</u> <u>0010</u>	A ↔ Br (0,0 → A ₃ ,A ₂)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	<u>0010</u> <u>0000</u>		C = "1"	Skip if C is True
SKE		21	<u>0010</u> <u>0001</u>		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	<u>0011</u> <u>0011</u> <u>0010</u> <u>0001</u>		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	<u>0011</u> <u>0011</u>	1st byte		Skip if G Bit is Zero
	0	01	<u>0000</u> <u>0001</u>	} 2nd byte	G ₀ = 0	
	1	11	<u>0001</u> <u>0001</u>		G ₁ = 0	
	2	03	<u>0000</u> <u>0011</u>		G ₂ = 0	
	3	13	<u>0001</u> <u>0011</u>		G ₃ = 0	
SKMBZ		0 1 2 3	<u>0000</u> <u>0001</u> <u>0001</u> <u>0001</u> <u>0000</u> <u>0011</u> <u>0001</u> <u>0011</u>		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
SKT		41	<u>0100</u> <u>0001</u>		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description				
INPUT/OUTPUT INSTRUCTIONS										
ING		33 2A	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1010</td></tr></table>	0011	0011	0010	1010	G → A	None	Input G Ports to A
0011	0011									
0010	1010									
ININ		33 28	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1000</td></tr></table>	0011	0011	0010	1000	IN → A	None	Input IN Inputs to A (Note 2)
0011	0011									
0010	1000									
INIL		33 29	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1001</td></tr></table>	0011	0011	0010	1001	IL ₃ , CKO, "0", IL ₀ → A	None	Input IL Latches to A (Note 3)
0011	0011									
0010	1001									
INL		33 2E	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1110</td></tr></table>	0011	0011	0010	1110	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
0011	0011									
0010	1110									
OBD		33 3E	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0011</td><td>1110</td></tr></table>	0011	0011	0011	1110	Bd → D	None	Output Bd to D Outputs
0011	0011									
0011	1110									
OGI	y	33 5-	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0101</td><td>y</td></tr></table>	0011	0011	0101	y	y → G	None	Output to G Ports Immediate
0011	0011									
0101	y									
OMG		33 3A	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0011</td><td>1010</td></tr></table>	0011	0011	0011	1010	RAM(B) → G	None	Output RAM to G Ports
0011	0011									
0011	1010									
XAS		4F	<table border="1"><tr><td>0100</td><td>1111</td></tr></table>	0100	1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 3)		
0100	1111									

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is only available on the 28-pin COP420L as the other devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data minus 1, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP420L/421L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If

SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 10-bit word, PC_{9:8}, A, M. PC_{9:8}, A, M. PC₉ and PC₈ are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

Description of Selected Instructions (Continued)

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see Figure 8) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3 – IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction. IL latches are *not cleared* on reset.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9 , PC_8 , A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack ($PC + 1 \rightarrow SA \rightarrow SB \rightarrow SC$) and replaces the least significant 8 bits of PC as follows: $A \rightarrow PC_{7,4}$, $RAM(B) \rightarrow PC_{3,0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" ($SC \rightarrow SB \rightarrow SA \rightarrow PC$), restoring the saved value of PC to continue sequential program execution. Since LQID pushes $SB \rightarrow SC$, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC ($SB \rightarrow SC$). Note the LQID takes two instruction cycle times to execute.

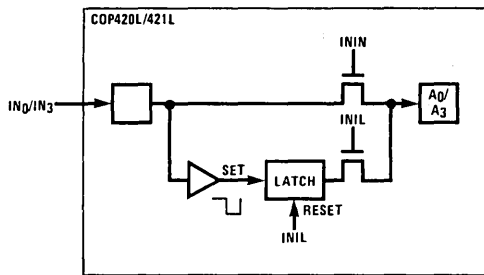
SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP420L/421L to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency \div 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

- The first word of a COP420L/421L program (ROM address 0) must be a CLR A (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.
- The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, or 15 will access data in the next group of four pages.



TL/DD/8825-21

FIGURE 8. INIL Hardware Implementation

Option List

The COP420L/421L mask-programmable options are assigned numbers which correspond with the COP420L pins.

The following is a list of COP420L options. When specifying a COP421L chip, Options 9, 10, 19, and 20 must all be set to zero. When specifying a COP422L chip, options 9, 10, 19, and 20 must all be set to zero; options 21 and 22 may not be set to one, three or five; and option 2 may not be set to one. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

The Option Table should be copied and sent in with your EPROM or disc.

- Option 1 = 0: Ground Pin—no options available
- Option 2: CKO Output
 = 0: clock generator output to crystal/resonator (0 not allowable value if Option 3 = 3)
 = 1: not an option
 = 2: general purpose input with load device to V_{CC}
 = 3: general purpose input, Hi-Z
- Option 3: CKI Input
 = 0: oscillator input divided by 32 (2 MHz max.)
 = 1: oscillator input divided by 16 (1 MHz max.)
 = 2: oscillator input divided by 8 (500 kHz max.)
 = 3: single-pin RC controlled oscillator ($\div 4$)
 = 4: External Schmitt trigger clock input ($\div 4$)
- Option 4: RESET Input
 = 0: load device to V_{CC}
 = 1: Hi-Z Input
- Option 5: L₇ Driver
 = 0: Standard output
 = 1: Open-drain output
 = 2: High current LED direct segment drive output
 = 3: High current TRI-STATE push-pull output
 = 4: Low-current LED direct segment drive output
 = 5: Low-current TRI-STATE push-pull output
- Option 6: L₆ Driver
 same as Option 5
- Option 7: L₅ Driver
 same as Option 5
- Option 8: L₄ Driver
 same as Option 5
- Option 9: IN₁ Input
 = 0: load device to V_{CC}
 = 1: Hi-Z input
- Option 10: IN₂ Input
 same as Option 9
- Option 11: V_{CC} pin
 = 0: Standard V_{CC}
- Option 12: L₃ Driver
 same as Option 5
- Option 13: L₂ Driver
 same as Option 5
- Option 14: L₁ Driver
 same as Option 5
- Option 15: L₀ Driver
 same as Option 5
- Option 16: SI Input
 same as Option 9
- Option 17: SO Driver
 = 0: standard output
 = 1: open-drain output
 = 2: push-pull output
- Option 18: SK Driver
 same as Option 17
- Option 19: IN₀ Input
 same as Option 9
- Option 20: IN₃ Input
 same as Option 9
- Option 21: G₀ I/O Port
 = 0: very-high current standard output
 = 1: very-high current open-drain output
 = 2: high current standard output
 = 3: high current open-drain output
 = 4: standard LSTTL output (fanout = 1)
 = 5: open-drain LSTTL output (fanout = 1)
- Option 22: G₁ I/O Port
 same as Option 21
- Option 23: G₂ I/O Port
 same as Option 21
- Option 24: G₃ I/O Port
 same as Option 21
- Option 25: D₃ Output
 same as Option 21
- Option 26: D₂ Output
 same as Option 21
- Option 27: D₁ Output
 same as Option 21
- Option 28: D₀ Output
 same as Option 21
- Option 29: L Input Levels
 = 0: standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
 = 1: higher voltage input levels
 ("0" = 1.2V, "1" = 3.6V)
- Option 30: IN Input Levels
 same as Option 29
- Option 31: G Input Levels
 same as Option 29
- Option 32: SI Input Levels
 same as Option 29
- Option 33: RESET Input
 = 0: Schmitt trigger input
 = 1: standard TTL input levels
 = 2: higher voltage input levels
- Option 34: CKO Input Levels
 (CKO = input; Option 2 = 2,3)
 same as Option 29
- Option 35: COP Bonding
 = 0: COP420L (28-pin device)
 = 1: COP421L (24-pin device)
 = 2: 28- and 24-pin versions
 = 3: COP422L (20-pin device)
 = 4: 28- and 20-pin versions
 = 5: 24- and 20-pin versions
 = 5: 28-, 24-, and 20-pin versions
- Option 36: Internal Initialization Logic
 = 0: normal operation
 = 1: no internal initialization logic

Option Table

The following EPROM option information is to be sent to National along with the EPROM.

OPTION DATA	
OPTION 1 VALUE =	<u>0</u> IS: GROUND PIN
OPTION 2 VALUE =	_____ IS: CKO OUTPUT
OPTION 3 VALUE =	_____ IS: CKI INPUT
OPTION 4 VALUE =	_____ IS: RESET INPUT
OPTION 5 VALUE =	_____ IS: L ₇ DRIVER
OPTION 6 VALUE =	_____ IS: L ₆ DRIVER
OPTION 7 VALUE =	_____ IS: L ₅ DRIVER
OPTION 8 VALUE =	_____ IS: L ₄ DRIVER
OPTION 9 VALUE =	_____ IS: IN1 INPUT
OPTION 10 VALUE =	_____ IS: IN2 INPUT
OPTION 11 VALUE =	<u>0</u> IS: VCC PIN
OPTION 12 VALUE =	_____ IS: L ₃ DRIVER
OPTION 13 VALUE =	_____ IS: L ₂ DRIVER
OPTION 14 VALUE =	_____ IS: L ₁ DRIVER
OPTION 15 VALUE =	_____ IS: L ₀ DRIVER
OPTION 16 VALUE =	_____ IS: SI INPUT
OPTION 17 VALUE =	_____ IS: SO DRIVER
OPTION 18 VALUE =	_____ IS: SK DRIVER

OPTION DATA	
OPTION 19 VALUE =	_____ IS: IN ₀ INPUT
OPTION 20 VALUE =	_____ IS: IN ₃ INPUT
OPTION 21 VALUE =	_____ IS: G ₀ I/O PORT
OPTION 22 VALUE =	_____ IS: G ₁ I/O PORT
OPTION 23 VALUE =	_____ IS: G ₂ I/O PORT
OPTION 24 VALUE =	_____ IS: G ₃ I/O PORT
OPTION 25 VALUE =	_____ IS: D ₃ OUTPUT
OPTION 26 VALUE =	_____ IS: D ₂ OUTPUT
OPTION 27 VALUE =	_____ IS: D ₁ OUTPUT
OPTION 28 VALUE =	_____ IS: D ₀ OUTPUT
OPTION 29 VALUE =	_____ IS: L INPUT LEVELS
OPTION 30 VALUE =	_____ IS: IN INPUT LEVELS
OPTION 31 VALUE =	_____ IS: G INPUT LEVELS
OPTION 32 VALUE =	_____ IS: SI INPUT LEVELS
OPTION 33 VALUE =	_____ IS: $\overline{\text{RESET}}$ INPUT
OPTION 34 VALUE =	_____ IS: CKO INPUT LEVELS
OPTION 35 VALUE =	_____ IS: COP BONDING
OPTION 36 VALUE =	_____ IS: INTERNAL INITIALIZATION LOGIC

TEST MODE (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the customer-programmed COP420L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

- a. RAM and Internal Logic Test Mode (SI = 1)
- b. ROM Test Mode (SI = 0)

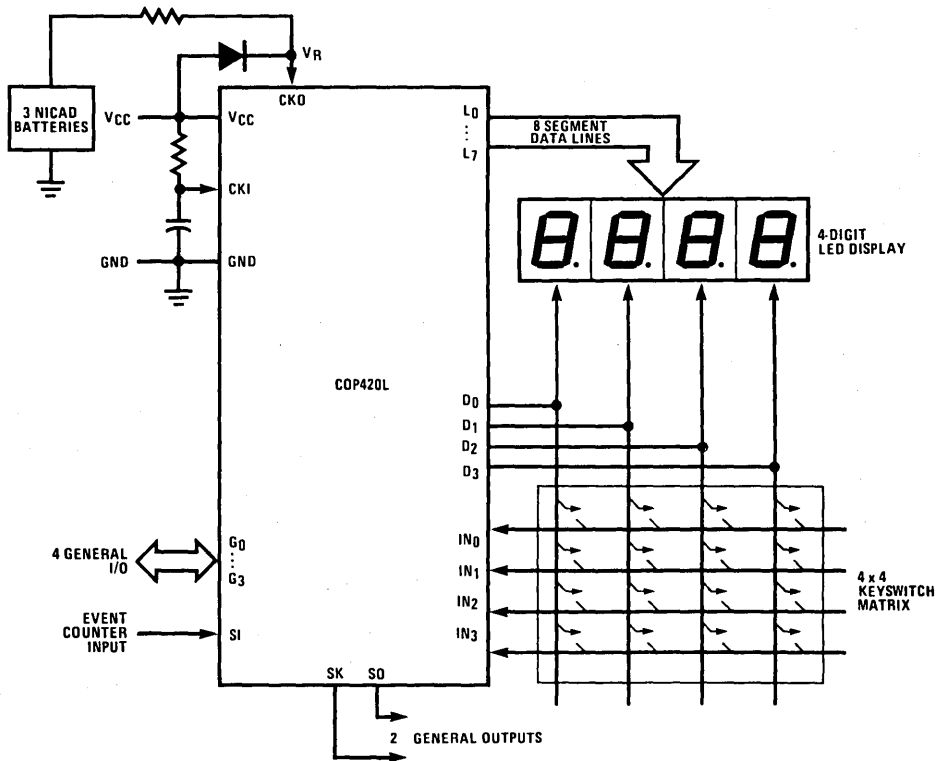
These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATIONS # 1: COP420L General Controller

Figure 9 shows an interconnect diagram for a COP420L used as a general controller. Operation of the system is as follows:

1. The L₇–L₀ outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display.
2. The D₃–D₀ outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN₃–IN₀ inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a general purpose input.
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G₃–G₀) are available for use as required by the user's application.

Typical Applications



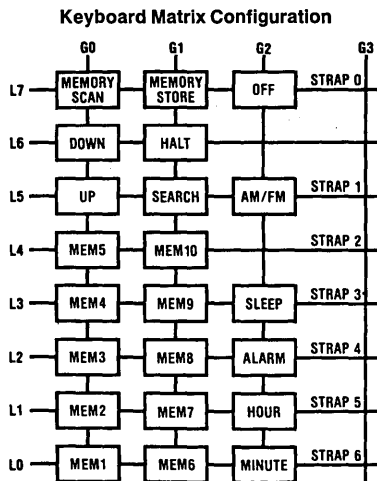
*SO, SI, SK may also be used for Serial I/O

TL/DD/8825-22

FIGURE 9. COP420L Keyboard/Display Interface

APPLICATION #2:

Digitally Tuned Radio Controller and Clock



TL/DD/8825-23

Typical Applications (Continued)

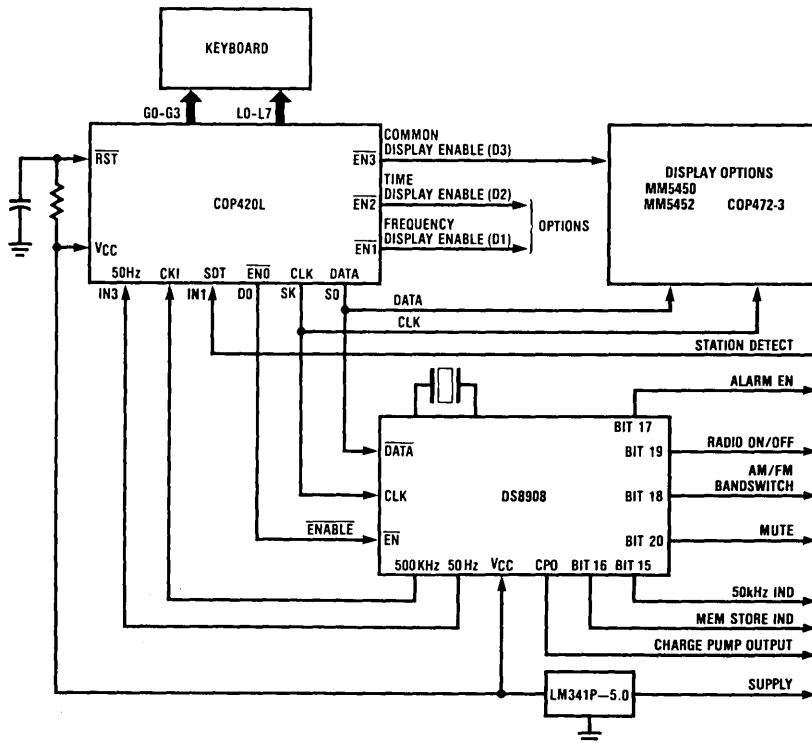


FIGURE 10. Digital Tuning System Block

TL/DD/8825-24

Functional Description

LOGIC I/Os

CKI Input: This input accepts an external 500 kHz signal, divides it by eight and outputs the quotient at the CLK output as the system clock.

RST Input: Schmitt trigger input to clear device upon initialization.

SDT Input: Interrupt input for station detection. The SDT signal is generated by the radio's station detector and used by the COP420L to determine if there is a valid station on the active frequency. The status of the SDT input is only relevant during station searching mode. A high on SDT will temporarily terminate the search mode for eight seconds.

ALM Input: A high on ALM will activate alarm output via slave device at alarm time. A low on the input will disable alarm function.

DATA Output: Push-pull output providing serial data to external devices.

CLK Output: Push-pull output providing system clock at data transmitting time.

50 Hz Input: A normally high input to accept a 50 Hz external time base for real-time calculation.

MOMENTARY KEYS DESCRIPTION

MEM 1–MEM 10: Each memory represents data of a favorite station in a certain band. Depression of one of these

keys will recall the previous stored data and transmit it to the PLL. The PLL will in turn change the radio's receiving frequency as well as the band if necessary. Memory recall keys can also turn on the radio.

UP: This key will manually increment receiving frequency. The first four steps of increment will be for fine tuning a station, after which will be fast slewing meant for manual receive frequency changing.

DOWN: Has the same function as UP key except that frequency is decremented.

MEMORY SCAN: This will start the radio scanning through all ten memories automatically at eight seconds per memory starting from Memory 1. This will also turn on the radio if it was off.

MEMORY STORE: Enables the memory store mode which lasts for three seconds. Depression of any memory key will store the active frequency and band in that memory and disable the store mode. Any function key will also disable the mode to prevent memory data being accidentally destroyed.

HALT: Depression of the HALT key will stop the search and scan functions at current frequency or memory. HALT also turns on the radio during off time and recall frequency display in signal display mode.

SEARCH: Activates station searching in the current band. Search speed is 50 ms per frequency step with wrapping

Functional Description (Continued)

around at end of band. An 8-second stop will take place on reaching a valid station. The HALT key or any function key will terminate the search. Search direction will normally be upwards unless the DOWN key has been depressed prior to the SEARCH key or during the search function in which case search direction will be downwards.

OFF: Turns off the radio or alarm when active.

AM/FM: Radio band switch.

SLEEP: Activates sleep mode, turns on radio on depression and off radio at the end of sleep period. Setting of sleep period is done by depressing the SLEEP and MINUTE key simultaneously.

ALARM: Enables alarm time setting. Depressing the HOUR or MINUTE key and ALARM key simultaneously will set the alarm hour and minute respectively.

HOUR: Sets the hour digits of time-related functions.

MINUTE: Sets the minute digits of time-related functions.

DIODE STRAPS CONNECTIONS

STRAP 0: Controls the on and off of radio. In applications where a toggle type ON/OFF switch is used, momentary OFF key can be omitted; connecting the strap will turn on the radio and vice versa. Must be connected to use momentary OFF key.

STRAP 1, 2: Selects the AM IF options.

STRAP 3: 12/24-hour clock select.

STRAP 4: 3/5 kHz AM step size select.

STRAP 5, 6: FM IF offsets select.

	STRAP 0	STRAP 3	STRAP 4
Connected	Radio ON	12 hour	5 kHz step
Open	Radio OFF	24 hour	3 kHz step

AM/FM IF OPTIONS

AM	STRAP 1	STRAP 2
455 kHz	X	X
460 kHz	X	✓
450 kHz	✓	X
260 kHz	✓	✓
FM	STRAP 5	STRAP 6
10.7 MHz	X	X
10.75 MHz	X	✓
10.65 MHz	✓	X
10.8 MHz	✓	✓

X = No connection.

✓ = Diode inserted.

INDIRECT FEATURES AND OPTIONS

As indicated in *Figure 10*, there are a few options and indirect features provided via the help of a slave device, namely the Phase Lock Loop, DS8906N.

DISPLAY OPTIONS

As mentioned above, the COP420L-HSB is MICROWIRE compatible. Internal circuitry enables it to directly interface with all of National's serial input MICROWIRE compatible display drivers whether they are of a direct drive or multiplex drive format. On *Figure 10* is a list of drivers available for the system. EN1 and EN2 are optional enable outputs meant for a dual display system in which EN3 will not be used. By dual display, it means that one display will be constantly showing time information and the other showing frequency information. Whereas in conventional single display systems, the display shows both time and frequency information in a time-sharing method. The National system provides a time-prioritized display-sharing method. That is, whenever a tuning function is completed, the frequency information will stay on the display for eight seconds then time display will take over. This is achieved by using EN3 for the driver's enable logic.

CONTROL OUTPUTS

Six open collector outputs controlled by the COP420L are provided from DS8906N, the phase lock loop for controlling radio switching circuits.

Radio ON/OFF: A high from this output indicates that the radio should be switched on and vice versa.

AM/FM: Output for controlling the AM/FM bandswitch. A high level output indicates FM and a low indicates the AM band.

MUTE: For muting the audio output when performing any frequency related function. The output will go high prior to the frequency change except when doing fine tuning.

ALARM ENABLE: Active high output for turning on the alarm circuit at alarm time.

50 kHz IND: For driving the 50 kHz indicator in FM band or the LSB in a 5-digit display. Output is active high.

MEM STORE IND: For driving the memory store mode indicator. Output is active high.

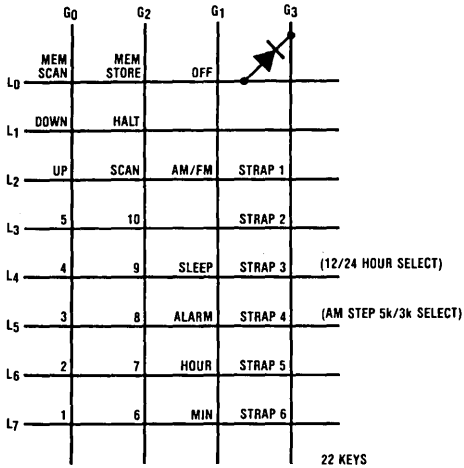
TYPICAL IMPLEMENTATION ALTERNATIVES

A full keyboard or any portion of it can be implemented with various applications for features/functions vs. cost/size.

Figure 11 shows two keyboard configurations with 22-key and 11-key keyboards for a desk-top/tuner system or auto-radio system, respectively.

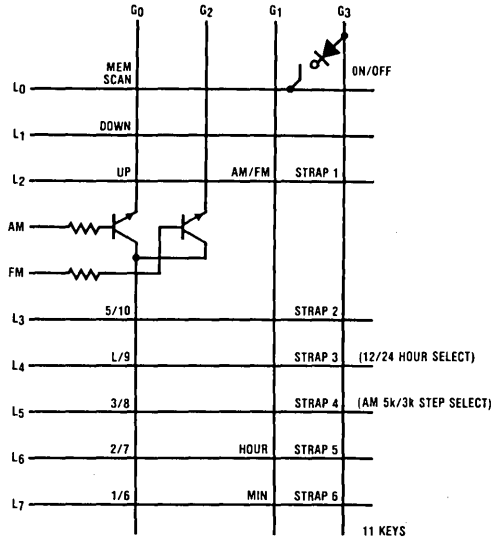
Functional Description (Continued)

Desk Top DTR Keyboard



TL/DD/8825-25

Car DTR Keyboard



TL/DD/8825-26

FIGURE 11



COP424C, COP425C, COP426C, COP324C, COP325C, COP326C and COP444C, COP445C, COP344C, COP345C Single-Chip 1k and 2k CMOS Microcontrollers

General Description

The COP424C, COP425C, COP426C, COP444C and COP445C fully static, Single-Chip CMOS Microcontrollers are members of the COPSTM family, fabricated using double-poly, silicon gate microCMOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP424C and COP444C are 28 pin chips. The COP425C and COP445C are 24-pin versions (4 inputs removed) and COP426C is 20-pin version with 15 I/O lines. Standard test procedures and reliable high-density techniques provide the medium to large volume customers with a customized microcontroller at a low end-product cost. These microcontrollers are appropriate choices in many demanding control environments especially those with human interface.

The COP424C is an improved product which replaces the COP420C.

Features

- Lowest power dissipation (50 μ W typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4 μ s instruction time, plus software selectable clocks
- 2k x 8 ROM, 128 x 4 RAM (COP444C/COP445C)
- 1k x 8 ROM, 64 x 4 RAM (COP424C/COP425C/COP426C)
- 23 I/O lines (COP444C and COP424C)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (2.4V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS output compatible
- Microbus™ compatible
- Software/hardware compatible with COP400 family
- Extended temperature range devices COP324C/COP325C/COP326C and COP344C/COP345C (-40°C to +85°C)
- Military devices (-55°C to +125°C) to be available

Block Diagram

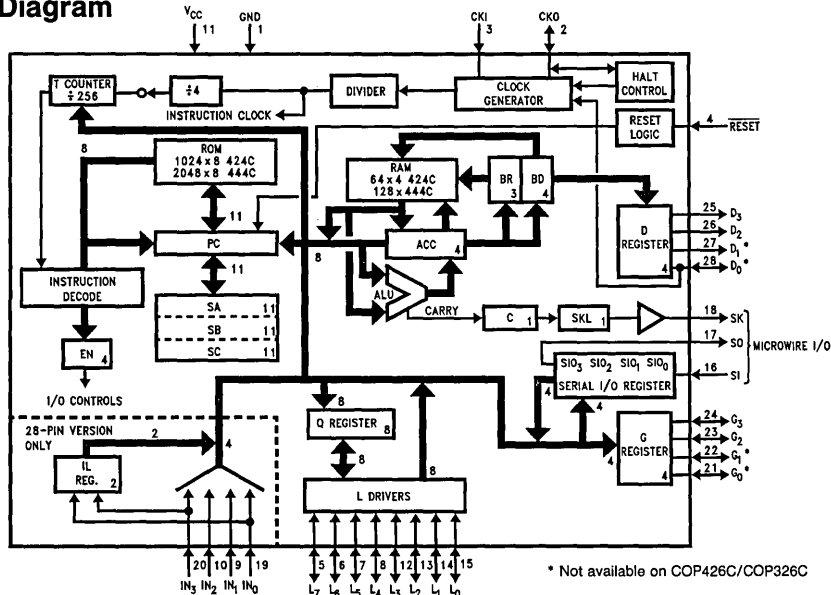


FIGURE 1

COP424C/COP425C/COP426C and COP444C/COP445C**Absolute Maximum Ratings**

Supply Voltage (V_{CC})	6V
Voltage at any Pin	-0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 0°C ≤ T_A ≤ 70°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple (Notes 4, 5)	Peak to Peak		0.1 V_{CC}	V
Supply Current (Note 1)	$V_{CC}=2.4V$, $t_c=64 \mu s$ $V_{CC}=5.0V$, $t_c=16 \mu s$ $V_{CC}=5.0V$, $t_c=4 \mu s$ (t_c is instruction cycle time)		120 700 3000	μA μA μA
HALT Mode Current (Note 2)	$V_{CC}=5.0V$, $F_{IN}=0$ kHz $V_{CC}=2.4V$, $F_{IN}=0$ kHz		40 12	μA μA
Input Voltage Levels RESET, CKI, D ₀ (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	$V_{CC}=4.5V$, $V_{IN}=0$	-30	-330	μA
Hi-Z Input Leakage		-1	+1	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels				
LSTTL Operation	Standard Outputs $V_{CC}=5.0V \pm 10\%$			
Logic High	$I_{OH} = -100 \mu A$	2.7		V
Logic Low	$I_{OL} = 400 \mu A$		0.4	V
CMOS Operation				
Logic High	$I_{OH} = -10 \mu A$	$V_{CC}-0.2$		V
Logic Low	$I_{OL} = 10 \mu A$		0.2	V
Output Current Levels (except CKO) Sink (Note 6)	$V_{CC}=4.5V$, $V_{OUT}=V_{CC}$ $V_{CC}=2.4V$, $V_{OUT}=V_{CC}$	1.2 0.2		mA mA
Source (Standard Option)	$V_{CC}=4.5V$, $V_{OUT}=0V$ $V_{CC}=2.4V$, $V_{OUT}=0V$	-0.5 -0.1		mA mA
Source (Low Current Option)	$V_{CC}=4.5V$, $V_{OUT}=0V$ $V_{CC}=2.4V$, $V_{OUT}=0V$	-30 -6	-330 -80	μA μA
CKO Current Levels (As Clock Out)				
Sink		0.3		mA
÷4	$V_{CC}=4.5V$, CKI = V_{CC} , $V_{OUT}=V_{CC}$	0.6		mA
÷8		1.2		mA
÷16				
Source		-0.3		mA
÷4	$V_{CC}=4.5V$, CKI = 0V, $V_{OUT}=0V$	-0.6		mA
÷8		-1.2		mA
÷16				
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	$V_{CC}=4.5V$, $V_{IN}=0.2V_{CC}$		0.7	mA
To Halt	$V_{CC}=4.5V$, $V_{IN}=0.7V_{CC}$		1.6	mA
TRI-STATE or Open Drain Leakage Current		-2.5	+2.5	μA

COP324C/COP325C/COP326C and COP344C/COP345C

Absolute Maximum Ratings

Supply Voltage	6V
Voltage at any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	-40°C to +85°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		3.0	5.5	V
Power Supply Ripple (Notes 4, 5)	Peak to Peak		0.1 V_{CC}	V
Supply Current (Note 1)	$V_{CC}=3.0V, t_c=64 \mu s$ $V_{CC}=5.0V, t_c=16 \mu s$ $V_{CC}=5.0V, t_c=4 \mu s$ (t_c is instruction cycle time)		180 800 3600	μA μA μA
HALT Mode Current (Note 2)	$V_{CC}=5.0V, F_{IN}=0 \text{ kHz}$ $V_{CC}=3.0V, F_{IN}=0 \text{ kHz}$		60 30	μA μA
Input Voltage Levels RESET, CKI, D _O (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	$V_{CC}=4.5V, V_{IN}=0$	-30	-440	μA
Hi-Z Input Leakage		-2	+2	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels				
LSTTL Operation	Standard Outputs $V_{CC}=5.0V \pm 10\%$			
Logic High	$I_{OH} = -100 \mu A$	2.7		V
Logic Low	$I_{OL} = 400 \mu A$		0.4	V
CMOS Operation				
Logic High	$I_{OH} = -10 \mu A$	$V_{CC}-0.2$		V
Logic Low	$I_{OL} = 10 \mu A$		0.2	V
Output Current Levels (except CKO) Sink (Note 6)	$V_{CC}=4.5V, V_{OUT}=V_{CC}$ $V_{CC}=3.0V, V_{OUT}=V_{CC}$	1.2 0.2		μA μA
Source (Standard Option)	$V_{CC}=4.5V, V_{OUT}=0V$ $V_{CC}=3.0V, V_{OUT}=0V$	-0.5 -0.1		μA μA
Source (Low Current Option)	$V_{CC}=4.5V, V_{OUT}=0V$ $V_{CC}=3.0V, V_{OUT}=0V$	-30 -8	-440 -200	μA μA
CKO Current Levels (As Clock Out)				
Sink		0.3		μA
÷4	$V_{CC}=4.5V, CKI=V_{CC}, V_{OUT}=V_{CC}$	0.6		μA
÷8		1.2		μA
÷16				
Source		-0.3		μA
÷4	$V_{CC}=4.5V, CKI=0V, V_{OUT}=0V$	-0.6		μA
÷8		-1.2		μA
÷16				
Allowable Sink/Source Current per Pin (Note 6)			5	μA
Allowable Loading on CKO (as HALT)			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	$V_{CC}=4.5V, V_{IN}=0.2V_{CC}$		0.9	μA
To Halt	$V_{CC}=4.5V, V_{IN}=0.7V_{CC}$		2.1	μA
TRI-STATE or Open Drain Leakage Current		-5	+5	μA

COP424C/COP425C/COP426C and COP444C/COP445C

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (tc)	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$	4 16	DC DC	μs μs
Operating CKI Frequency	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$	DC	1.0	MHz
$\div 4$ mode			2.0	MHz
$\div 8$ mode			4.0	MHz
$\div 16$ mode		250	kHz	
$\div 4$ mode		500	kHz	
$\div 8$ mode	1.0	MHz		
$\div 16$ mode				
Duty Cycle (Note 4)	$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 4$ MHz External Clock		60	ns
Fall Time (Note 4)	$f_1 = 4$ MHz External Clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)	$R = 30\text{k} \pm 5\%$, $V_{CC} = 5\text{V}$ $C = 82$ pF $\pm 5\%$ ($\div 4$ Mode)	5	11	μs
Inputs: (See Figure 3)				
t_{SETUP}	G Inputs } $V_{CC} \geq 4.5\text{V}$ SI Input } All Others }	$t_c/4 + 7$		μs
			0.3	μs
			1.7	μs
t_{HOLD}	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$	0.25 1.0		μs μs
Output Propagation Delay	$V_{\text{OUT}} = 1.5\text{V}$, $C_L = 100$ pF, $R_L = 5\text{k}$ $V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$		1.0 4.0	μs μs
Microbus Timing	$C_L = 50$ pF, $V_{CC} = 5\text{V} \pm 5\%$			
Read Operation (Figure 4)				
Chip Select Stable before $\overline{\text{RD}}$ — t_{CSR}		65		ns
Chip Select Hold Time for $\overline{\text{RD}}$ — t_{RCS}		20		ns
$\overline{\text{RD}}$ Pulse Width — t_{RR}		400		ns
Data Delay from $\overline{\text{RD}}$ — t_{RD}			375	ns
$\overline{\text{RD}}$ to Data Floating — t_{DF} (Note 4)			250	ns
Write Operation (Figure 5)				
Chip Select Stable before $\overline{\text{WR}}$ — t_{CSW}		65		ns
Chip Select Hold Time for $\overline{\text{WR}}$ — t_{WCS}		20		ns
$\overline{\text{WR}}$ Pulse Width — t_{WW}		400		ns
Data Set-Up Time for $\overline{\text{WR}}$ — t_{DW}		320		ns
Data Hold Time for $\overline{\text{WR}}$ — t_{WD}		100		ns
INTR Transition Time from $\overline{\text{WR}}$ — t_{WI}			700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 17.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC} , L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than $0.2V_{CC}$ when part is running in order to prevent entering test mode.

COP324C/COP325C/COP326C and COP344C/COP345C

AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units	
Instruction Cycle Time (t_c)	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 3.0\text{V}$	4 16	DC DC	μs μs	
Operating CKI Frequency	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 3.0\text{V}$	DC	1.0	MHz	
÷ 4 mode			2.0	MHz	
÷ 8 mode			4.0	MHz	
÷ 16 mode		DC	250	kHz	
÷ 4 mode		DC	500	kHz	
÷ 8 mode		DC	1.0	MHz	
Duty Cycle (Note 4)	$f_1 = 4\text{ MHz}$	40	60	%	
Rise Time (Note 4)	$f_1 = 4\text{ MHz external clock}$		60	ns	
Fall Time (Note 4)	$f_1 = 4\text{ MHz external clock}$		40	ns	
Instruction Cycle Time RC Oscillator (Note 4)	$R = 30\text{k} \pm 5\%$, $V_{CC} = 5\text{V}$ $C = 82\text{ pF} \pm 5\%$ ($\div 4\text{ Mode}$)	5	11	μs	
Inputs: (See Figure 3)	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 3.0\text{V}$	$t_c/4 + .7$ 0.3 1.7 0.25 1.0		μs μs μs μs μs	
t_{SETUP}					G Inputs
					SI Inputs
					All Others
t_{HOLD}					
Output Propagation Delay	$V_{\text{OUT}} = 1.5\text{V}$, $C_L = 100\text{ pF}$, $R_L = 5\text{k}$ $V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 3.0\text{V}$		1.0 4.0	μs μs	
Microbus Timing	$C_L = 50\text{ pF}$, $V_{CC} = 5\text{V} \pm 5\%$				
Read Operation (Figure 4)					
Chip Select Stable before $\overline{\text{RD}}$ – t_{CSR}		65		ns	
Chip Select Hold Time for $\overline{\text{RD}}$ – t_{RCS}		20		ns	
$\overline{\text{RD}}$ Pulse Width – t_{RR}		400		ns	
Data Delay from $\overline{\text{RD}}$ – t_{RD}			375	ns	
$\overline{\text{RD}}$ to Data Floating – t_{DF} (Note 4)			250	ns	
Write Operation (Figure 5)					
Chip Select Stable before $\overline{\text{WR}}$ – t_{CSW}		65		ns	
Chip Select Hold Time for $\overline{\text{WR}}$ – t_{WCS}		20		ns	
$\overline{\text{WR}}$ Pulse Width – t_{WW}		400		ns	
Data Set-Up Time for $\overline{\text{WR}}$ – t_{DW}		320		ns	
Data Hold Time for $\overline{\text{WR}}$ – t_{WD}		100		ns	
INTR Transition Time from $\overline{\text{WR}}$ – t_{WI}			700	ns	

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 17.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC} , L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

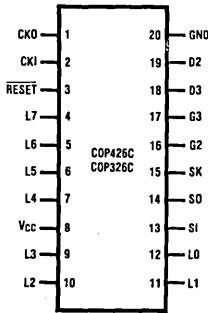
Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than $0.2V_{CC}$ when part is running in order to prevent entering test mode.

Connection Diagrams

DIP and S.O. Wide

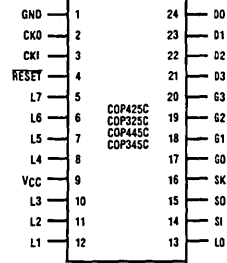


Top View

Order Number COP326C-XXX/D or COP426C-XXX/D
See NS Hermetic Package D20A (Prototype Package Only)
Order Number COP326C-XXX/N or COP426C-XXX/N
See NS Molded Package N20A
Order Number COP326C-XXX/WM or COP426C-XXX/WM
See NS Surface Mount Package M20B

TL/DD/5259-16

DIP and S.O. Wide

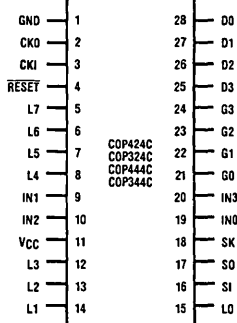


Top View

Order Number COP325C-XXX/D, COP445C-XXX/D, COP425C-XXX/D or COP345C-XXX/D
See NS Hermetic Package D24C (Prototype Package Only)
Order Number COP325C-XXX/N, COP345C-XXX/N, COP425C-XXX/N or COP445C-XXX/N
See NS Molded Package N24A
Order Number COP325C-XXX/WM, COP345C-XXX/WM, COP425C-XXX/WM or COP445C-XXX/WM
See NS Surface Mount Package M24B

TL/DD/5259-2

Dual-In-Line Package



Top View

Order Number COP324C-XXX/D, COPC324-XXX/WM, COP344C-XXX/D, COP424C-XXX/D, COPC424-XXX/WM or COP444C-XXX/D
See NS Hermetic Package D28C (Prototype Package Only)
Order Number COP324C-XXX/N, COP344C-XXX/N, COPC344-XXX/WM, COP424C-XXX/N, COP444C-XXX/N or COPC444-XXX/WM
See NS Molded Package N28B

TL/DD/5259-3

FIGURE 2

Pin	Description
L7-L0	8-bit bidirectional port with TRI-STATE
G3-G0	4-bit bidirectional I/O port
D3-D0	4-bit output port
IN3-IN0	4-bit input port (28-pin package only)
SI	Serial input or counter input
SO	Serial or general purpose output

Pin	Description
SK	Logic controlled clock output
CKI	Chip oscillator input
CKO	Oscillator output, HALT I/O port or general purpose input
RESET	Reset input
VCC	Most positive power supply
GND	Ground

Functional Description

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

For ease of reading only the COP424C/425C/COP426C/444C/445C are referenced; however, all such references apply equally to COP324C/325C/COP326C/344C/345C.

PROGRAM MEMORY

Program Memory consists of ROM, 1024 bytes for the COP424C/425C/426C and 2048 bytes for the COP444C/445C. These bytes of ROM may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by a 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

Data memory consists of a 512-bit RAM for the COP444C/445C, organized as 8 data registers of 16×4 -bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register.

Data memory consists of a 256-bit RAM for the COP424C/425C/426C, organized as 4 data registers of 16×4 -bits digits. The B register is 6 bits long. Upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions.

The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, to input 4 bits of L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. When the T counter overflows, an overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 10a*.

Four general-purpose inputs, IN3-IN0, are provided. IN1, IN2 and IN3 may be selected, by a mask-programmable option as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in Microbus application.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. In the dual clock mode, D0 latch controls the clock selection (see dual oscillator below).

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G0 may be mask-programmed as an output for Microbus applications.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the Microbus option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M. As explained above, the Microbus option allows L I/O port data to be latched into the Q register.

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

- 0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon

each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.

- 1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
- 2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.

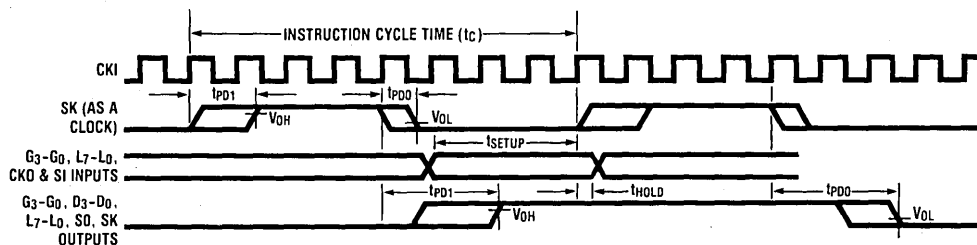


FIGURE 3. Input/Output Timing Diagrams (divide by 8 mode)

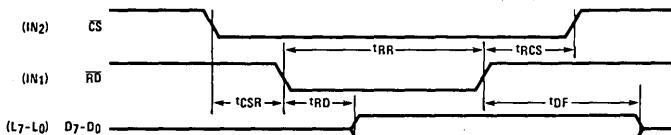


FIGURE 4. Microbus Read Operation Timing

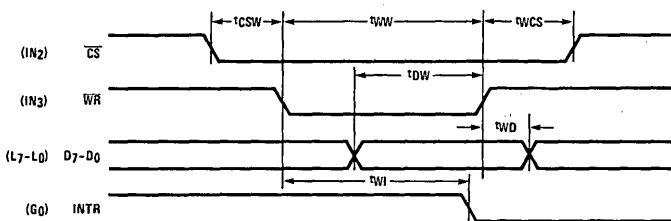


FIGURE 5. Microbus Write Operation Timing

Functional Description (Continued)

3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.
- b. An interrupt will be recognized only on the following conditions:
 1. EN1 has been set.
 2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN₁ input.
 3. A currently executing instruction has been completed.
 4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).
- c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

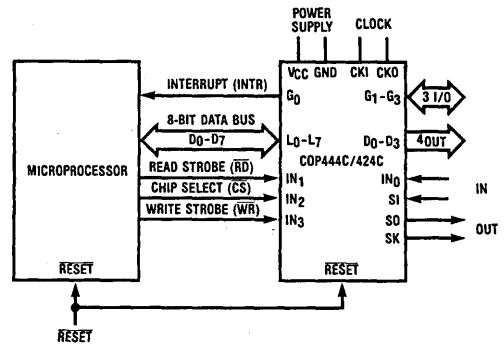
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

MICROBUS INTERFACE

The COP444C/424C has an option which allows it to be used as a peripheral microprocessor, inputting and outputting data from and to a host microprocessor (μP). IN₁, IN₂ and IN₃ general purpose inputs become Microbus compatible read-strobe, chip-select, and write-strobe lines, respectively. IN₁ becomes \overline{RD} — a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the uP. IN₂ becomes \overline{CS} — a logic "0" on this line selects the COP444C/424C as the uP peripheral device by enabling the operation of the \overline{RD} and \overline{WR} lines and allows for the selection of one of several peripheral components. IN₃ becomes \overline{WR} — a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP444C/424C. G₀ becomes INTR a "ready" output, reset by a write pulse from the uP on the \overline{WR} line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP444C/424C.

This option has been designed for compatibility with National's Microbus — a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See Microbus National Publication.) The functioning and timing relationships between the signal lines affected by this option are as specified for the Microbus interface, and are given in the AC electrical characteristics and shown in the timing diagrams (Figures 4 and 5). Connection of the COP444C/424C to the Microbus is shown in Figure 6.



TL/DD/5259-7

FIGURE 6. Microbus Option Interconnect

TABLE I. Enable Register Modes — Bits EN0 and EN3

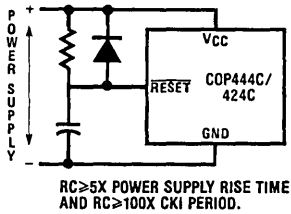
EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 7* must be connected to the **RESET** pin (the conditions in *Figure 7* must be met). The **RESET** pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC} . Initialization will occur whenever a logic "0" is applied to the **RESET** input, providing it stays low for at least three instruction cycle times.

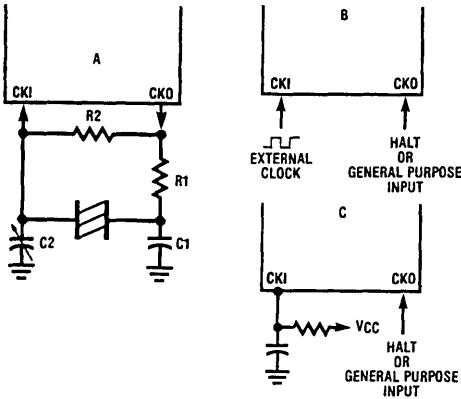
Note: If CKI clock is less than 32 kHz, the internal reset logic (option #29 = 1) MUST be disabled and the external RC circuit must be used.



TL/DD/5259-8

FIGURE 7. Power-Up Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/5259-9

Crystal or Resonator

Crystal Value	Component Values			
	R1	R2	C1(pF)	C2(pF)
32 kHz	220k	20M	30	6-36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6-36
4.0 MHz	1k	1M	30	6-36

RC Controlled Oscillator ($\pm 5\% R, \pm 5\% C$)

R	C	Cycle Time	V_{CC}
30k	82 pF	5-11 μ s	$\geq 4.5V$
60k	100 pF	12-24 μ s	2.4-4.5V

Note: $15k \leq R \leq 150k$
 $50 \text{ pF} \leq C \leq 150 \text{ pF}$

FIGURE 8. Oscillator Component Values

TIMER

The timer can be operated as a time-base counter.

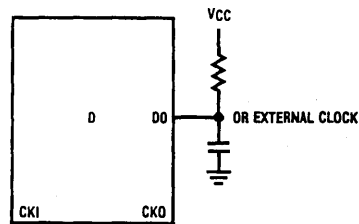
The instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The pre-scaler is cleared during execution of a CAMT instruction and on reset.

For example, using a 4 MHz crystal with a divide-by-16 option, the instruction cycle frequency of 250 kHz increments the 10-bit timer every 4 μ s. By presetting the counter and detecting overflow, accurate timeouts between 16 μ s (4 counts) and 4.096 ms (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

HALT MODE

The COP444C/445C/424C/425C/426C is a FULLY STAT-IC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by the HALT instruction or by forcing CKO high when it is mask-programmed as an HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The chip may be awakened by one of two different methods:

- Continue function: by forcing CKO low, if it mask-programmed as an HALT I/O port, the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- Restart: by forcing the **RESET** pin low (see Initialization).



Functional Description (Continued)

The HALT mode is the minimum power dissipation state.

Note: If the user has selected dual-clock with D0 as external oscillator (option 30=2) AND the COP444C/424C is running with the D0 clock, the HALT mode — either hardware or software — will NOT be entered. Thus, the user should switch to the CKI clock to HALT. Alternatively, the user may stop the D0 clock to minimize power.

CKO PIN OPTIONS

a. Two-pin oscillator — (Crystal). See *Figure 9A*.

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. The HALT mode may be entered by program control (HALT instruction) which forces CKO high, thus inhibiting the crystal network. The circuit can be awakened only by forcing the RESET pin to a logic "0" (restart).

b. One-pin oscillator — (RC or external). See *Figure 9B*.

If a one-pin oscillator system is chosen, two options are available for CKO:

- CKO can be selected as the HALT I/O port. In that case, it is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and CKO output will stay high to keep the chip stopped if the external driver returns to high impedance state. By forcing a low level to CKO, the chip will continue and CKO will stay low.
- As another option, CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction.

OSCILLATOR OPTIONS

There are four basic clock oscillator configurations available as shown by *Figure 8*.

- Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency optionally divided by 4, 8 or 16.
- External Oscillator. The external frequency is optionally divided by 4, 8 or 16 to give the instruction cycle time. CKO is the HALT I/O port or a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port or a general purpose input.

d. Dual oscillator. By selecting the dual clock option, pin D0 is now a single pin oscillator input. Two configurations are available: RC controlled Schmitt trigger oscillator or external oscillator.

The user may software select between the D0 oscillator (in that case, the instruction cycle time equals the D0 oscillation frequency divided by 4) by setting the D0 latch high or the CKI (CKO) oscillator by resetting D0 latch low. Note that even in dual clock mode, the counter, if mask-programmed as a time-base counter, is always connected to the CKI oscillator.

For example, the user may connect up to a 1 MHz RC circuit to D0 for faster processing and a 32 kHz watch crystal to CKI and CKO for minimum current drain and time keeping.

Note: CTMA instruction is not allowed when chip is running from D0 clock.

Figures 10A and 10B show the clock and timer diagrams with and without Dual clock.

COP445C AND COP425C 24-PIN PACKAGE OPTION

If the COP444C/424C is bonded in a 24-pin package, it becomes the COP445C/425C, illustrated in *Figure 2*, Connection diagrams. Note that the COP445C/425C does not contain the four general purpose IN inputs (IN3–IN0). Use of this option precludes, of course, use of the IN options, interrupt feature, external event counter feature, and the Microbus option which uses IN1–IN3. All other options are available for the COP445C/425C.

Note: If user selects the 24-pin package, options 9, 10, 19 and 20 must be selected as a "0" (load to V_{CC} on the IN inputs). See option list.

COP426C 20-PIN PACKAGE OPTION

If the COP425C is bonded as 20-pin device it becomes the COP426C. Note that the COP426C contains all the COP425C pins except D₀, D₁, G₀, and G₁.

Block Diagram (Continued)

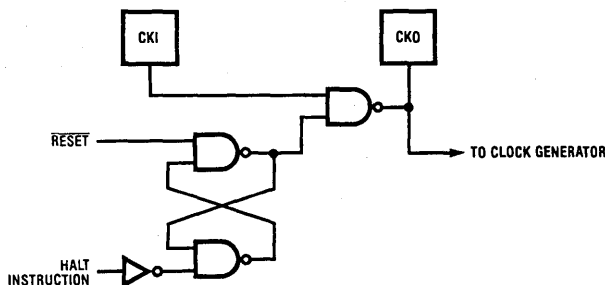
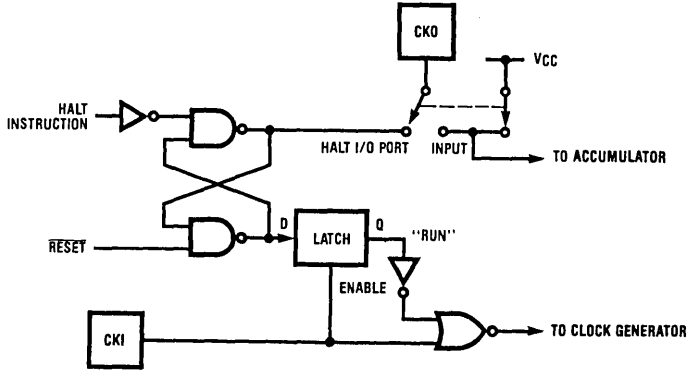


FIGURE 9A. Halt Mode — Two-Pin Oscillator

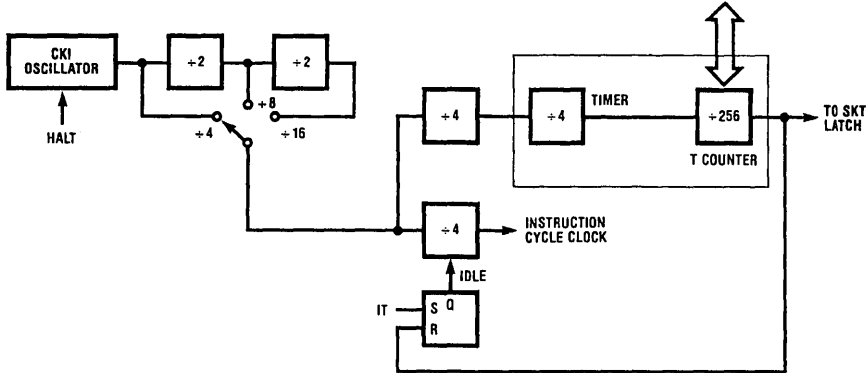
TL/DD/5259-10

Block Diagram (Continued)



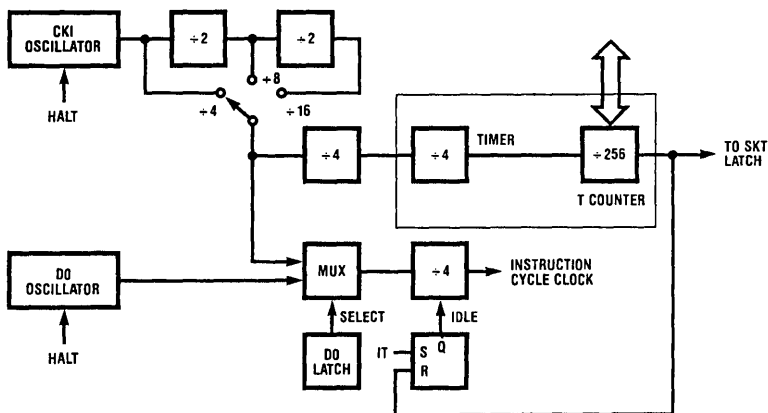
TL/DD/5259-11

FIGURE 9B. Halt Mode — One-Pin Oscillator



TL/DD/5259-12

FIGURE 10A. Clock and Timer without Dual-Clock



TL/DD/5259-13

FIGURE 10B. Clock and Timer with Dual-Clock

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

TABLE II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit accumulator
B	7-bit RAM address register (6-bit for COP424C)
Br	Upper 3 bits of B (register address) (2-bit for COP424C)
Bd	Lower 4 bits of B (digit address)
C	1-bit carry register
D	4-bit data output port
EN	4-bit enable register
G	4-bit general purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA,SB,SC	11-bit 3-level subroutine stack
SIO	4-bit shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

Instruction Operand Symbols

d	4-bit operand field, 0–15 binary (RAM digit select)
r	3(2)-bit operand field, 0–7(3) binary (RAM register select)
a	11-bit operand field, 0–2047 (1023)
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x

Operational Symbols

+	Plus
–	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
\bar{A}	One's complement of A
⊕	Exclusive-or
:	Range of values

TABLE III. COP444C/445C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
ADT		4A	<u>0100</u> <u>1010</u>	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5–	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add Immediate. Skip on Carry ($y \neq 0$)
CASC		10	<u>0001</u> <u>0000</u>	$\bar{A} + \text{RAM}(B) + C \rightarrow A$ Carry $\rightarrow C$	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	Ones complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER CONTROL INSTRUCTIONS						
JID		FF	<u>1111</u> <u>1111</u>	ROM(PC _{10:8} ,A,M) → PC _{7:0}	None	Jump Indirect (Notes 1, 3)
JMP	a	6-- --	<u>0110</u> <u>0</u> <u>a_{10:8}</u> <u>a_{7:0}</u>	a → PC	None	Jump
JP	a	--	<u>1</u> <u>a_{6:0}</u> (pages 2,3 only) or <u>11</u> <u>a_{5:0}</u> (all other pages)	a → PC _{6:0} a → PC _{5:0}	None	Jump within Page (Note 4)
JSRP	a	--	<u>10</u> <u>a_{5:0}</u>	PC+1 → SA → SB → SC 00010 → PC _{10:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 5)
JSR	a	6-- --	<u>0110</u> <u>1</u> <u>a_{10:8}</u> <u>a_{7:0}</u>	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	<u>0100</u> <u>1000</u>	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	<u>0100</u> <u>1001</u>	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	<u>0011</u> <u>0011</u> <u>0011</u> <u>1000</u>		None	HALT Processor
IT		33 39	<u>0011</u> <u>0011</u> <u>0011</u> <u>1001</u>		None	IDLE till Timer Overflows then Continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33 3F	<u>0011</u> <u>0011</u> <u>0011</u> <u>1111</u>	A → T _{7:4} RAM(B) → T _{3:0}	None	Copy A, RAM to T
CTMA		33 2F	<u>0011</u> <u>0011</u> <u>0010</u> <u>1111</u>	T _{7:4} → RAM(B) T _{3:0} → A	None	Copy T to RAM, A (Note 9)
CAMQ		33 3C	<u>0011</u> <u>0011</u> <u>0011</u> <u>1100</u>	A → Q _{7:4} RAM(B) → Q _{3:0}	None	Copy A, RAM to Q
CQMA		33 2C	<u>0011</u> <u>0011</u> <u>0010</u> <u>1100</u>	Q _{7:4} → RAM(B) Q _{3:0} → A	None	Copy Q to RAM, A
LD	r	-5	<u>00</u> <u>r</u> <u>0101</u> (r=0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 --	<u>0010</u> <u>0011</u> <u>0</u> <u>r</u> <u>d</u>	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	<u>1011</u> <u>1111</u>	ROM(PC _{10:8} ,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	<u>0100</u> <u>1100</u> <u>0100</u> <u>0101</u> <u>0100</u> <u>0010</u> <u>0100</u> <u>0011</u>	0 → RAM(B) ₀ 0 → RAM(B) ₁ 0 → RAM(B) ₂ 0 → RAM(B) ₃	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	<u>0100</u> <u>1101</u> <u>0100</u> <u>0111</u> <u>0100</u> <u>0110</u> <u>0100</u> <u>1011</u>	1 → RAM(B) ₀ 1 → RAM(B) ₁ 1 → RAM(B) ₂ 1 → RAM(B) ₃	None	Set RAM Bit

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate 1 and Increment Bd
X	r	-6	00 r 0110 (r=0:3)	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 0011 1 r d	RAM(r,d) ↔ A	None	Exchange A with RAM Pointed to Directly by r,d
XDS	r	-7	00 r 0111 (r=0:3)	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	-4	00 r 0100 (r=0:3)	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	--	00 r (d-1) (r=0:3; d=0,9:15) or 0011 0011 1 r d (any r, any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A ↔ Br	None	Exchange A with Br (Note 8)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C="1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte	G ₀ = 0	Skip if G Bit is Zero
	0	01	0000 0001	} 2nd byte	G ₁ = 0	
	1	11	0001 0001		G ₂ = 0	
	2	03	0000 0011		G ₃ = 0	
	3	13	0001 0011			
SKMBZ		01	0000 0001		RAM(B) ₀ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001		RAM(B) ₁ = 0	
	2	03	0000 0011		RAM(B) ₂ = 0	
	3	13	0001 0011		RAM(B) ₃ = 0	
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
ININ		33	0011 0011	IN → A	None	Input IN Inputs to A (Note 2)
		28	0010 1000			
INIL		33	0011 0011	IL ₃ , CKO, "0", IL ₀ → A	None	Input IL Latches to A (Note 3)
		29	0010 1001			
INL		33	0011 0011	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM,A
		2E	0010 1110			
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OGI	y	33	0011 0011	y → G	None	Output to G Ports Immediate
		5-	0101 y			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin packages since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data minus 7, e.g., to load the lower four bits of B(Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: For 2K ROM devices, A ↔ Br (0 → A3). For 1K ROM devices, A ↔ Br (0,0 → A3, A2).

Note 9: Do not use CTMA instruction when dual-clock option is selected and part is running from D₀ clocks.

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10:PC8,A,M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC+1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC(7:4), RAM(B) → PC(3:0), leaving PC(10), PC(9) and PC(8) unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10:8,A,M. PC10,PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT ; load T counter
SKT  ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKO and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively,

and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A0 is input into A1. IL latches are cleared on reset. IL latches are not available on the COP445C/425C, and COP426C.

INSTRUCTION SET NOTES

- The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.
- The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Note: The COP424C/425C/426C needs only 10 bits to address its ROM. Therefore, the eleventh bit (P10) is ignored.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal or resonator generated clock input will draw additional current. An R/C oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate operating current drain.

$$I_{CO} = I_Q + V \times 40 \times F_i + V \times 1400 \times F_i / D_V$$

where I_{CO} = chip operating current drain in microamps
 quiescent leakage current (from curve)
 CKI frequency in MegaHertz
 chip V_{CC} in volts
 divide by option selected

For example at 5 volts V_{CC} and 400 kHz (divide by 4)

$$I_{CO} = 20 + 5 \times 40 \times 0.4 + 5 \times 1400 \times 0.4 / 4$$

$$I_{CO} = 20 + 80 + 700 = 800 \mu A$$

At 2.4 volts V_{CC} and 30 kHz (divide by 4)

$$I_{CO} = 6 + 2.4 \times 40 \times 0.03 + 2.4 \times 1400 \times 0.03 / 4$$

$$I_{CO} = 6 + 2.88 + 25.2 = 34.08 \mu A$$

Power Dissipation (Continued)

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$I_{ci} = I_Q + V \times 40 \times F_i$$

For example, at 5 volts V_{CC} and 400 kHz

$$I_{ci} = 20 + 5 \times 40 \times 0.4 = 100 \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$I_{ta} = I_{CO} \times \frac{T_o}{T_o + T_i} + I_{ci} \times \frac{T_i}{T_o + T_i}$$

where: I_{ta} = total average current

I_{CO} = operating current

I_{ci} = idle current

T_o = operating time

T_i = idle time

I/O OPTIONS

Outputs have the following optional configurations, illustrated in Figure 11:

- a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC} , compatible with CMOS and LSTTL.
- b. Low Current — This is the same configuration as a. above except that the sourcing current is much less.

- c. Open Drain — An N-channel device to ground only, allowing external pull-up as required by the user's application.
- d. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control.
- e. Low-Current TRI-STATE L Output — This is the same as d. above except that the sourcing current is much less.
- f. Open-Drain TRI-STATE L Output — This has the N-channel device to ground only.

All inputs have the following options:

- g. Input with on chip load device to V_{CC} .
- h. Hi-Z input which must be driven by the users logic.

When using either the G or L I/O ports as inputs, a pull-up device is necessary. This can be an external device or the following alternative is available: Select the low-current output option. Now, by setting the output registers to a logic "1" level, the P-channel devices will act as the pull-up load. Note that when using the L ports in this fashion the Q registers must be set to a logic "1" level and the L drivers MUST BE ENABLED by an LEI instruction (see description above).

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in Figure 12 for each of these devices to allow the designer to effectively use these I/O configurations.

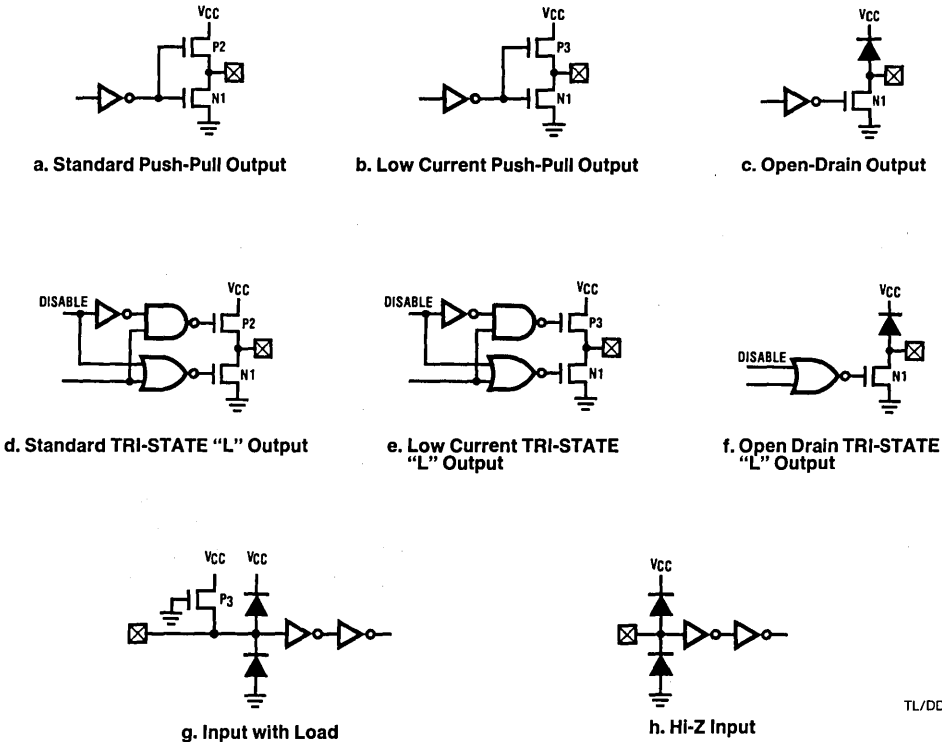


FIGURE 11. Input/Output Configurations

TL/DD/5259-14

Power Dissipation (Continued)

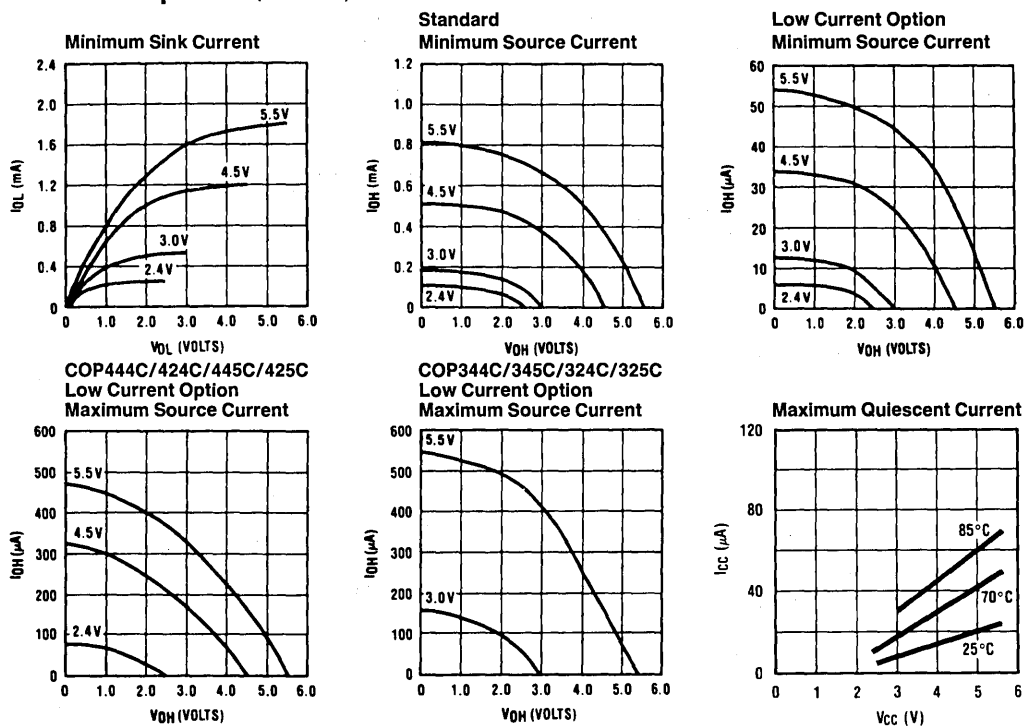


FIGURE 12. Input/Output Characteristics

TL/DD/5259-15

Option List

The COP444C/445C/424C/425C/COP426C mask-programmable options are assigned numbers which correspond with the COP444C/424C pins.

The following is a list of options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

PLEASE FILL OUT THE OPTION TABLE on the next page. Xerox the option data and send it in with your disk or EPROM.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Pin

- = 0: clock generator output to crystal/resonator
- = 1: HALT I/O port
- = 2: general purpose input with load device to V_{CC}
- = 3: general purpose input, high-Z

Option 3: CKI input

- = 0: Crystal controlled oscillator input divide by 4
- = 1: Crystal controlled oscillator input divide by 8
- = 2: Crystal controlled oscillator input divide by 16
- = 4: Single-pin RC controlled oscillator (divide by 4)
- = 5: External oscillator input divide by 4
- = 6: External oscillator input divide by 8
- = 7: External oscillator input divide by 16

Option 4: $\overline{\text{RESET}}$ input

- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 5: L7 Driver

- = 0: Standard TRI-STATE push-pull output
- = 1: Low-current TRI-STATE push-pull output
- = 2: Open-drain TRI-STATE output

Option 6: L6 Driver — (same as option 5)

Option 7: L5 Driver — (same as option 5)

Option 8: L4 Driver — (same as option 5)

Option 9: IN1 input

- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN2 input — (same as option 9)

Option 11 = 0: V_{CC} Pin — no option available

Option 12: L3 Driver — (same as option 5)

Option 13: L2 Driver — (same as option 5)

Option 14: L1 Driver — (same as option 5)

Option 15: L0 Driver — (same as option 5)

Option 16: SI input — (same as option 9)

Option 17: SO Driver

- = 0: Standard push-pull output
- = 1: Low-current push-pull output
- = 2: Open-drain output

Option List (Continued)

Option 18: SK Driver — (same as option 17)

Option 19: IN0 Input — (same as option 9)

Option 20: IN3 Input — (same as option 9)

Option 21: G0 I/O Port — (same as option 17)

Option 22: G1 I/O Port — (same as option 17)

Option 23: G2 I/O Port — (same as option 17)

Option 24: G3 I/O Port — (same as option 17)

Option 25: D3 Output — (same as option 17)

Option 26: D2 Output — (same as option 17)

Option 27: D1 Output — (same as option 17)

Option 28: D0 Output — (same as option 17)

Option 29: Internal Initialization Logic

= 0: Normal operation

= 1: No internal initialization logic

Option 30: Dual Clock

= 0: Normal operation

= 1: Dual Clock. D0 RC oscillator } (opt. #28 must = 2)

= 2: Dual Clock. D0 ext. clock input }

Option 31: Timer

= 0: No Option Available

Option 32: Microbus

= 0: Normal

= 1: Microbus (opt. #31 must = 0)

Option 33: COP bonding

(1K and 2K Microcontroller)

= 0: 28-pin package

= 1: 24-pin package

= 2: Same die purchased in both
24 and 28 pin version.

(1K Microcontroller only)

= 3: 20-pin package

= 4: 28- and 20-pin package

= 5: 24- and 20-pin package

= 6: 28-, 24- and 20-pin package

Note:—if opt. #33 = 1 or 2 then opt. #9, 10, 19, 20 and 32 must = 0—if opt. #33 = 3, 4, 5 or 6 then opt. #9, 10, 19, 20, 21, 22, 30 and 32 must = 0.

Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA	OPTION DATA
OPTION 1 VALUE = <u> 0 </u> IS: GROUND PIN	OPTION 17 VALUE = <u> </u> IS: SO DRIVER
OPTION 2 VALUE = <u> </u> IS: CKO PIN	OPTION 18 VALUE = <u> </u> IS: SK DRIVER
OPTION 3 VALUE = <u> </u> IS: CKI INPUT	OPTION 19 VALUE = <u> </u> IS: IN0 INPUT
OPTION 4 VALUE = <u> </u> IS: RESET INPUT	OPTION 20 VALUE = <u> </u> IS: IN3 INPUT
OPTION 5 VALUE = <u> </u> IS: L(7) DRIVER	OPTION 21 VALUE = <u> </u> IS: G0 I/O PORT
OPTION 6 VALUE = <u> </u> IS: L(6) DRIVER	OPTION 22 VALUE = <u> </u> IS: G1 I/O PORT
OPTION 7 VALUE = <u> </u> IS: L(5) DRIVER	OPTION 23 VALUE = <u> </u> IS: G2 I/O PORT
OPTION 8 VALUE = <u> </u> IS: L(4) DRIVER	OPTION 24 VALUE = <u> </u> IS: G3 I/O PORT
OPTION 9 VALUE = <u> </u> IS: IN1 INPUT	OPTION 25 VALUE = <u> </u> IS: D3 OUTPUT
OPTION 10 VALUE = <u> </u> IS: IN2 INPUT	OPTION 26 VALUE = <u> </u> IS: D2 OUTPUT
OPTION 11 VALUE = <u> </u> IS: VCC PIN	OPTION 27 VALUE = <u> </u> IS: D1 OUTPUT
OPTION 12 VALUE = <u> </u> IS: L(3) DRIVER	OPTION 28 VALUE = <u> </u> IS: D0 OUTPUT
OPTION 13 VALUE = <u> </u> IS: L(2) DRIVER	OPTION 29 VALUE = <u> </u> IS: INT INIT LOGIC
OPTION 14 VALUE = <u> </u> IS: L(1) DRIVER	OPTION 30 VALUE = <u> </u> IS: DUAL CLOCK
OPTION 15 VALUE = <u> </u> IS: L(0) DRIVER	OPTION 31 VALUE = <u> 0 </u> IS: TIMER
OPTION 16 VALUE = <u> </u> IS: SI INPUT	OPTION 32 VALUE = <u> </u> IS: MICROBUS
	OPTION 33 VALUE = <u> </u> IS: COP BONDING



COP444L/COP445L/COP344L/COP345L Single-Chip N-Channel Microcontrollers

General Description

The COP444L, COP445L, COP344L, and COP345L Single-Chip N-Channel Microcontrollers are members of the COPSM family, fabricated using N-channel, silicon gate MOS technology. These controller oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP445L is identical to the COP444L, but with 19 I/O lines instead of 23. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

The COP344L and COP345L are exact functional equivalents, but extended temperature range versions of the COP444L and COP445L respectively.

Features

- Low cost
- Powerful instruction set
- 2k x 8 ROM, 128 x 4 RAM
- 23 I/O lines (COP444L)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 15 μ s instruction time
- Single supply operation (4.5–6.3V)
- Low current drain (11 mA max.)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRESM serial I/O capability
- General purpose and TRI-STATE[®] outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range devices
COP344L/COP345L (–40°C to +85°C)

Block Diagram

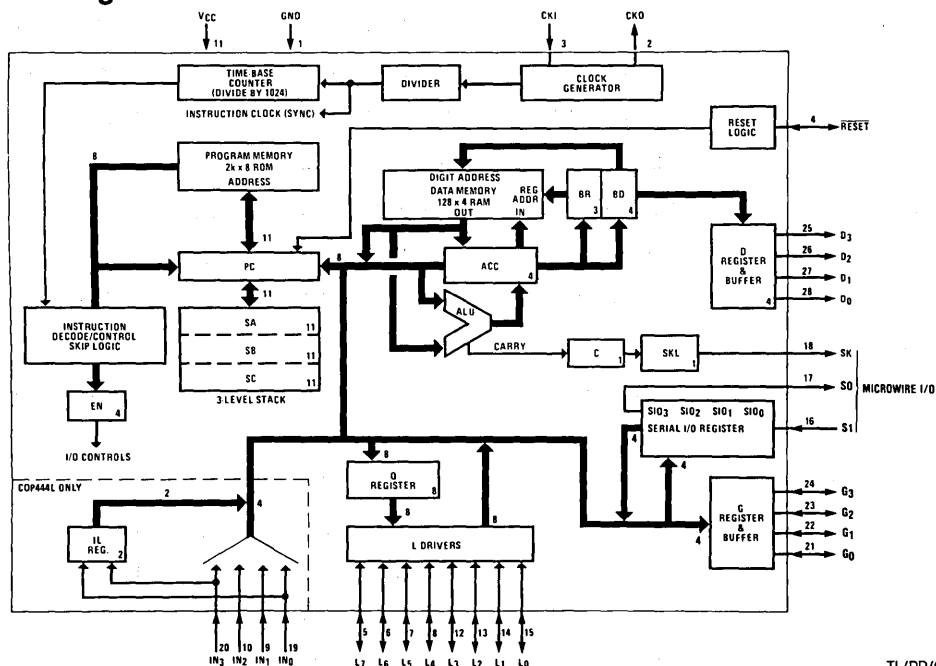


FIGURE 1

TL/DD/6928-1

COP444L/COP445L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	0.75 Watt at 25°C 0.4 Watt at 70°C

Total Source Current	120 mA
Total Sink current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 6.3\text{V}$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})		4.5	6.3	V
Power Supply Ripple (Notes 1, 4)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		13	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input ($\div 32$, $\div 16$, $\div 8$)				
Logic High (V_{IH})	$V_{CC} = \text{Max.}$	3.0		V
Logic High (V_{IH})	$V_{CC} = 5\text{V} \pm 5\%$	2.0	0.4	V
Logic Low (V_{IL})		-0.3		V
Schmitt Trigger Input ($\div 4$)				
Logic High (V_{IH})		$0.7 V_{CC}$		V
Logic Low (V_{IL})		-0.3	0.6	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		$0.7 V_{CC}$		V
Logic Low		-0.3	0.6	V
SO Input Level (Test Mode)	(Note 3)	2.0	2.5	V
All Other Inputs				
Logic High	$V_{CC} = \text{Max.}$	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, $V_{CC} = 5\text{V} \pm 10\%$	-0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 4)			7	pF
Hi-Z Input Leakage		-1	+1	μA
Output Voltage Levels				
LSTTL Operation	$V_{CC} = 5\text{V} \pm 5\%$			
Logic High (V_{OH})	$I_{OH} = -25 \mu\text{A}$	2.7		V
Logic Low (V_{OL})	$I_{OL} = 0.36 \text{ mA}$		0.4	V
CMOS Operation (Note 2)				
Logic High	$I_{OH} = -10 \mu\text{A}$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10 \mu\text{A}$		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

COP444L/COP445L (Continued)**DC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 6.3\text{V}$ unless otherwise noted. (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OL}} = 0.4\text{V}$	1.2		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.9		mA
L_0 – L_7 Outputs and Standard	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.4		mA
G_0 – G_3, D_0 – D_3 Outputs (I_{OL})	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.4		mA
G_0 – G_3 and D_0 – D_3 Outputs with	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OL}} = 1.0\text{V}$	11		mA
High Current Options (I_{OL})	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 1.0\text{V}$	7.5		mA
G_0 – G_3 and D_0 – D_3 Outputs with	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OL}} = 1.0\text{V}$	22		mA
Very High Current Options (I_{OL})	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 1.0\text{V}$	15		mA
CKI (Single-pin RC oscillator)	$V_{\text{CC}} = 4.5\text{V}, V_{\text{IH}} = 3.5\text{V}$	2		mA
CKO	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OL}} = 0.4\text{V}$	0.2		mA
Output Source Current				
Standard Configuration,				
All Outputs (I_{OH})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OH}} = 2.0\text{V}$	–75	–480	μA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 2.0\text{V}$	–30	–250	μA
Push-Pull Configuration				
SO and SK Outputs (I_{OH})	$V_{\text{CC}} = 9.5\text{V}, V_{\text{OH}} = 4.75\text{V}$	–1.4		mA
	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OH}} = 2.4\text{V}$	–1.4		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 1.0\text{V}$	–1.2		mA
LED Configuration, L_0 – L_7				
Outputs, Low Current				
Drivers Option (I_{OH})	$V_{\text{CC}} = 6.0\text{V}, V_{\text{OH}} = 2.0\text{V}$	–1.5	–13	mA
LED Configuration, L_0 – L_7				
Outputs, High Current				
Driver Option (I_{OH})	$V_{\text{CC}} = 6.0\text{V}, V_{\text{OH}} = 2.0\text{V}$	–3.0	–25	mA
TRI-STATE Configuration,				
L_0 – L_7 Outputs, Low				
Current Driver Option (I_{OH})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OH}} = 3.2\text{V}$	–0.8		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 1.5\text{V}$	–0.9		mA
TRI-STATE Configuration,				
L_0 – L_7 Outputs, High				
Current Driver Option (I_{OH})	$V_{\text{CC}} = 6.3\text{V}, V_{\text{OH}} = 3.2\text{V}$	–1.6		mA
	$V_{\text{CC}} = 4.5\text{V}, V_{\text{OH}} = 1.5\text{V}$	–1.8		mA
Input Load Source Current	$V_{\text{CC}} = 5.0\text{V}$	–10	–140	μA
CKO Output				
RAM Power Supply Option				
Power Requirement	$V_{\text{R}} = 3.3\text{V}$		3.0	mA
TRI-STATE Output Leakage Current		–2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined				
D, G Ports			120	mA
L_7 – L_4			120	mA
L_3 – L_0			4	mA
All Other Pins			4	mA
			1.5	mA
Total Source Current Allowed				
All I/O Combined				
L_7 – L_4			120	mA
L_3 – L_0			60	mA
Each L Pin			60	mA
All Other Pins			30	mA
			1.5	mA

COP344L/COP345L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	-40°C to +85°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	0.75 Watt at 25°C 0.25 Watt at 85°C

Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	5.5	V
Power Supply Ripple (Notes 1, 4)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		15	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V _{IH})	V _{CC} = Max.	3.0		V
Logic High (V _{IH})	V _{CC} = 5V ± 5%	2.2	0.3	V
Logic Low (V _{IL})		-0.3		
Schmitt Trigger Input				
Logic High (V _{IH})		0.7 V _{CC}		V
Logic Low (V _{IL})		-0.3	0.4	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.4	V
SO Input Level (Test Mode)		2.2	2.5	V
All Other Inputs				
Logic High	V _{CC} = Max.	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V _{CC} = 5V ± 5%	-0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 4)			7	pF
Hi-Z Input Leakage		-2	+2	μA
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ± 10%			
Logic High (V _{OH})	I _{OH} = -20 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)				
Logic High	I _{OH} = -10 μA	V _{CC} -1		V
Logic Low	I _{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

COP344L/COP345L (Continued)

DC Electrical Characteristics

-40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted. (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	V _{CC} = 5.5V, V _{OL} = 0.4V	1.0		mA
	V _{CC} = 4.5V, V _{OL} = 0.4V	0.8		mA
L ₀ -L ₇ Outputs, and Standard	V _{CC} = 5.5V, V _{OL} = 0.4V	0.4		mA
G ₀ -G ₃ , D ₀ -D ₃ Outputs (I _{OL})	V _{CC} = 4.5V, V _{OL} = 0.4V	0.4		mA
G ₀ -G ₃ and D ₀ -D ₃ Outputs with	V _{CC} = 5.5V, V _{OL} = 1.0V	9		mA
High Current Options (I _{OL})	V _{CC} = 4.5V, V _{OL} = 1.0V	7		mA
G ₀ -G ₃ and D ₀ -D ₃ Outputs with	V _{CC} = 5.5V, V _{OL} = 1.0V	18		mA
Very High Current Options (I _{OL})	V _{CC} = 4.5V, V _{OL} = 1.0V	14		mA
CKI (Single-Pin RC Oscillator)	V _{CC} = 4.5V, V _{IH} = 3.5V	2		mA
CKO	V _{CC} = 4.5V, V _{OL} = 0.4V	0.2		mA
Output Source Current				
Standard Configuration,	V _{CC} = 5.5V, V _{OH} = 2.0V	-55	-600	μA
All Outputs (I _{OH})	V _{CC} = 4.5V, V _{OH} = 2.0V	-28	-350	μA
Push-Pull Configuration	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.1		mA
SO and SK Outputs (I _{OH})	V _{CC} = 4.5V, V _{OH} = 1.0V	-1.2		mA
LED Configuration, L ₀ -L ₇				
Outputs, Low Current	V _{CC} = 6.0V, V _{OH} = 2.0V	-1.4	-17	mA
Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-0.7	-15	mA
LED Configuration, L ₀ -L ₇				
Outputs, High Current	V _{CC} = 6.0V, V _{OH} = 2.0V	-2.7	-34	mA
Driver Option (I _{OH})	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.4	-30	mA
TRI-STATE Configuration,				
L ₀ -L ₇ Outputs, Low	V _{CC} = 5.5V, V _{OH} = 2.7V	-0.6		mA
Current Driver Option (I _{OH})	V _{CC} = 4.5V, V _{OH} = 1.5V	-0.9		mA
TRI-STATE Configuration,				
L ₀ -L ₇ Outputs, High	V _{CC} = 5.5V, V _{OH} = 2.7V	-1.2		mA
Current Driver Option (I _{OH})	V _{CC} = 4.5V, V _{OH} = 1.5V	-1.8		mA
Input Load Source Current	V _{CC} = 5.0V	-10	-200	μA
CKO Output				
RAM Power Supply Option	V _R = 3.3V		4.0	mA
Power Requirement				
TRI-STATE Output Leakage Current		-5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L ₇ -L ₄			4	mA
L ₃ -L ₀			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L ₇ -L ₄			60	mA
L ₃ -L ₀			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics

COP444L/445L: $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 6.3\text{V}$ unless otherwise noted.

COP344L/345L: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 5.5\text{V}$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time— t_C		16	40	μs
CKI				
Input Frequency— f_i	$\div 32$ Mode	0.8	2.0	MHz
	$\div 16$ Mode	0.4	1.0	MHz
	$\div 8$ Mode	0.2	0.5	MHz
	$\div 4$ Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time (Note 2)	$f_i = 2$ MHz		120	ns
Fall Time (Note 2)			80	ns
CKI Using RC ($\div 4$)	$R = 56\text{ k}\Omega \pm 5\%$ $C = 100\text{ pF} \pm 10\%$			
Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
INPUTS:				
$\text{IN}_3\text{--}\text{IN}_0, \text{G}_3\text{--}\text{G}_0, \text{L}_7\text{--}\text{L}_0$				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: $C_L = 50\text{ pF}, R_L = 20\text{ k}\Omega, V_{\text{OUT}} = 1.5\text{V}$			
SO, SK Outputs			4.0	μs
$t_{\text{pd1}}, t_{\text{pd0}}$				
All Other Outputs			5.6	μs
$t_{\text{pd1}}, t_{\text{pd0}}$				

Note 1: Variation due to the device included.

Note 2: This parameter is only sampled and not 100% tested.

Connection Diagrams

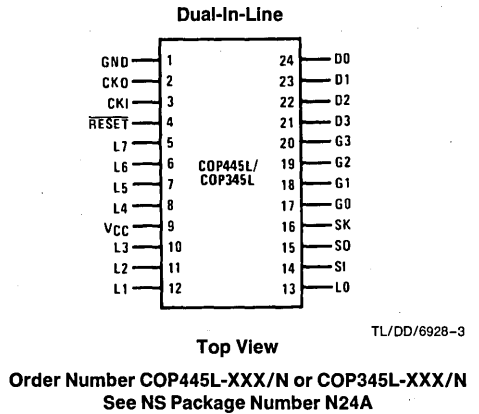
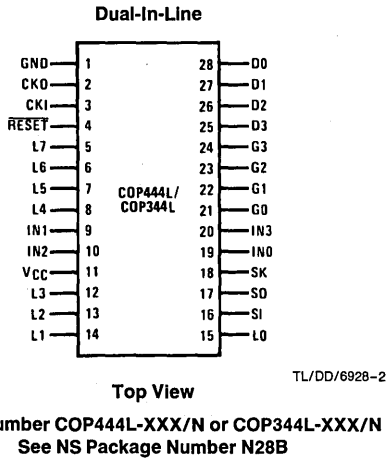


FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L7-L0	8 bidirectional I/O ports with TRI-STATE	CKI	System oscillator input
G3-G0	4 bidirectional I/O ports	CKO	System oscillator output (or general purpose input, RAM power supply, or SYNC input)
D3-D0	4 general purpose outputs	RESET	System reset input
IN3-IN0	4 general purpose inputs (COP444L only)	VCC	Power supply
SI	Serial input (or counter input)	GND	Ground
SO	Serial output (or general purpose output)		
SK	Logic-controlled clock (or general purpose output)		

Timing Diagrams

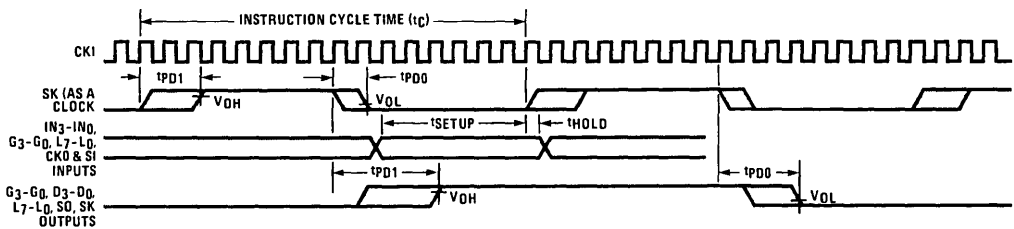


FIGURE 3a. Input/Output Timing Diagrams (Crystal Divide-by-16 Mode)

TL/DD/6928-4

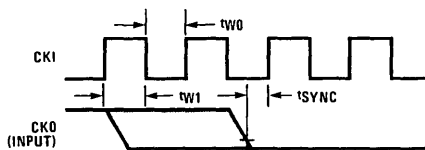


FIGURE 3b. Synchronization Timing

TL/DD/6928-5

Functional Description

A block diagram of the COP444L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

All functional references to the COP444L/COP445L also apply to the COP344L/COP345L.

PROGRAM MEMORY

Program Memory consists of a 2048 byte ROM. As can be seen by an examination of the COP444L/445L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by a 11-bit PC register. Its binary value selects one of the 2048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value. Three levels of subroutine nesting are implemented by the 11-bit subroutine save registers, SA, SB, and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register descriptor, below.)

Four general-purpose inputs, IN₃–IN₀, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃–EN₀).

1. The least significant bit of the enable register, EN₀, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN₁ set the IN₁ input is enabled as an interrupt input. Immediately following an interrupt, EN₁ is reset to disable further interrupts.
3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN₃ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides a summary of the modes associated with EN₃ and EN₀.

Functional Description (Continued)

Enable Register Modes—Bits EN₃ and EN₀

EN ₃	EN ₀	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

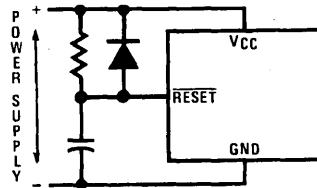
INTERRUPT

The following features are associated with the IN₁ interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN₁ is reset.
- An interrupt will be acknowledged only after the following conditions are met:
 - EN₁ has been set.
 - A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN₁ input.
 - A currently executing instruction has been completed
 - All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.
- Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.
- The first instruction of the interrupt routine at hex address 0FF must be a NOP.
- A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user use provide an external RC network and diode to the RESET pin as shown below. If the RC network is not used, the RESET pin must be pulled up to V_{CC} either by the internal load or by an external resistor (≥40 kΩ) to V_{CC}. The RESET pin is configured as a Schmitt trigger input. Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.



TL/DD/6928-6

RC ≥ 5 x Power Supply Rise Time (R ≥ 40k)

Power-Up Clear Circuit

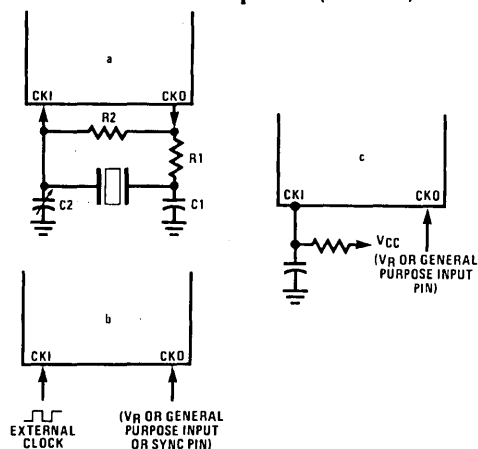
Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

OSCILLATOR

There are four basic clock oscillator configurations available as shown by Figure 4.

- Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32 (optional by 16 or 8).
- External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 32 (optional by 16 or 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R), as a general purpose input.
- RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or as a general purpose input.

Functional Description (Continued)



TL/DD/6928-7

Crystal Oscillator

Crystal Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220
2.097 MHz	1k	1M	30	6-36

RC Controlled Oscillator

R (k Ω)	C (pF)	Instruction Cycle Time (μ s)
51	100	19 \pm 15%
82	56	19 \pm 13%

NOTE: 200 k Ω \geq R \geq 25 k Ω 360 pF \geq C \geq 50 pF

FIGURE 4. COP444L/445L Oscillator

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP444L/445L system timing configuration does not require use of the CKO pin.

I/O OPTIONS

COP444L/445L outputs have the following optional configurations, illustrated in Figure 5.

- Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC} , compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.
- Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. Push-Pull—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. Standard L—same as a., but may be disabled. Available on L outputs only.

e. Open Drain L—same as b., but may be disabled. Available on L outputs only.

f. LED Direct Drive—an enhancement-mode device to ground and to V_{CC} , meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note: Series current limiting resistors have to be used if the higher operating voltage option is selected and LEDs are driven directly.

g. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC} . These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

COP444L/COP445L inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC} .

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1-6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT} curves are given in Figure 6 for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

The SO, SK outputs can be configured as shown in a., b., or c. The D and G outputs can be configured as shown in a. or b. Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured in d., e., f. or g.

An important point to remember if using configuration d. or f. with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see Figure 6, device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to logic "1".

RAM KEEP-ALIVE OPTION

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the lower four (Br = 0, 1, 2, 3) registers of RAM. To insure that RAM data integrity is maintained, the following conditions *must* be met:

- $\overline{\text{RESET}}$ must go low before V_{CC} goes low during power off; V_{CC} must go high before $\overline{\text{RESET}}$ goes high on power-up.
- V_R must be within the operating range of the chip, and equal to $V_{CC} \pm 1V$ during normal operation.
- V_R must be $\geq 3.3V$ with V_{CC} off.

Functional Description (Continued)

COP445L

If the COP444L is bonded as a 24-pin device, it becomes the COP455L, illustrated in *Figure 2*, COP444L/445L Connection Diagrams. Note that the COP445L does not contain

the four general purpose IN inputs (IN₃-IN₀). Use of this option precludes, of course, use of the IN options and the interrupt feature, which uses IN₁. All other options are available for the COP445L.

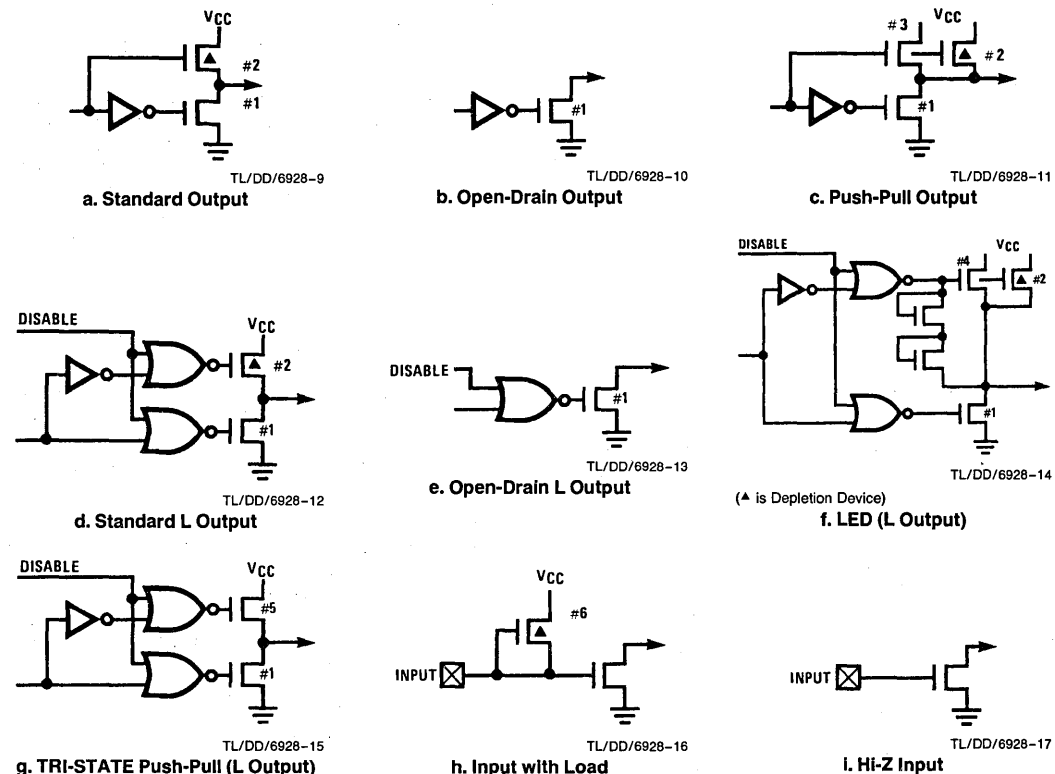


FIGURE 5. Output Configuration

L-Bus Considerations

False states may be generated on L₀-L₇ during the execution of the CAMQ instruction. The L-Ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. The following short program illustrates this situation.

START:

```

CLRA          ;ENABLE THE Q
LEI 4         ;REGISTER TO L LINES
LBI TEST
STII 3
AISC 12

```

LOOP:

```

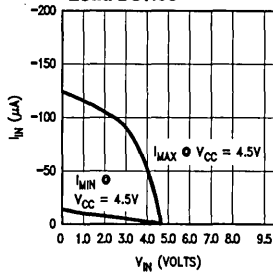
LBI TEST     ;LOAD Q WITH X'C3
CAMQ
JP LOOP

```

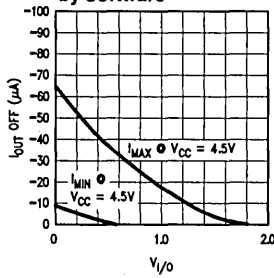
In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L₀, L₁, L₆, L₇, and logic lows on L₂-L₅ via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'C3) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L₀, L₁, L₆, L₇, and positive glitches on L₂-L₅. Glitch durations are under 2 μ s, although the exact value may vary due to data patterns, processing parameters, and the L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics

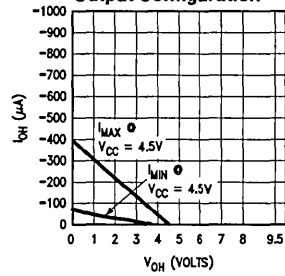
Current for Inputs with Load Device



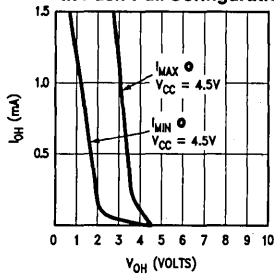
Input Current for L₀ through L₇ when Output Programmed Off by Software



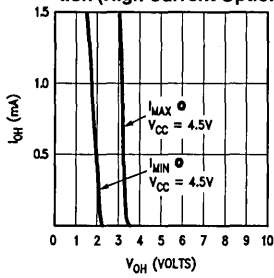
Source Current for Standard Output Configuration



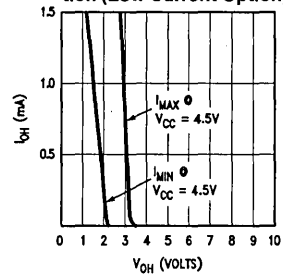
Source Current for SO and SK In Push-Pull Configuration



Source Current for L₀ through L₇ in TRI-STATE Configuration (High Current Option)



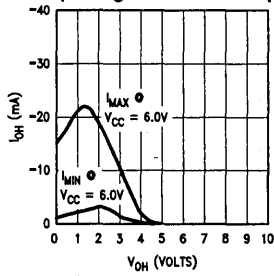
Source Current for L₀ through L₇ in TRI-STATE configuration (Low Current Option)



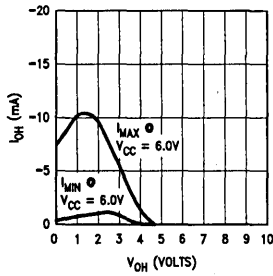
TL/DD/6928-18

Typical Performance Characteristics (Continued)

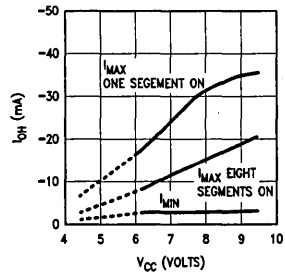
LED Output Source Current (for High Current LED Option)



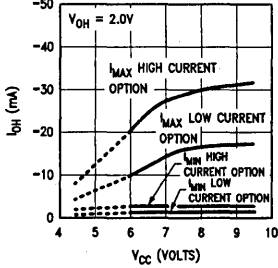
LED Output Source Current (for Low Current LED Option)



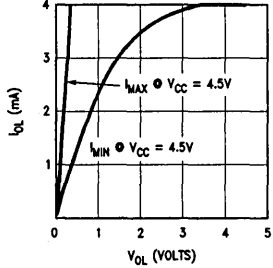
LED Output Direct Segment Drive High Current Options on L₀-L₇ Very High Current Options on D₀-D₃ or G₀-G₃



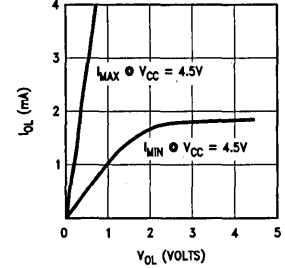
LED Output Direct Segment Drive



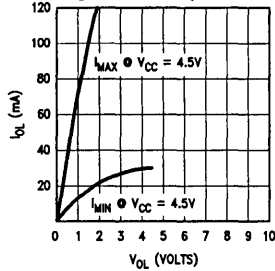
Output Sink Current for SO and SK



Output Sink Current for L₀-L₇ and Standard Drive Option for D₀-D₃ and G₀-G₃



Output Sink Current G₀-G₃ and D₀-D₃ with Very High Current Option



Output Sink Current for G₀-G₃ and D₀-D₃ (for High Current Option)

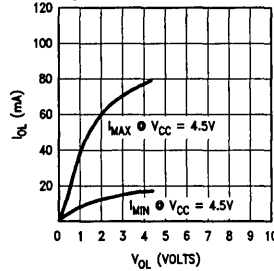


FIGURE 6a. COP444L/COP445L Input/Output Characteristics

TL/DD/6928-19

Typical Performance Characteristics (Continued)

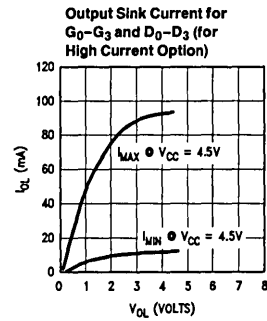
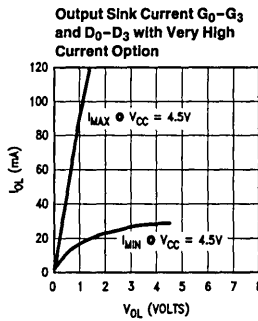
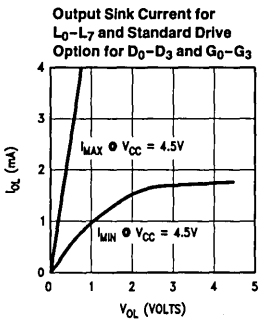
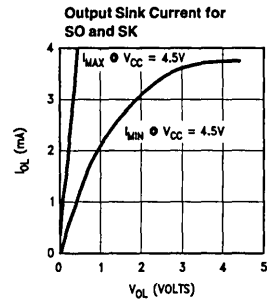
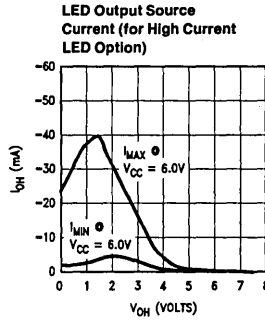
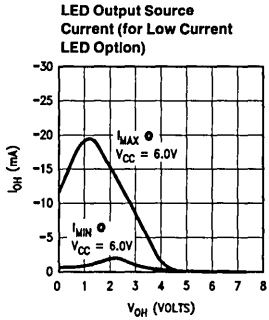
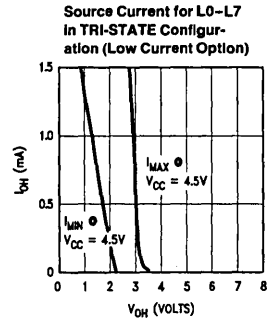
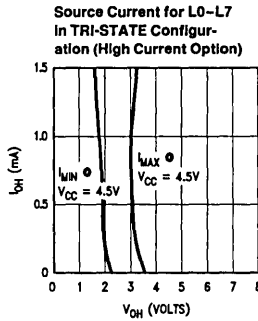
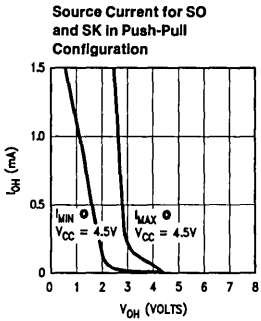
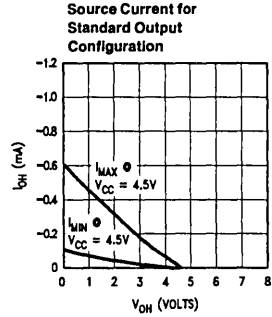
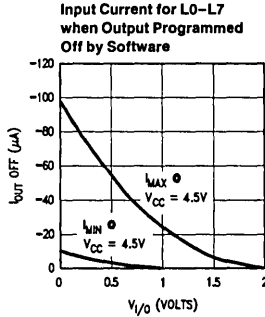
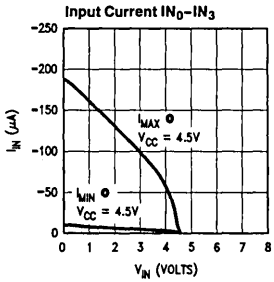


FIGURE 6b. COP344L/COP345L Input/Output Characteristics

TL/DD/6928-20

COP444L/COP445L/COP344L/COP345L Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP444L/445L instruction set.

TABLE I. COP444L/445L/344L/345L Instruction Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	3-bit Operand Field, 0-7 binary (RAM Register Select)
Br	Upper 3 bits of B (register address)	a	11-bit Operand Field, 0-2047 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0-15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register	OPERATIONAL SYMBOLS	
G	4-bit Register to latch data for G I/O Port	+	Plus
IL	Two 1-bit latches associated with the IN ₃ or IN ₀ inputs	-	Minus
IN	4-bit Input Port	→	Replaces
L	8-bit TRI-STATE I/O Port	↔	Is exchanged with
M	4-bit contents of RAM Memory pointed to by B Register	=	Is equal to
PC	11-bit ROM Address Register (program counter)	\bar{A}	The one's complement of A
Q	8-bit Register to latch data for L I/O Port	⊕	Exclusive-OR
SA	11-bit Subroutine Save Register A	:	Range of values
SB	11-bit Subroutine Save Register B		
SC	11-bit Subroutine Save Register C		
SIO	4-bit Shift Register and Counter		
SK	Logic-Controlled Clock Output		

TABLE II. COP444L/445L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	A + RAM(B) → A	None	Add RAM to A
ADT		4A	<u>0100</u> <u>1010</u>	A + 10 ₁₀ → A	None	Add Ten to A
AISC	y	5-	<u>0101</u> y	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	<u>0001</u> <u>0000</u>	\bar{A} + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	<u>0000</u> <u>0000</u>	0 → A	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	\bar{A} → A	None	Ones complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" → C	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" → C	None	Set C

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
XOR		02	0000 0010	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A
JID		FF	1111 1111	$\text{ROM}(PC_{10:8}, A, M) \rightarrow PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6--	0110 0 a _{10:8} a _{7:0}	$a \rightarrow PC$	None	Jump
JP	a	--	1 a _{6:0} (pages 2,3 only)	$a \rightarrow PC_{6:0}$	None	Jump within Page (Note 4)
			11 a _{5:0} (all other pages)	$a \rightarrow PC_{5:0}$		
JSRP	a	--	10 a _{5:0}	$PC + 1 \rightarrow SA \rightarrow SB$ $\rightarrow SC$ $00010 \rightarrow PC_{10:6}$ $a \rightarrow PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6--	0110 1 a _{10:8} a _{7:0}	$PC + 1 \rightarrow SA \rightarrow SB$ $\rightarrow SC$ $a \rightarrow PC$	None	Jump to Subroutine
RET		48	0100 1000	$SC \rightarrow SB \rightarrow SA \rightarrow PC$	None	Return from Subroutine
RETSK		49	0100 1001	$SC \rightarrow SB \rightarrow SA \rightarrow PC$	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	$A \rightarrow Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 1100	$\text{RAM}(B) \rightarrow Q_{3:0}$		
CQMA		33	0011 0011	$Q_{7:4} \rightarrow \text{RAM}(B)$	None	Copy Q to RAM, A
		2C	0010 1100	$Q_{3:0} \rightarrow A$		
LD	r	-5	00 r 0101 (r = 0:3)	$\text{RAM}(B) \rightarrow A$ $Br \oplus r \rightarrow Br$	None	Load RAM into A Exclusive-OR Br with r
LDD	r,d	23	0010 0011 0 r d	$\text{RAM}(r,d) \rightarrow A$	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	$\text{ROM}(PC_{10:8}, A, M) \rightarrow Q$ $SB \rightarrow SC$	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 1100	$0 \rightarrow \text{RAM}(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	$0 \rightarrow \text{RAM}(B)_1$		
	2	42	0100 0010	$0 \rightarrow \text{RAM}(B)_2$		
	3	43	0100 0011	$0 \rightarrow \text{RAM}(B)_3$		
SMB	0	4D	0100 1101	$1 \rightarrow \text{RAM}(B)_0$	None	Set RAM Bit
	1	47	0100 1101	$1 \rightarrow \text{RAM}(B)_1$		
	2	46	0100 0110	$1 \rightarrow \text{RAM}(B)_2$		
	3	4B	0100 1011	$1 \rightarrow \text{RAM}(B)_3$		
STII	y	7--	0111 y	$y \rightarrow \text{RAM}(B)$ $Bd + 1 \rightarrow Bd$	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110 (r = 0:3)	$\text{RAM}(B) \leftrightarrow A$ $Br \oplus r \rightarrow Br$	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	0010 0011 1 r d	$\text{RAM}(r,d) \leftrightarrow A$	None	Exchange A with RAM pointed to directly by r,d

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	00 r 0111 (r = 0:3)	RAM(B) \leftrightarrow A Bd - 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	00 r 0100 (r = 0:3)	RAM(B) \leftrightarrow A Bd + 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A \rightarrow Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd \rightarrow A	None	Copy Bd to A
LBI	r,d	--	00 r (d-1) (r = 0:3; d = 0, 9:15) or 33 -- 1 r d any r, any d)	r,d \rightarrow B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6-	0001 0011 0110 y	y \rightarrow EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A \leftrightarrow Br (0 \rightarrow A ₃)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	} 2nd byte	G ₀ = 0	
	1	11	0001 0001		G ₁ = 0	
	2	03	0000 0011		G ₂ = 0	
	3	13	0001 0011		G ₃ = 0	
SKMBZ		01 11 203 313	0000 0001 0001 0001 0000 0011 0001 0011		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G \rightarrow A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN \rightarrow A	None	Input IN Inputs to A (Note 2)
INIL		33 29A	0011 0011 0010 1001	IL ₃ , CKO, "0", IL ₀ \rightarrow A	None	Input IL Latches to A (Note 3)

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS (Continued)						
INL		33	0011 0011	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
		2E	0010 1110			
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OGI	y	33	0011 0011	y → G	None	Output to G Ports Immediate
		5-	0101 t			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin COP445L or COP345L since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14 or 15. The machine code for the lower 4 bits equals the binary value of the "d" data minus 1, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP444L/445L programs.

SOFTWARE AND OPCODE DIFFERENCES IN THE COP444L INSTRUCTION SET

The COP444L is essentially a COP420L with a double RAM and ROM. Because of this increased memory space certain instructions have expanded capability in the COP444L. Note that there are no new instructions in the COP444L and that all instructions perform the same operations in the COP444L as they did in the COP420L. The expanded capability is merely to allow appropriate handling of the increased memory space. The affected instructions are:

JMP	a	(a = address)
JSR	a	(a = address)
LDD	r,d	(r,d = RAM address Br,Bd)
XAD	r,d	(r,d = RAM address Br,Bd)
LBI	r,d	(r,d = RAM address Br,Bd; only two byte form of the instruction affected)

XABR

The JMP and JSR instructions are modified in that the address a may be anywhere within the 2048 words of ROM space. The opcodes are as follows:

JMP	0110 0 a _{10:9:8}	JSR	0110 1 a _{10:9:8}
	a _{7:0}		a _{7:0}

The LDD, XAD, and two byte LBI are modified so that they may address the entire RAM space. The opcodes are as follows:

LDD	0110 0011	XAD	0010 0011
	0 r d		1 r d
LBI	0011 0011		
	1 r d		

The XABR instruction change is transparent to the user. The opcode is not changed nor is the function of the instruction. The change is that values of 0 through 7 in A will address registers in the COP444L (i.e., the lower three bits of A become the Br value following the instruction). In the COP420L, the lower two bits of A became the Br value following an XABR instruction.

Note that those instructions which have an exclusive-or argument (LD, X, XIS, XDS) are not affected. The argument is still two bits of the opcode. This means that the exclusive-or aspect of these instructions works within blocks of four registers. It is not possible to toggle Br from a value between 0 and 3 to a value between 4 and 7 by means of these instructions.

Description of Selected Instructions (Continued)

There are no other software or opcode differences between the COP444L and the COP420L. Examination of the above changes indicates that the existing opcodes for those instructions have merely been extended. There is no fundamental change.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC_{10:8}, A, M. PC₁₀, PC₉ and PC₈ are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

INIL INSTRUCTION

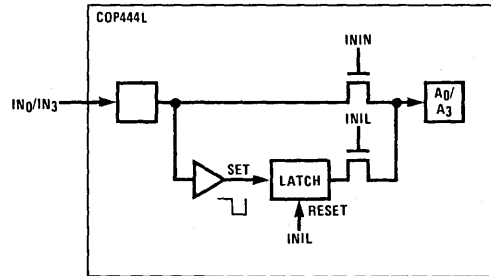
INIL (Input IL Latches to A) inputs 2 latches, IL₃ and IL₀ (see Figure 7) and CKO into A. The IL₃ and IL₀ latches are set if a low-going pulse ("1" to "0") has occurred on the IN₃ and IN₀ inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL₃ and IL₀ into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN₃ and IN₀ lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN₃-IN₀ are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset; IL₃-IL₀ not input on 445L.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC₁₀, PC₉, PC₈, A, M. LQID can be used for table lookup or code con-

version such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → PC_{7:4}, RAM(B) → PC_{3:0}, leaving PC₁₀, PC₉ and PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.



TL/DD/6928-21

FIGURE 7. INIL Hardware Implementation

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP444L/445L to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷ 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

Description of Selected Instructions (Continued)

INSTRUCTION SET NOTES

- The first word of a COP444L/445L program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.
- The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23 or 27 will access data in the next group of four pages.

Option List

The COP444L/445L mask-programmable options are assigned numbers which correspond with the COP444L pins.

The following is a list of COP444L options. When specifying a COP445L chip, Options 9, 10, 19, and 20 must all be set to zero. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground Pin—no options available

Option 2: CKO Output

- = 0: clock generator output to crystal/resonator
(0 not allowable value if option 3 = 3)
- = 1: pin is RAM power supply (V_R) input
- = 2: general purpose input, load device to V_{CC}
- = 3: general purpose input, Hi-Z

Option 3: CKI Input

- = 0: oscillator input divided by 32 (2 MHz max.)
- = 1: oscillator input divided by 16 (1 MHz max.)
- = 2: oscillator input divided by 8 (500 kHz max.)
- = 3: single-pin RC controlled oscillator divided by 4
- = 4: External Schmitt Trigger level clock divide by 4

Option 4: $\overline{\text{RESET}}$ Input

- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 5: L_7 Driver

- = 0: Standard output
- = 1: Open-drain output
- = 2: High current LED direct segment drive output
- = 3: High current TRI-STATE push-pull output
- = 4: Low-current LED direct segment drive output
- = 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver

same as Option 5

Option 7: L_5 Driver

same as Option 5

Option 8: L_4 Driver

same as Option 5

Option 9: IN_1 Input

- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN_2 Input

same as Option 9

Option 11: V_{CC} pin Operating Voltage

- | COP44XL | COP34XL |
|---------------------|----------------|
| = 0: +4.5V to +6.3V | +4.5V to +5.5V |

Option 12: L_3 Driver

same as Option 5

Option 13: L_2 Driver

same as Option 5

Option 14: L_1 Driver

same as Option 5

Option 15: L_0 Driver

same as Option 5

Option 16: SI Input

same as Option 9

Option List (Continued)

Option 17: SO Driver

- = 0: standard output
- = 1: open-drain output
- = 2: push-pull output

Option 18: SK Driver

same as Option 17

Option 19: IN₀ Input

same as Option 9

Option 20: IN₃ Input

same as Option 9

Option 21: G₀ I/O Port

- = 0: very-high current standard output
- = 1: very-high current open-drain output
- = 2: high current standard output
- = 3: high current open-drain output
- = 4: standard LSTTL output (fanout = 1)
- = 5: open-drain LSTTL output (fanout = 1)

Option 22: G₁ I/O Port

same as Option 21

Option 23: G₂ I/O Port

same as Option 21

Option 24: G₃ I/O Port

same as Option 21

Option 25: D₃ Output

same as Option 21

Option 26: D₂ Output

same as Option 21

Option 27: D₁ Output

same as Option 21

Option 28: D₀ Output

same as Option 21

Option 29: L Input Levels

- = 0: standard TTL input levels
("0" = 0.8V, "1" = 2.0V)
- = 1: higher voltage input levels
("0" = 1.2V, "1" = 3.6V)

Option 30: IN Input Levels

same as Option 29

Option 31: G Input Levels

same as Option 29

Option 32: SI Input Levels

same as Option 29

Option 33: RESET Input

- = 0: Schmitt trigger input levels
- = 1: standard TTL input levels
- = 2: higher voltage input levels

Option 34: CKO Input Levels (CKO=input; Option 2=2, 3)
same as Option 29

Option 35: COP Bonding

- = 0: COP444L (28-pin device)
- = 1: COP445L (24-pin device)
- = 2: both 28- and 24-pin versions

Option 36: Internal Initialization Logic

- = 0: normal operation
- = 1: no internal initialization logic

COP444L Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA	OPTION DATA
OPTION 1 VALUE= <u> 0 </u> IS: GROUND PIN	OPTION 21 VALUE= <u> </u> IS: G0 I/O PORT
OPTION 2 VALUE= <u> </u> IS: CKO PIN	OPTION 22 VALUE= <u> </u> IS: G1 I/O PORT
OPTION 3 VALUE= <u> </u> IS: CKI PIN	OPTION 23 VALUE= <u> </u> IS: G2 I/O PORT
OPTION 4 VALUE= <u> </u> IS: RESET INPUT	OPTION 24 VALUE= <u> </u> IS: G3 I/O PORT
OPTION 5 VALUE= <u> </u> IS: L(7) DRIVER	OPTION 25 VALUE= <u> </u> IS: D3 OUTPUT
OPTION 6 VALUE= <u> </u> IS: L(6) DRIVER	OPTION 26 VALUE= <u> </u> IS: D2 OUTPUT
OPTION 7 VALUE= <u> </u> IS: L(5) DRIVER	OPTION 27 VALUE= <u> </u> IS: D1 OUTPUT
OPTION 8 VALUE= <u> </u> IS: L(4) DRIVER	OPTION 28 VALUE= <u> </u> IS: D0 OUTPUT
OPTION 9 VALUE= <u> </u> IS: IN1 INPUT	OPTION 29 VALUE= <u> </u> IS: L INPUT LEVELS
OPTION 10 VALUE= <u> </u> IS: IN2 INPUT	OPTION 30 VALUE= <u> </u> IS: IN INPUT LEVELS
OPTION 11 VALUE= <u> 0 </u> IS: VCC PIN	OPTION 31 VALUE= <u> </u> IS: G INPUT LEVELS
OPTION 12 VALUE= <u> </u> IS: L(3) DRIVER	OPTION 32 VALUE= <u> </u> IS: SI INPUT LEVELS
OPTION 13 VALUE= <u> </u> IS: L(2) DRIVER	OPTION 33 VALUE= <u> </u> IS: RESET INPUT
OPTION 14 VALUE= <u> </u> IS: L(1) DRIVER	OPTION 34 VALUE= <u> </u> IS: CKO INPUT LEVELS
OPTION 15 VALUE= <u> </u> IS: L(0) DRIVER	OPTION 35 VALUE= <u> </u> IS: COP BONDING
OPTION 16 VALUE= <u> </u> IS: SI INPUT	OPTION 36 VALUE= <u> </u> IS: INTERNAL INITIALIZATION LOGIC
OPTION 17 VALUE= <u> </u> IS: SO DRIVER	
OPTION 18 VALUE= <u> </u> IS: SK DRIVER	
OPTION 19 VALUE= <u> </u> IS: IN0 INPUT	
OPTION 20 VALUE= <u> </u> IS: IN3 INPUT	

Typical Applications

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP444L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

- a. RAM and Internal Logic Test Mode (SI = 1)
- b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATION # 1: COP444L GENERAL CONTROLLER

Figure 8 shows an interconnect diagram for a COP444L used as a general controller. Operation of the system is as follows:

1. The L₇-L₀ outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display
2. The D₃-D₀ outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN₃-IN₀ inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a general-purpose input.

5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G₃-G₀) are available for use as required by the user's application.
7. Normal reset operation is selected.

COP444L EVALUATION (See COP Note 4)

The 444L-EVAL is a pre-programmed COP444L, containing several routines which facilitate user familiarization and evaluation of the COP444L operating characteristics. It may be used as an up/down counter or timer, interfacing to any combination of (1) an LED digit or lamps, (2) 4-digit LED Display Controller, (3) a 4-digit VF Display Controller, and/or (4) a 4-digit LCD Display Controller. Alternatively, it may be used as a simple music synthesizer.

SAMPLE CIRCUITS

1. By making only the oscillator, power supply and "L7" connections, (Figure 9) an approximate 1 Hz square wave will be produced at output "D1." This output may be observed with an oscilloscope, or connected to additional TTL or CMOS circuitry.
2. By making the indicated connections to a small LED digit (NSA1541A, NSA1166, or equiv.—larger digits will be proportionately dimmer), the counter actions may be observed. Place the "up/down" switch in the "up" (open) position and apply a TTL-compatible signal at the "counter-input." Placing the "up/down" switch in the "down" (closed) position causes the count to decrement on each high-to-low input transition.
3. All 4 digits of the counter may be displayed by connecting a standard display controller (COP472 for LCD, MM5450 for LED) as shown in Figure 9.

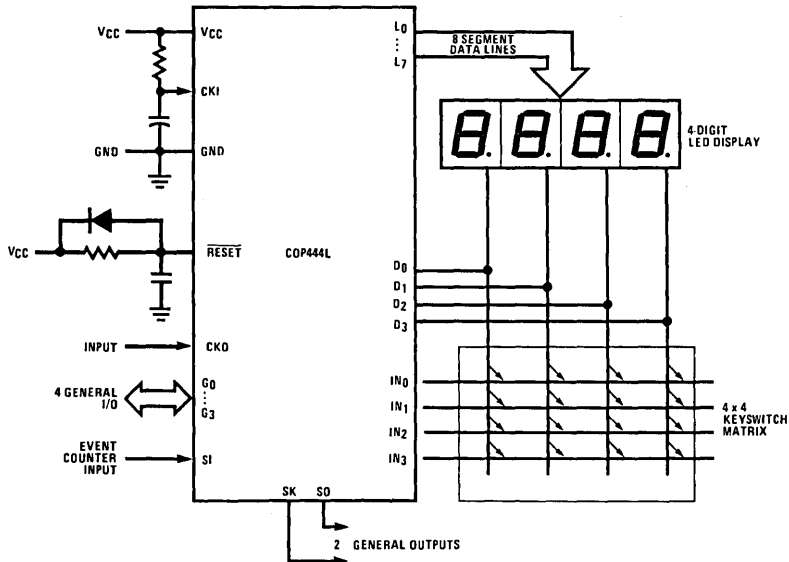


FIGURE 8. COP444L Keyboard/Display Interface

TL/DD/6928-22

Typical Applications (Continued)

Any combination of the single LED digit and display controllers may be used simultaneously, and will display the same data.

4. The simple counter described above becomes a timer when the 1 Hz output is connected to the "counter input." Up or down counting may be used with input frequencies up to 1 kHz. Improved timing accuracies may be obtained by substituting the 2.097 MHz crystal oscillator circuit of *Figure 4a* for the RC network shown in *Figure 9*, or by connecting a more stable external frequency to the "counter input" in place of the 1 Hz signal.
5. An "entertaining" use of the 444L-EVAL is as a simple music synthesizer (or electronic organ). By attaching a simple switch matrix (or keyboard), a speaker or piezo-ceramic transducer, and grounding "L7", the user can play "music" (*Figure 10*). Three modes of operation are available: Play a note, play one of four stored tunes, or record a tune for subsequent replay.

a. Play A Note

Twelve keys, representing the 12 notes in one octave, are labeled "C" through "B"; depressing a key causes

a square wave of the corresponding frequency to be outputted to the speaker. Depressing "LShift" or "UShift" causes the next note to be shifted to the next lower octave (one-half frequency) or the next upper octave (double frequency), respectively.

b. Play Stored Tune

Depressing "Play" followed by " $\frac{1}{8}$ ", " $\frac{1}{4}$ ", " $\frac{1}{2}$ ", or "1" will cause one of 4 stored tunes to be played.

c. Record Tune

Any combination of notes and rests up to a total of 48 may be stored in RAM for later replay. To store a note, press the appropriate note key, followed by the duration of the note ($\frac{1}{8}$ -note, $\frac{1}{4}$ -note, $\frac{1}{2}$ -note, whole (1)-note, followed by "Store"; a rest is stored by selecting the duration and pressing "Store." When the tune is complete, press "Play" followed by "Store"; the tune will be played for immediate audition. Subsequent depression of "Play" and "Store" will replay the last stored tune.

Note: The accuracy of the tones produced is a function of the oscillator accuracy and stability; the crystal oscillator is recommended.

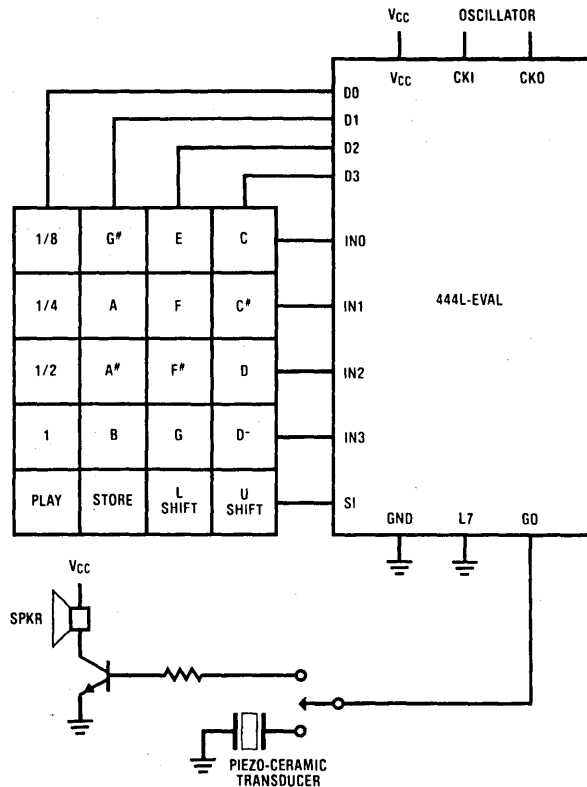
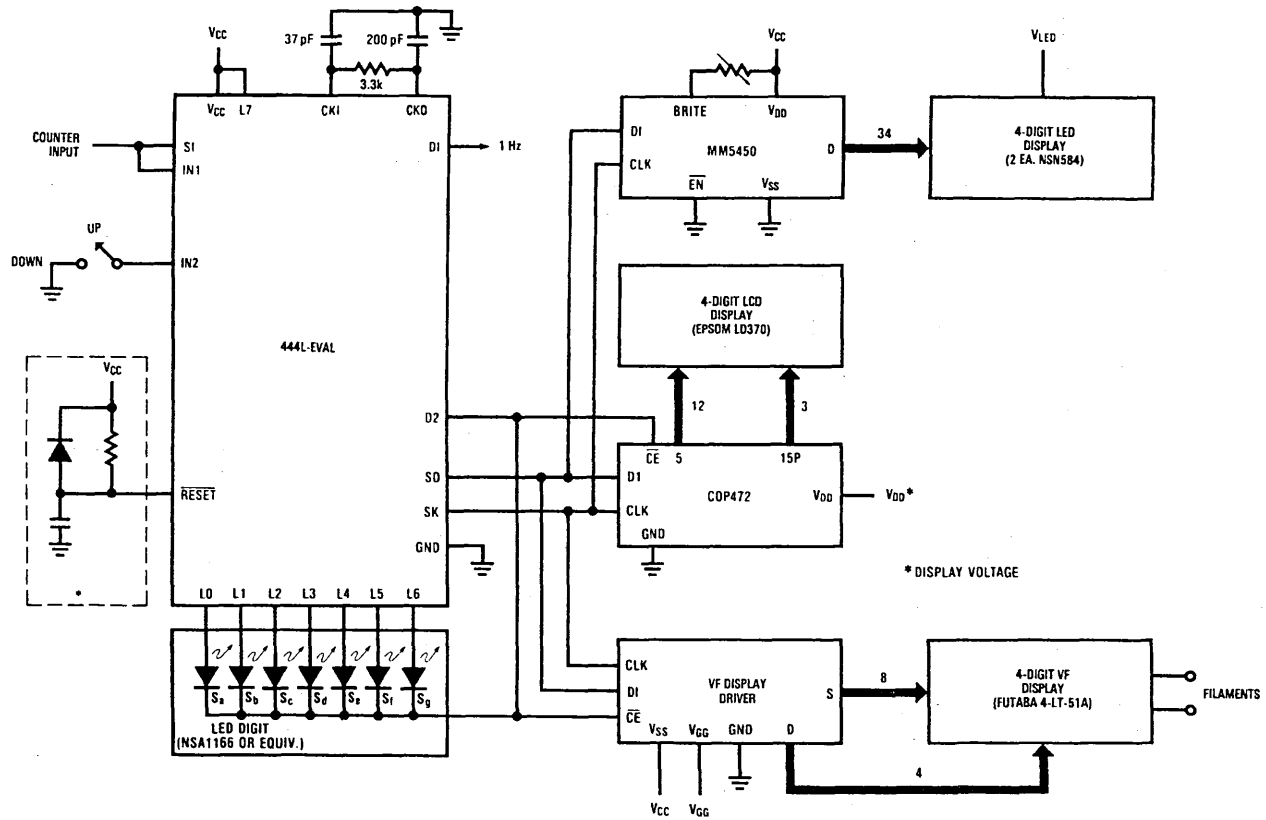


FIGURE 9. Counter/Timer

TL/DD/6928-23



* See "Initialization"

FIGURE 10. Music Synthesizer

TL/DD/6928-24



COP401L ROMless N-Channel Microcontroller

General Description

The COP401L ROMless Microcontroller is a member of the COPSM family of microcontrollers, fabricated using N-channel, silicon gate MOS technology. The COP401L contains CPU, RAM, I/O and is identical to a COP410L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP401L will perform exactly as the COP410L. This important benefit facilitates development and debug of a COP program prior to masking the final part.

The COP401L is intended for emulation only, not intended for volume production. Use COP402 or COP404L for volume production.

Features

- Circuit equivalent of COP410L
- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- Separate RAM power supply pin for RAM keep-alive applications
- Two-level subroutine stack
- 15 μ s instruction time
- Single supply operation (4.5–5.5V)
- Low current drain (8 mA max.)
- Internal binary counter register with serial I/O
- MICROWIRETM compatible serial I/O
- General purpose outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Pin-for-pin compatible with COP402 and COP404L

Block Diagram

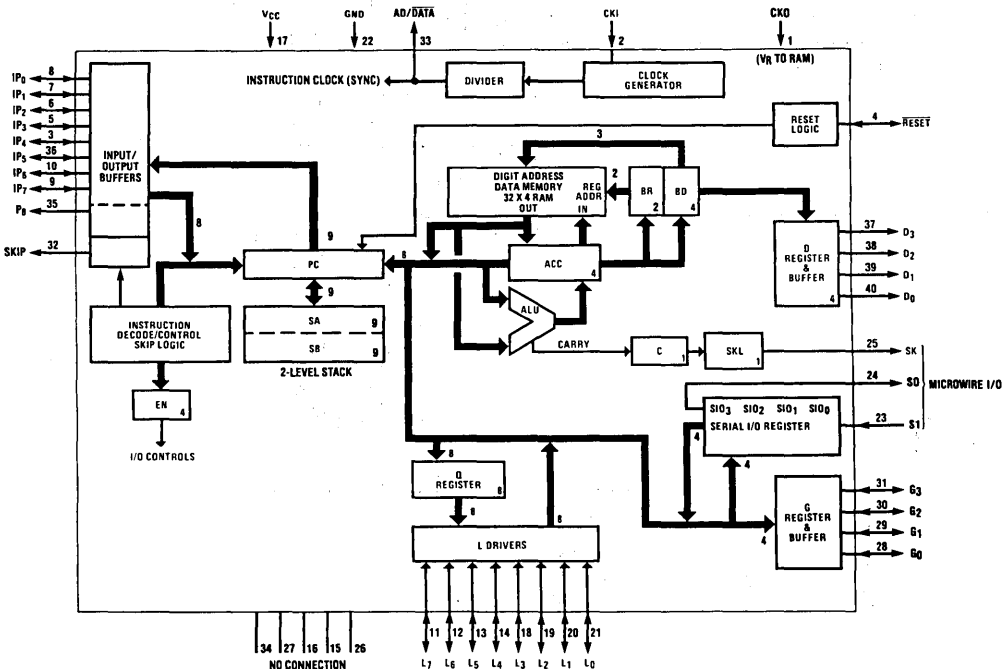


FIGURE 1

TL/DD/6913-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temp. (Soldering, 10 sec.)	300°C

Power Dissipation	0.75W at 25°C
	0.4W at 70°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})		4.5	5.5	V
Power Supply Ripple (Notes 2, 3)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input	$V_{CC} = \text{Max}$			
Logic High (V_{IH})		2.0		V
Logic Low (V_{IL})		-0.3	0.4	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		$0.7 V_{CC}$		V
Logic Low		-0.3	0.6	V
IP0-IP7 Input Levels				
Logic High				V
Logic High	$V_{CC} = 5\text{V} \pm 5\%$	2.0		V
Logic Low		-0.3	0.8	V
All Other Inputs				
Logic High				V
Logic High	$V_{CC} = 5\text{V} \pm 5\%$	2.0		V
Logic Low		-0.3	0.8	V
Input Capacitance (Note 3)			7	pF
Output Voltage Levels				
LSTTL Operation	$V_{CC} = 5\text{V} \pm 10\%$			
Logic High (V_{OH})	$I_{OH} = -25 \mu\text{A}$	2.7		V
Logic Low (V_{OL})	$I_{OL} = 0.36 \text{ mA}$		0.4	V
IP0-IP7, P8, SKIP	(Note 1)			
Logic Low	$I_{OL} = 1.6 \text{ mA}$		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 4.5\text{V}$, $V_{OL} = 0.4\text{V}$	0.9		mA
L ₀ -L ₇ and G ₀ -G ₃ Outputs	$V_{CC} = 4.5\text{V}$, $V_{OL} = 0.4\text{V}$	0.4		mA
D ₀ -D ₃ Outputs	$V_{CC} = 4.5\text{V}$, $V_{OL} = 1.0\text{V}$	15		mA
CKO				
RAM Power Supply Input	$V_R = 3.3\text{V}$		1.5	mA

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 9.5\text{V}$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Source Current				
D ₀ -D ₃ , G ₀ -G ₃ Outputs (I _{OH})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OH}} = 2.0\text{V}$	-30	-250	μA
SO and SK Outputs (I _{OH})	$V_{\text{CC}} = 4.5\text{V}$, $V_{\text{OH}} = 1.0\text{V}$	-1.2		mA
L ₀ -L ₇ Outputs	$V_{\text{CC}} = 5.5\text{V}$, $V_{\text{OH}} = 2.0\text{V}$	-0.3	-25	mA
Input Load Source Current (I _I)	$V_{\text{CC}} = 5.0\text{V}$, $V_L = 0\text{V}$	-10	-140	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D Port			100	mA
L ₇ -L ₄ , G Port			4	mA
L ₃ -L ₀			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L ₇ -L ₄			60	mA
L ₃ -L ₀			60	mA
Each L Pin			25	mA
All Other Pins			1.5	mA

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 5.5\text{V}$ unless otherwise specified.

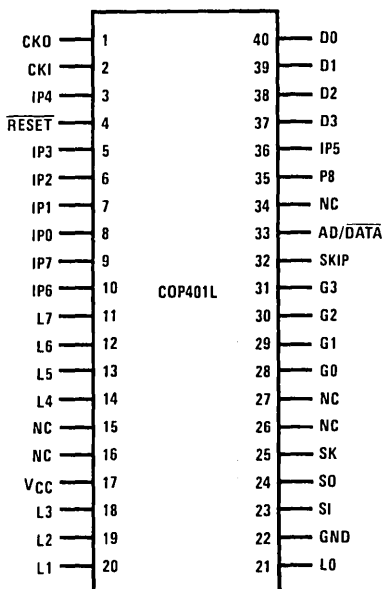
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		15	40	μs
CKI				
Input Frequency f_i	(÷32 Mode)	0.8	2.1	MHz
Duty Cycle		30	60	%
Rise Time (Note 3)	$f_i = 2.097\text{ MHz}$		120	ns
Fall Time (Note 3)			80	ns
INPUTS:				
SI, IP7-IP0				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
G ₃ -G ₀ , L ₇ -L ₀				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
OUTPUT PROPAGATION DELAY	Test Condition: $C_L = \text{pF}$, $V_{\text{OUT}} = 1.5\text{V}$ $R_L = 20\text{ k}\Omega$			
SO, SK Outputs	$R_L = \text{k}\Omega$		4.0	μs
D ₃ -D ₀ , G ₃ -G ₀ , L ₇ -L ₀	$R_L = \text{k}\Omega$		5.6	μs
IP7-IP0, P8, SKIP	$R_L = 5\text{ k}\Omega$		7.2	μs

Note 1: Pull-up resistors required.

Note 2: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 3: This parameter is only sampled and not 100% tested. Variation due to the device included.

Connection Diagram



Order Number COP401L/N
NS Package Number N40A

TL/DD/6913-2

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L7-L0	8 bidirectional I/O ports with LED segment drive	CKI	System oscillator input
G3-G0	4 bidirectional I/O ports	CKO	RAM power supply input
D3-D0	4 general purpose outputs	RESET	System reset input
SI	Serial input (or counter input)	VCC	Power supply
SO	Serial output (or general purpose output)	GND	Ground
SK	Logic-controlled clock (or general purpose output)	IP7-IP0	8 bidirectional ROM address and data ports
AD/DATA	Address Out/data in flag	P8	Most significant ROM address bit output
		SKIP	Instruction skip output

Timing Diagram

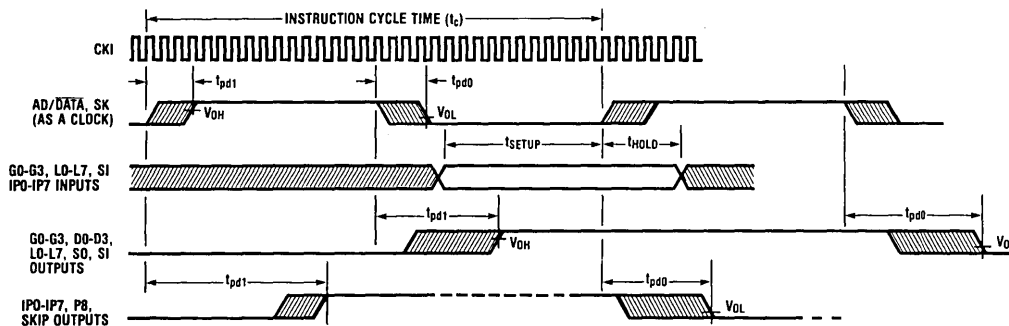


FIGURE 3. Input/Output

TL/DD/6913-3

Functional Description

A block diagram of the COP401L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

PROGRAM MEMORY

Program Memory consists of a 512-byte external memory. As can be seen by an examination of the COP401L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs. The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table 3).

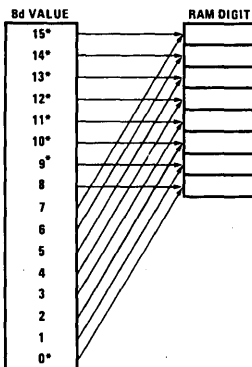


FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

TL/DD/6913-4

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP401L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa-Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift register.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃-EN₀).

1. The least significant bit of the enable register, EN₀, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.

2. EN₁ is not used. It has no effect on COP401L operation.

Functional Description (Continued)

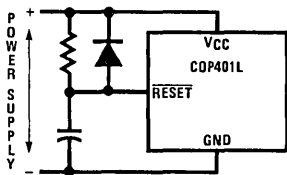
TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

- With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
- EN_3 , in conjunction with EN_0 , affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3 . With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0 .

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below (Figure 5). The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC} . Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.



TL/DD/6913-5

$RC \geq$ Power Supply Rise Time

FIGURE 5. Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP401L is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

- random addressing
- TTL-compatible TRI-STATE® outputs
- TTL-compatible inputs
- access time = 5 μ s max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P8 and IP7 through IP0 during the time that AD/DATA is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/DATA line; P8 is a dedicated address output, and does not need to be latched. When AD/DATA is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/DATA output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

CKI is an external clock input signal. The external frequency is divided by 32 to give the instruction cycle time. The divide-by-32 configuration was chosen to make the COP 401L compatible with the COP404L and the COPS™ Development System. However, the $\div 32$ configuration is not available on the COP410L/COP411L. It is therefore possible to exactly emulate the system speed (cycle time), but not possible to drive the 401L with the system clock during emulation.

Functional Description (Continued)

CKO (RAM POWER)

CKO is configured as a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. This pin must be connected to V_{CC} if the power backup feature is not used. To insure that RAM integrity is maintained, the following conditions must be met:

1. \overline{RESET} must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before \overline{RESET} goes high on power-up.
2. During normal operation, V_R must be within the operating range of the chip with $(V_{CC}-1) \leq V_R \leq V_{CC}$.
3. V_R must be $\geq 3.3V$ with V_{CC} off.

INPUT/OUTPUT CONFIGURATIONS

COP401L outputs have the following configurations, illustrated in *Figure 6*:

- Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC} , compatible with LSTTL and CMOS input requirements. (Used on D and G outputs.)
- Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. (Used on IP, P and SKIP outputs.)
- Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled en-

hancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. (Used on SO and SK outputs.)

- LED Direct Drive**—an enhancement-mode device to ground and to V_{CC} , meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. (Used on L outputs.)

COP401L inputs have an on-chip depletion load device to V_{CC} .

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of five devices (numbered 1–5, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 7* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 7*, Device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to a logic "1".

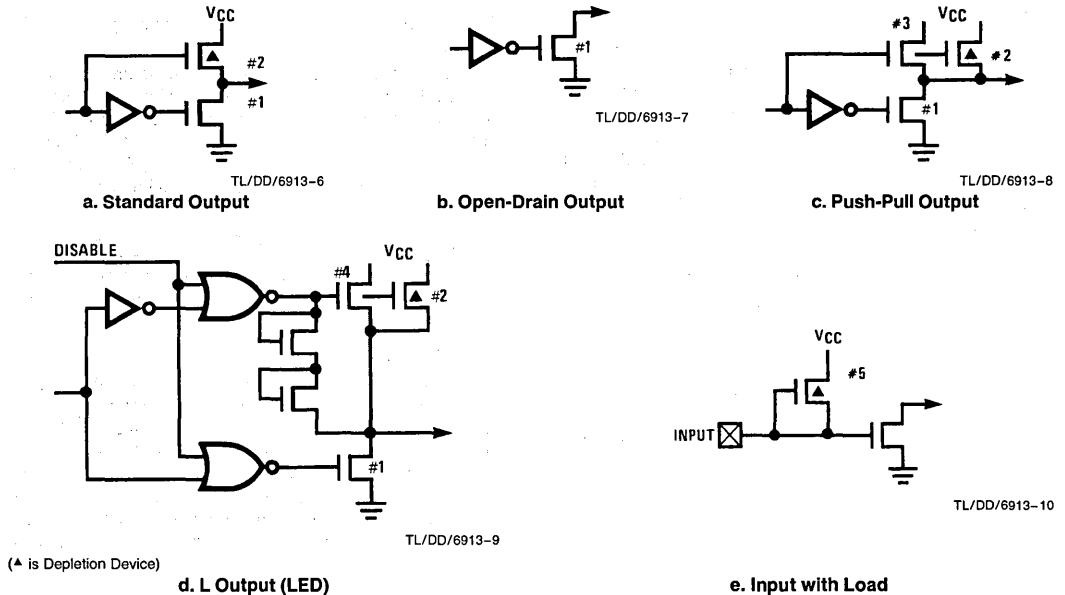


FIGURE 6. Output Configurations

Typical Performance Characteristics

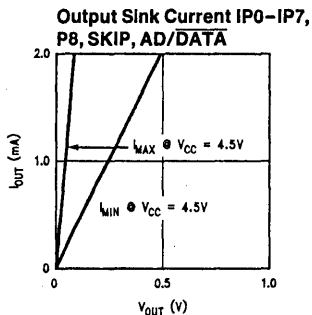
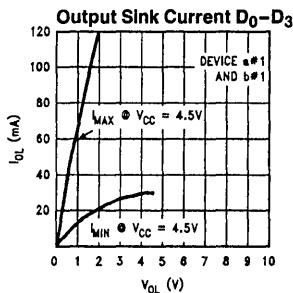
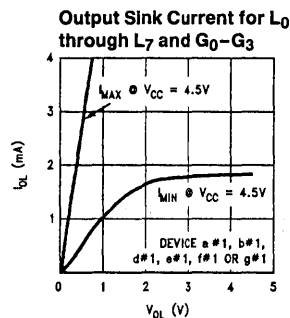
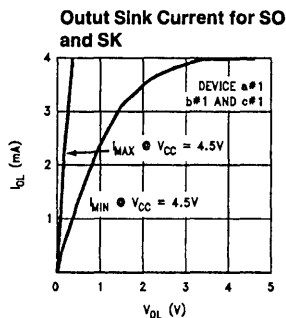
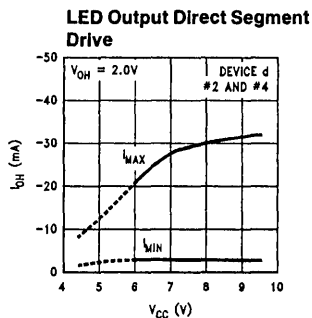
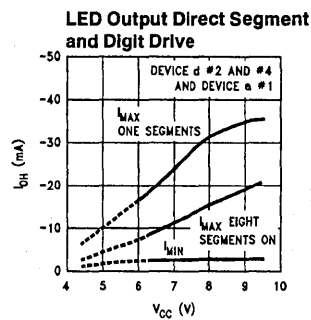
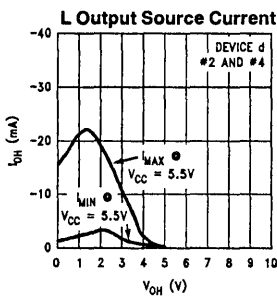
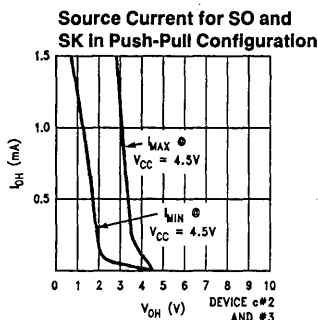
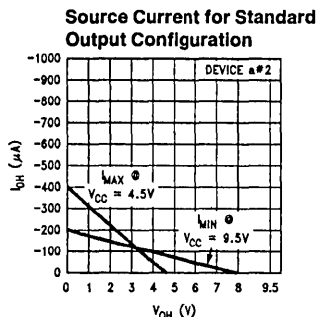
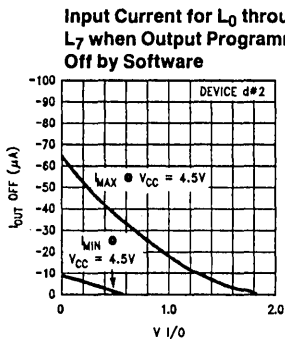
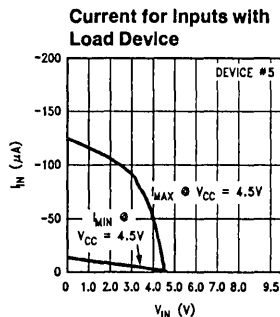


FIGURE 7. I/O Characteristics

TL/DD/69134-11

COP401L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP401L instruction set.

TABLE II. COP401L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
-	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
\bar{A}	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP401L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5-	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

COP410L Instruction Set (Continued)

TABLE III. COP401L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	$\boxed{1111} \boxed{1111}$	ROM (PC ₈ , A, M) → PC _{7:0}	None	Jump Indirect (Note 2)
JMP	a	6-- --	$\boxed{0110} \boxed{000} \boxed{a_8}$ $\boxed{a_{7:0}}$	a → PC	None	Jump
JP	a	--	$\boxed{1} \boxed{a_{6:0}}$ (pages 2,3 only) or $\boxed{11} \boxed{a_{5:0}}$ (all other pages)	a → PC _{6:0} a → PC _{5:0}	None	Jump within Page (Note 3)
JSRP	a	--	$\boxed{10} \boxed{a_{5:0}}$	PC + 1 → SA → SB 010 → PC _{8:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 4)
JSR	a	6-- --	$\boxed{0110} \boxed{100} \boxed{a_8}$ $\boxed{a_{7:0}}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	$\boxed{0100} \boxed{1000}$	SB → SA → PC	None	Return from Subroutine
RETSK		49	$\boxed{0100} \boxed{1001}$	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	$\boxed{0011} \boxed{0011}$ $\boxed{0011} \boxed{1100}$	A → Q _{7:4} RAM(B) → Q _{3:0}	None	Copy A, RAM to Q
LD	r	-5	$\boxed{00} \boxed{r} \boxed{0101}$	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	$\boxed{1011} \boxed{1111}$	ROM(PC ₈ , A, M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0 1 2 3	4C 45 42 43	$\boxed{0100} \boxed{1100}$ $\boxed{0100} \boxed{0101}$ $\boxed{0100} \boxed{0010}$ $\boxed{0100} \boxed{0011}$	0 → RAM(B) ₀ 0 → RAM(B) ₁ 0 → RAM(B) ₂ 0 → RAM(B) ₃	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	$\boxed{0100} \boxed{1101}$ $\boxed{0100} \boxed{0111}$ $\boxed{0100} \boxed{0110}$ $\boxed{0100} \boxed{1011}$	1 → RAM(B) ₀ 1 → RAM(B) ₁ 1 → RAM(B) ₂ 1 → RAM(B) ₃	None	Set RAM Bit
STII	y	7--	$\boxed{0111} \boxed{y}$	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	$\boxed{00} \boxed{r} \boxed{0110}$	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	$\boxed{0010} \boxed{0011}$ $\boxed{1011} \boxed{1111}$	RAM(3,15) ↔ A	None	Exchange A with RAM (3,15)
XDS	r	-7	$\boxed{00} \boxed{r} \boxed{0111}$	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	$\boxed{00} \boxed{r} \boxed{0100}$	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

COP410L Instruction Set (Continued)

TABLE III. COP401L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r, d	-	00 r (d - 1) (d = 0, 9:15)	r, d → B	Skip until not a LBI	Load B Immediate with r, d (Note 5)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ	0 1 2 3	33 01 11 03 13	0011 0011 0000 0001 0001 0001 0000 0011 0001 0011	1st byte 2nd byte	G ₀ = 0 G ₁ = 0 G ₂ = 0 G ₃ = 0	Skip if G Bit is Zero
SKMBZ	0 1 2 3	01 11 03 13	0000 0001 0001 0001 0000 0011 0001 0011		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 0011 0010 1110	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (010 is loaded into the upper 3 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data minus 1, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP401L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC₈, A, M. PC₈ is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC₈, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → PC_{7,4}, RAM(B) → PC_{3,0}, leaving PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

- The first word of a COP401L program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.
- The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or 7 will access data in the next group of 4 pages.

Typical Applications

PROM-BASED SYSTEM

The COP401L may be used to emulate the COP410L. *Figure 8* shows the interconnect to implement a COP401L hardware emulation. This connection uses one MM5204 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP₇–IP₀ are bidirectional inputs and outputs. When the AD/ $\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP₇–IP₀ output addresses. The 8-bit latch (MM74C373) latches the address to drive the memory.

When AD/ $\overline{\text{DATA}}$ turns off, the EPROM is enabled and the IP₇–IP₀ pins will input the memory data. PB outputs the most significant address bit to the memory. (SKIP output may be used for program debug if needed.)

24 of the COP401L pins may be configured exactly the same as a COP410L.

Typical Applications (Continued)

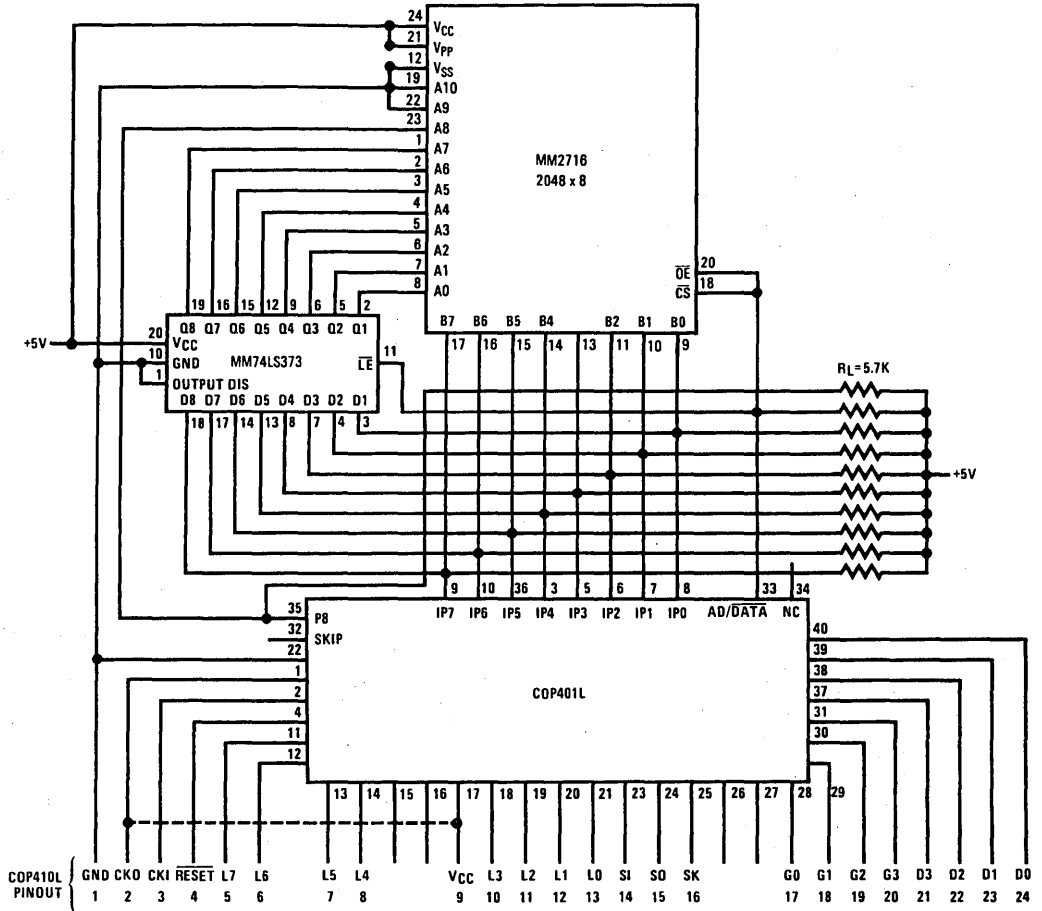


FIGURE 8. COP401L Used to Emulate a COP410L

TL/DD/6913-12

Option Table

COP401L MASK OPTIONS

The following COP401L options have been implemented in this basic version of the COP401L.

Option Value	Comment	Option Value	Comment
Option 1 = 0	Ground—no option	Option 14 = 0	SI has load to V_{CC}
Option 2 = 1	CKO is RAM power supply input	Option 15 = 2	SO is push-pull output
Option 3 = N/A	CKI is external clock divide-by-32 (not available on COP401L)	Option 16 = 2	SK is push-pull output
Option 4 = 0	Reset has load to V_{CC}	Option 17 = 0	
Option 5 = 2		Option 18 = 0	G outputs are standard
Option 6 = 2		Option 19 = 0	
Option 7 = 2	L outputs are LED direct-drive	Option 20 = 0	
Option 8 = 2		Option 21 = 0	
Option 9 = 1	V_{CC} pin 4.5V to 9.5V operation	Option 22 = 0	D outputs are standard very high current
Option 10 = 2		Option 23 = 0	
Option 11 = 2		Option 24 = 0	
Option 12 = 2	L outputs are LED direct-drive	Option 25 = 0	L
Option 13 = 2		Option 26 = 0	G Have standard TTL input levels
		Option 27 = 0	SI
		Option 28 = N/A	40-pin package



COP401L-X13/COP401L-R13 ROMless N-Channel Microcontroller

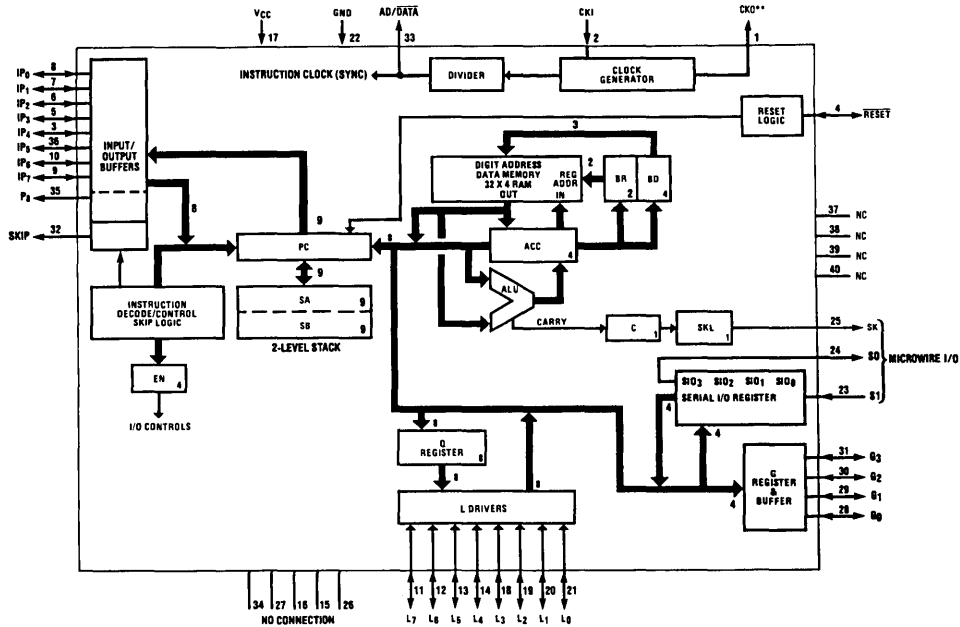
General Description

The COP401L-X13/COP401L-R13 ROMless Microcontrollers are members of the COPSTM family of microcontrollers, fabricated using N-channel, silicon gate MOS technology. The COP401L-X13/COP401L-R13 contain CPU, RAM, I/O and are identical to a COP413L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP401L-X13/COP401L-R13 will perform exactly as the COP413L. This important benefit facilitates development and debug of a COP program prior to masking the final part. There are two clock oscillator configurations available. The crystal oscillator configuration is called COP401L-X13 and the RC oscillator configuration is called COP401L-R13.

Features

- Circuit equivalent of COP413L
- Low cost
- Powerful instruction set
- 512 × 8 ROM, 32 × 4 RAM
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5–5.5V)
- Low current drain (8 mA max)
- Internal binary counter register with serial I/O
- MICROWIRE™ compatible serial I/O
- General purpose outputs
- Software/hardware compatible with other members of COP400 family
- Pin-for-pin compatible with COP402 and COP404L
- High noise immunity inputs ($V_{IL} = 1.2V$, $V_{IH} = 3.6V$)

Block Diagram



**COP401L-X13 only

FIGURE 1

TL/DD/8528-1

COP401L-X13/COP401L-R13 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.3 to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C

Power Dissipation COP413L 0.3 Watt at 70°C

Total Source Current 25 mA

Total Sink Current 40 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V _{CC})		4.5	5.5	V
Power Supply Ripple (Notes 1, 3)	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷ 8)				
Logic High (V _{IH})		3.0		V
Logic Low (V _{IL})			0.4	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V _{CC}		V
Logic Low			0.6	V
SO Input Level (Test Mode)	(Note 2)	2.5		V
IP0-IP7, SI Input Level				
Logic High	(TTL Level)	2.0		V
Logic Low			0.8	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance (Note 3)			7	pF
Reset Input Leakage		-1	+1	μA
Output Current Levels				
Output Sink Current (I _{OL})				
SO and SK Outputs	V _{OL} = 0.4V	0.9		mA
L0-L7 Outputs, G0-G3	V _{OL} = 0.4V	0.4		mA
CKO	V _{OL} = 0.4V	0.2		mA
IP0-IP7, P8, SKIP, AD/ $\overline{\text{DATA}}$	V _{OL} = 0.4V	1.6		mA
Output Source Current (I _{OH})				
L0-L7 G0-G3, SO, SK	V _{OH} = 2.4V	-25		μA
IP0-IP7, P8, SKIP, AD/ $\overline{\text{DATA}}$	V _{OH} = 2.4V	-25		μA
SO, SK	V _{OH} = 1.0V	-1.2		mA
IP0-IP7, P8, SKIP, AD/ $\overline{\text{DATA}}$	V _{OH} = 1.0V	-1.2		mA
SI Input Load Source Current	V _{IL} = 0V	-10	-140	μA
Total Sink Current Allowed				
L7-L4, G Port			4	mA
L3-L0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

Note 3: This parameter is only sampled and not 100% tested. Variation due to the device included.

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 5.5\text{V}$

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time - t_c		16	40	μs
CKI				
Input Frequency - f_i	$\div 8$ Mode	0.2	0.5	MHz
Duty Cycle		30	60	%
Rise Time (Note 2)	$f_i = 0.5$ MHz		500	ns
Fall Time (Note 2)			200	ns
CKI Using RC ($\div 4$)	R = $56\text{ k}\Omega \pm 5\%$ C = $100\text{ pF} \pm 10\%$			
Instruction Cycle Time (Note 1)		16	28	μs
Inputs:				
G3-G0, L7-L0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI, IP0-IP7				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: $C_L = 50\text{ pF}$, $V_{\text{OUT}} = 1.5\text{V}$ $R_L = 20\text{ k}\Omega$			
SO, SK Outputs			4.0	μs
t_{pd1} , t_{pd0}	$R_L = 20\text{ k}\Omega$			
L, G Outputs			5.6	μs
t_{pd1} , t_{pd0}	$R_L = 5\text{ k}\Omega$			
IP0-IP7, P8, SKIP			7.2	μs
t_{pd1} , t_{pd0}				

Note 1: Variation due to the device included.

Note 2: This parameter is only sampled and not 100% tested.

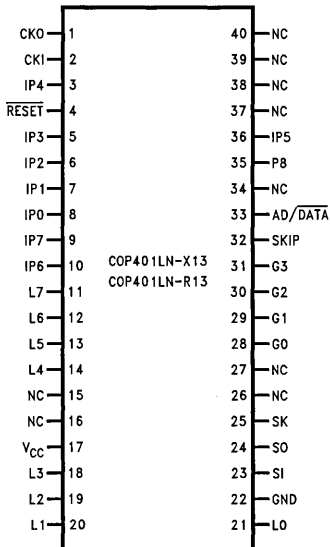
Connection Diagram

FIGURE 2

TL/DD/8528-2

Pin Descriptions

Pin	Description
L7-L0	8 bidirectional I/O ports
G3-G0	4 bidirectional I/O ports
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/DATA	Address out/data in flag
CKI	System oscillator input
CKO	System oscillator output or NC
RESET	System reset input
V _{CC}	Power supply
GND	Ground
IP7-IP0	8 bidirectional ROM address and data ports
P8	Most significant ROM address bit output
SKIP	Instruction skip output

Order Number COP401LN-X13 or COP401LN-R13
See NS Package Number N40A

Timing Waveform

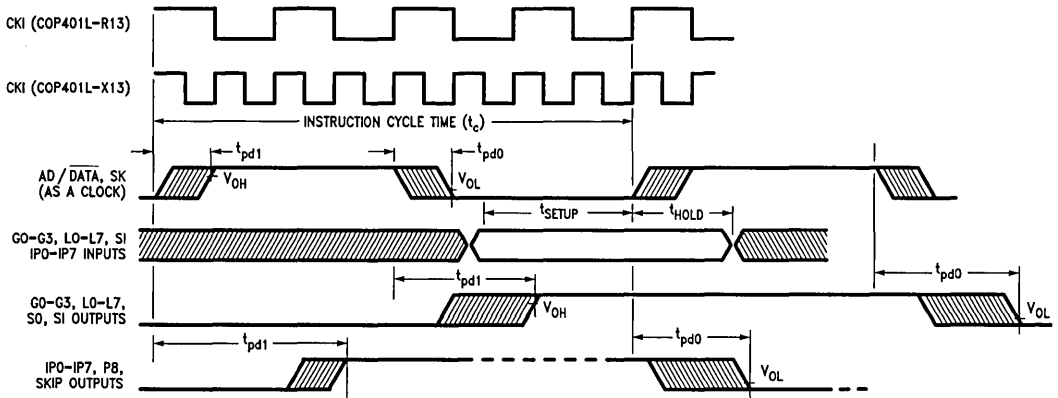


FIGURE 3. Input/Output Timing Diagram

TL/DD/8528-3

Functional Description

A block diagram of the COP401L-X13/COP401L-R13 is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

PROGRAM MEMORY

Program Memory consists of a 512-byte external memory. As can be seen by an examination of the COP401L-X13/COP401L-R13 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table 3).

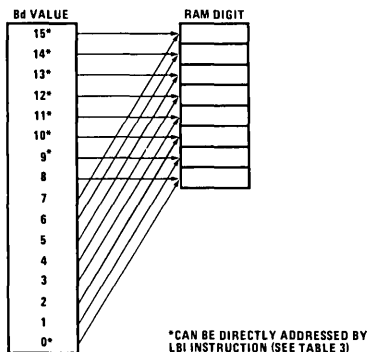


FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP401L-X13/COP401L-R13, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in-/serial-out shift register or as a binary counter depending on the contents of the EN Register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in-/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN registers (EN₃-EN₀).

1. The least significant bit of the enable register, EN₀, selects the SIO Register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI Input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO Output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN₁ is not used. It has no effect on COP401L-X13/COP401L-R13 operation.
3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high impedance input state.

TL/DD/8528-4

Functional Description (Continued)

TABLE I. Enable Register Modes - Bits EN₃ and EN₀

EN ₃	EN ₀	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

4. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN₃ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table 1 provides a summary of the modes associated with EN₃ and EN₀.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below (Figure 5). The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.

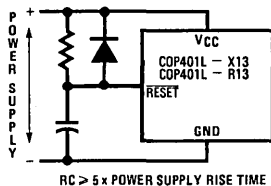


Figure 5. Power-Up Clear Circuit

TL/DD/8528-5

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP401L-X13/COP401L-R13 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE[®] outputs

3. TTL-compatible inputs

4. access time = 5 μ s max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P8 and IP7 through IP0 during the time that AD/DAT \bar{A} is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/DAT \bar{A} line; P8 is a dedicated address output, and does not need to be latched. When AD/DAT \bar{A} is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/DAT \bar{A} output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

There are two basic clock oscillator configurations available as shown by Figure 6.

- a. The COP401L-X13 is a Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.
- b. The COP401L-R13 is a RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO becomes no connection.

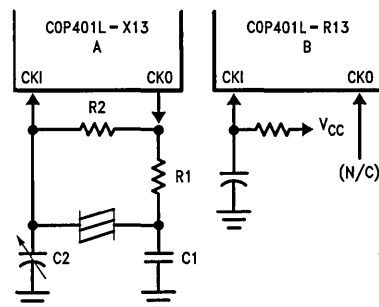
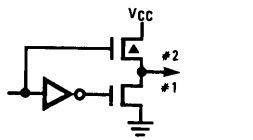


FIGURE 6. COP401L-X13/COP401L-R13 Oscillator

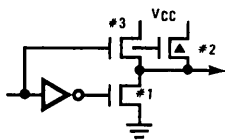
TL/DD/8528-6

Functional Description (Continued)



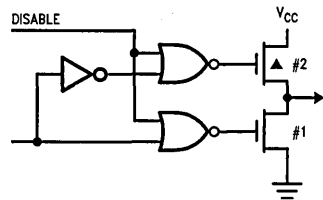
a. Standard Output

TL/DD/8528-7



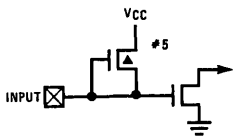
b. Push-Pull Output

TL/DD/8528-8



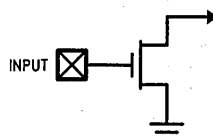
c. Standard L Output

TL/DD/8528-9



d. Input With Load

TL/DD/8528-10



e. Hi-Z Input

TL/DD/8528-11

FIGURE 7. Input and Output Configurations

Ceramic Resonator Oscillator

Resonator Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (k Ω)	C (pF)	Instruction Cycle Time (in μ s)
51	100	19 \pm 15%
82	56	19 \pm 13%

Note: 200 k Ω \geq R \geq 25 k Ω
220 pF \geq C \geq 50 pF

I/O CONFIGURATIONS

COP401L-X13/COP401L-R13 inputs and outputs have the following configurations, illustrated in *Figure 7*.

- G0–G3—an enhancement mode device to ground in conjunction with depletion-mode device to V_{CC} .
- SO, SK, IP0–IP7, P8, SKIP, AD/DATA—an enhancement mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC} . This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.
- L0–L7—same as a, but may be disabled.
- SI has on-chip depletion load device to V_{CC} .
- $\overline{\text{RESET}}$ has a Hi-Z input which must be driven to a "1" or "0" by external components.

Curves are given in *Figure 8* to allow the designer to effectively use the I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current, however, when the L lines are used as inputs, the disabled depletion device can not be relied on to source sufficient current to pull an input to a logic "1".

Typical Performance Characteristics

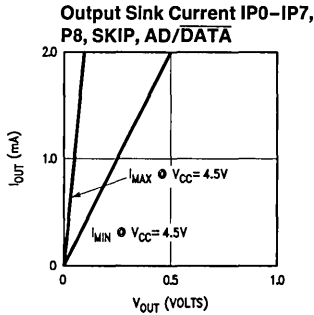
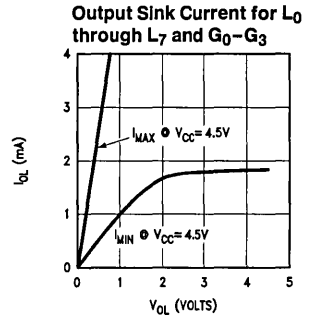
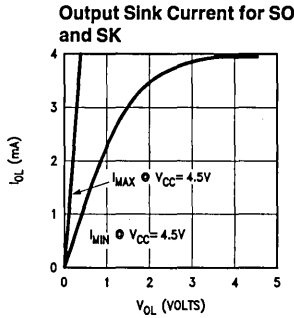
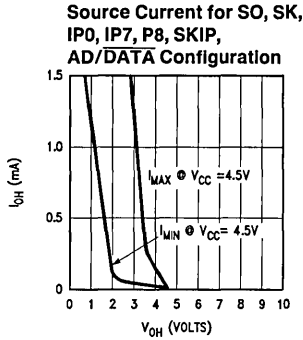
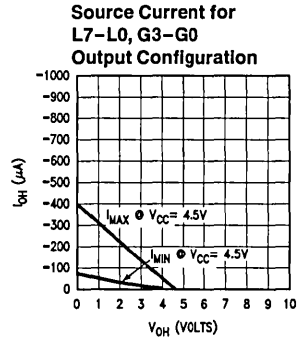
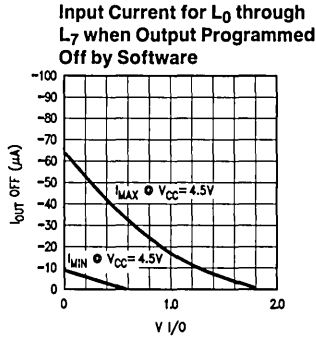
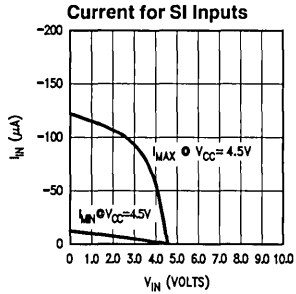


FIGURE 8. I/O Characteristics

TL/DD/8528-12

COP401L-X13/COP401L-R13 Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP401L-X13/COP401L-R13 instruction set.

TABLE II. COP401L-X13/COP401L-R13 Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic Controlled Clock Output
Instruction Operand Symbols	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
Operational Symbols	
+	Plus
–	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
\bar{A}	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP401L-X13/COP401L-R13 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
AISC	y	5-	0101 y	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	0000 0000	$0 \rightarrow A$	None	Clear A
COMP		40	0100 0000	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" $\rightarrow C$	None	Reset C
SC		22	0010 0010	"1" $\rightarrow C$	None	Set C
XOR		02	0000 0010	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	$\text{ROM}(\text{PC}_8, \text{A}, \text{M}) \rightarrow \text{PC}_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6-	0110 000 a ₈	$a \rightarrow \text{PC}$	None	Jump
JP	a	-	1 a _{6:0}	$a \rightarrow \text{PC}_{6:0}$	None	Jump within-Page (Note 3)
			(pages 2, 3 only) or			
			11 a _{5:0}	$a \rightarrow \text{PC}_{5:0}$		
			(all other pages)			
JSRP	a	-	10 a _{5:0}	$\text{PC} + 1 \rightarrow \text{SA} \rightarrow \text{SB}$	None	Jump to Subroutine Page (Note 4)
				$010 \rightarrow \text{PC}_{8:6}$ $a \rightarrow \text{PC}_{5:0}$		
JSR	a	6-	0110 100 a ₈	$\text{PC} + 1 \rightarrow \text{SA} \rightarrow \text{SB}$	None	Jump to Subroutine
			a _{7:0}	$a \rightarrow \text{PC}$		
RET		48	0100 1000	$\text{SB} \rightarrow \text{SA} \rightarrow \text{PC}$	None	Return from Subroutine
RETSK		49	0100 1001	$\text{SB} \rightarrow \text{SA} \rightarrow \text{PC}$	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	$A \rightarrow \text{Q}_{7:4}$	None	Copy A, RAM to Q
		3C	0011 1100	$\text{RAM}(B) \rightarrow \text{Q}_{3:0}$		
LD	r	-5	00 r 0101	$\text{RAM}(B) \rightarrow A$	None	Load RAM into A,
				$\text{Br} \oplus r \rightarrow \text{Br}$		Exclusive-OR Br with r
LQID		BF	1011 1111	$\text{ROM}(\text{PC}_8, \text{A}, \text{M}) \rightarrow \text{Q}$	None	Load Q Indirect (Note 2)
				$\text{SA} \rightarrow \text{SB}$		
RMB	0	4C	0100 1100	$0 \rightarrow \text{RAM}(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	$0 \rightarrow \text{RAM}(B)_1$		
	2	42	0100 0010	$0 \rightarrow \text{RAM}(B)_2$		
	3	43	0100 0011	$0 \rightarrow \text{RAM}(B)_3$		
SMB	0	4D	0100 1101	$1 \rightarrow \text{RAM}(B)_0$	None	Set RAM Bit
	1	47	0100 0111	$1 \rightarrow \text{RAM}(B)_1$		
	2	46	0100 0110	$1 \rightarrow \text{RAM}(B)_2$		
	3	4B	0100 1011	$1 \rightarrow \text{RAM}(B)_3$		
STII	y	7-	0111 y	$y \rightarrow \text{RAM}(B)$	None	Store Memory Immediate and Increment Bd
				$\text{Bd} + 1 \rightarrow \text{Bd}$		Increment Bd
X	r	-6	00 r 0110	$\text{RAM}(B) \leftrightarrow A$	None	Exchange RAM with A,
				$\text{Br} \oplus r \rightarrow \text{Br}$		Exclusive-OR Br with r

TABLE III. COP401L-X13/COP401L-R13 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XAD	3,15	23	<u>0010</u> <u>0011</u> BF <u>1011</u> <u>1111</u>	RAM(3,15) ↔ A	None	Exchange A with RAM (3,15)
XDS	r	-7	<u>00</u> <u>r</u> <u>0111</u>	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	<u>00</u> <u>r</u> <u>0100</u>	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	<u>0101</u> <u>0000</u>	A → Bd	None	Copy A to Bd
CBA		4E	<u>0100</u> <u>1110</u>	Bd → A	None	Copy Bd to A
LBI	r,d	-	<u>00</u> <u>r</u> <u>(d-1)</u> (d = 0,9;15)	r,d → B	Skip until not a LBI	Load B immediate with r,d (Note 5)
LEI	y	33 6-	<u>0011</u> <u>0011</u> <u>0110</u> <u>y</u>	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	<u>0010</u> <u>0000</u>	1st byte 2nd byte	C = "1"	Skip if C is True
SKE		21	<u>0010</u> <u>0001</u>		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	<u>0011</u> <u>0011</u>		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		21	<u>0010</u> <u>0001</u>		G ₀ = 0 G ₁ = 0 G ₂ = 0 G ₃ = 0	Skip if G Bit is Zero
	0	01	<u>0000</u> <u>0001</u>			
	1	11	<u>0001</u> <u>0001</u>			
	2	03	<u>0000</u> <u>0011</u>			
	3	13	<u>0001</u> <u>0011</u>			
SKMBZ	0	01	<u>0000</u> <u>0001</u>		RAM(B) ₀ = 0	Skip if RAM Bit is Zero
	1	11	<u>0001</u> <u>0001</u>		RAM(B) ₁ = 0	
	2	03	<u>0000</u> <u>0011</u>	RAM(B) ₂ = 0		
	3	13	<u>0001</u> <u>0011</u>	RAM(B) ₃ = 0		
INPUT/OUTPUT INSTRUCTIONS						
ING		33	<u>0011</u> <u>0011</u>	G → A	None	Input G Ports to A
		2A	<u>0010</u> <u>1010</u>			
INL		33	<u>0011</u> <u>0011</u>	L _{7:4} → RAM(B)	None	Input L Ports to RAM, A
		2E	<u>0010</u> <u>1110</u>	L _{3:0} → A		
OMG		33	<u>0011</u> <u>0011</u>	RAM(B) → G	None	Output RAM to G Ports
		3A	<u>0011</u> <u>1010</u>			
XAS		4F	<u>0100</u> <u>1111</u>	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined) Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1* e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP401L-X13/COP401L-R13 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC₈, A, M. PC₈ is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC₈, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → PC_{7:4}, RAM (B) → PC_{3:0}, leaving PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

- The first word of a COP401L-X13/COP401L-R13 program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

- The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or will access data in the next group of 4 pages.

COPS Programming Manual

For detailed information on writing COPS programs, the COPS Programming Manual 424410284-001 provides an in-depth discussion of the COPS architecture, instruction set and general techniques of COPS programming. This manual is written with the programmer in mind.

Typical Applications

PROM-Based System

The COP401L-X13/COP401L-R13 may be used to emulate the COP413L. *Figure 9* shows the interconnect to implement a COP401L-X13/COP401L-R13 hardware emulation. This connection uses one MM2716 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP₇-IP₀ are bidirectional inputs and outputs. When the AD/ \overline{DATA} clocking output turns on, the EPROM drivers are disabled and IP₇-IP₀ output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

When AD/ \overline{DATA} turns off, the EPROM is enabled and the IP₇-IP₀ pins will input the memory data. P8 outputs the most significant address bit to the memory. (SKIP output may be used for program debug if needed.)

Twenty of the COP401L-X13/COP401L-R13 pins may be configured exactly the same as the COP413L. Selection of the COP401L-X13 or COP401L-R13 depends upon which oscillator is selected for the COP413L.

Oscillator Requirement

	Order ROMless
COP413L Option 1 = 0	Ceramic Resonator or external input frequency divided by 8. CKO is oscillator out.
Option 1 = 1	Single Pin RC controlled oscillator divided by 4. CKO is no connection.

COP401L-R13

Typical Applications (Continued)

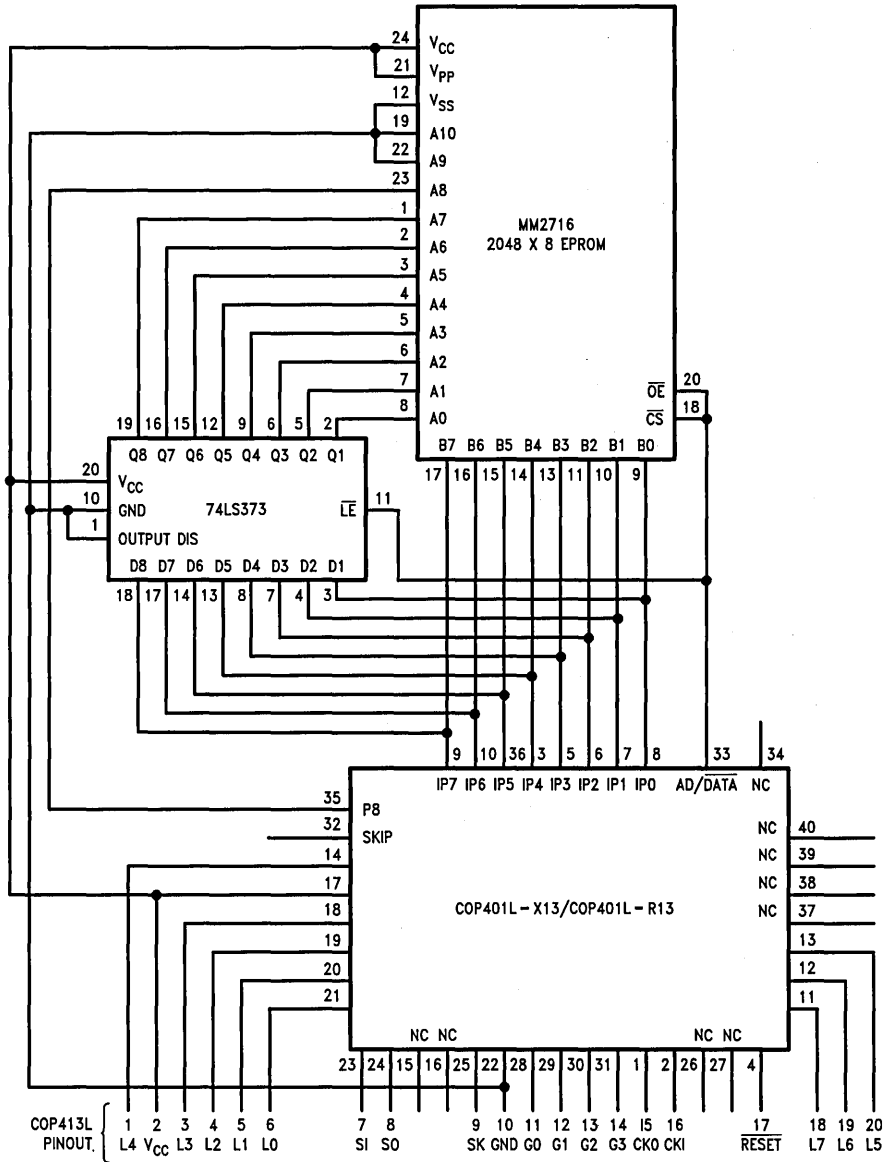


FIGURE 9. COP401L-X13/COP401L-R13 Used to Emulate a COP413L

TL/DD/8528-13

COP402-5 ROMless N-Channel Microcontroller

General Description

The COP402-5 ROMless Microcontroller is a member of the COPSTM family, fabricated using N-channel silicon gate MOS technology. The part contains CPU, RAM, and I/O, and is identical to a COP420 device, except the ROM has been removed; pins have been added to output the ROM address and to input ROM data. In a system, the COP402 will perform exactly as the COP420; this important benefit facilitates development and debug of a COP420; this important benefit facilitates development and debug of a COP420 program prior to masking the final part. The device is also appropriate in low volume applications, or when the program may require changing.

Features

- Extended temperature (-40°C to +85°C) COP302, available as special order
- Low cost
- Exact circuit equivalent of COP420
- Standard 40-pin dual-in-line package
- Interfaces with standard PROM or ROM
- 64 x 4 RAM, addresses up to 1k x 8 ROM
- Powerful instruction set
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 4.0 μs instruction time
- Single supply operation (4.5V to 6.3V)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ serial I/O capability
- Software/hardware compatible with other members of COP400 family

Block Diagram

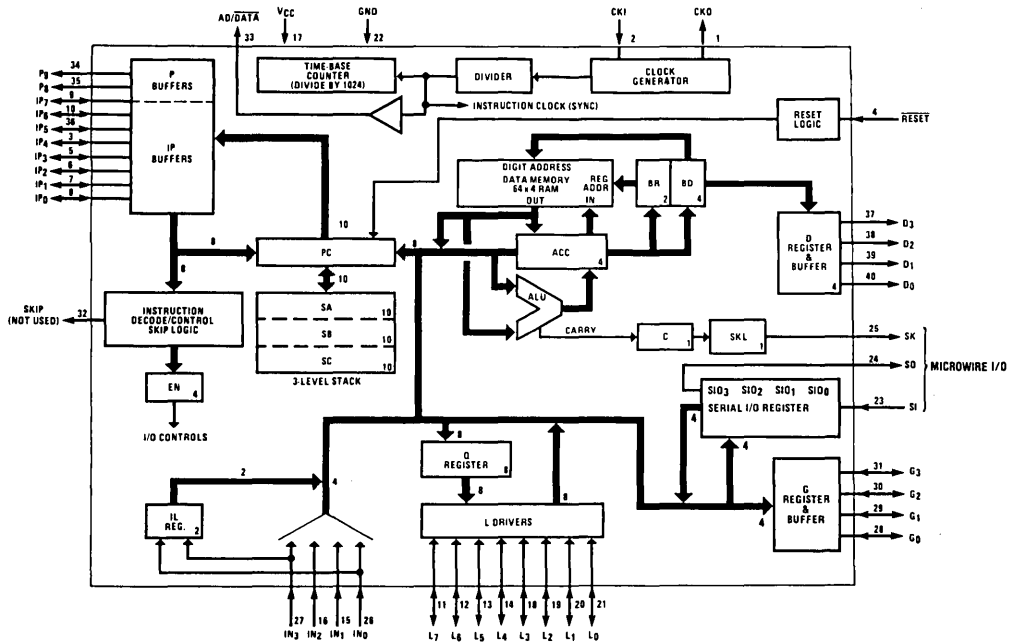


FIGURE 1

TL/DD/6915-1

COP402 and COP302

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin	-0.3V to +7V
Operating Temperature Range	0°C to 70°C
COP402-5	0°C to 70°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (soldering, 10 sec.)	300°C

Package Power Dissipation	750 mW at 25°C
	400 mW at 70°C
	250 mW at 85°C

Total Sink Current	50 mA
Total Source Current	70 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP402-5

DC Electrical Characteristics 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	5.5	V
Power Supply Ripple (Notes 2, 3)	Peak to Peak		0.4	V
Supply Current	All Outputs Open V _{CC} = 5V		40	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High		2.4		V
Logic Low		-0.3	0.4	V
Schmitt Trigger Input				
RESET				
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	V _{CC} = 5V ± 5%	2.0		V
Logic Low		-0.3	0.8	V
Input Load Source Current	V _{CC} = 5V, V _{IN} = 0V	-100	-800	μA
Input Capacitance (Note 2)			7	pF
Hi-Z Input Leakage	V _{CC} = 5V	-1	+1	μA
Output Voltage Levels				
D, G, L, SK, SO Outputs				
TTL Operation	V _{CC} = 5V ± 10%			
Logic High	I _{OH} = -100 μA	2.4		V
Logic Low	I _{OL} = 1.6 mA	-0.3	0.4	V
IP0-IP7, P8, P9, SKIP, CKO, AD/DATA				
Logic High	I _{OH} = -75 μA	2.4		V
Logic Low	I _{OL} = 400 μA	-0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = 10 μA	-0.3	0.2	V
Output Current Levels				
LED Direct Drive (Note 1)				
Logic High	V _{CC} = 5.5V V _{OH} = 2.0V	1.5	14	mA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			-15	mA
Per Pin (All Others)			-1.5	mA

COP402**AC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$, $4.5\text{V} \leq V_{\text{CC}} \leq 5.5\text{V}$ unless otherwise noted

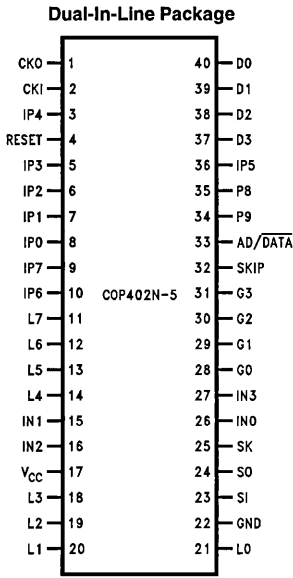
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	$\div 16$ Mode	1.6	4.0	MHz
CKI Duty Cycle		40	60	%
Rise Time (Note 3)	Frequency = 4 MHz		60	ns
Fall Time (Note 3)	Frequency = 4 MHz		40	ns
Inputs:				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
Output Propagation Delay	Test Conditions: $R_L = 5\text{k}$, $C_L = 50\text{ pF}$, $V_{\text{OUT}} = 1.5\text{V}$			
SO and SK				
t_{pd1}			1.0	μs
t_{pd0}			1.0	μs
CKO				
t_{pd1}			0.25	μs
t_{pd0}			0.25	μs
AD/ $\overline{\text{DATA}}$, SKIP				
t_{pd1}			0.6	μs
t_{pd0}			0.6	μs
All Other Outputs				
t_{pd1}			1.4	μs
t_{pd0}			1.4	μs

Note 1: TRI-STATE® and LED configurations are excluded.

Note 2: Voltage change must be less than 0.5V in a 1 ms period.

Note 3: This parameter is only sampled and not 100% tested.

Connection Diagram



TL/DD/6915-2

Top View

Order Number COP402N-5
See NS Package Number N40A

FIGURE 2

Pin Descriptions

Pin	Description
L7-L0	8 bidirectional I/O ports with TRI-STATE
G3-G0	4 bidirectional I/O ports
D3-D0	4 general purpose outputs
IN3-IN0	4 general purpose inputs
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/DATA	Address out/data in flag
SKIP	Instruction skip output
CKI	System oscillator input
CKO	System oscillator output
RESET	System reset input
V _{CC}	Power supply
GND	Ground
IP7-IP0	8 bidirectional ROM address and data ports
P8, P9	2 most significant ROM address outputs

Timing Diagrams

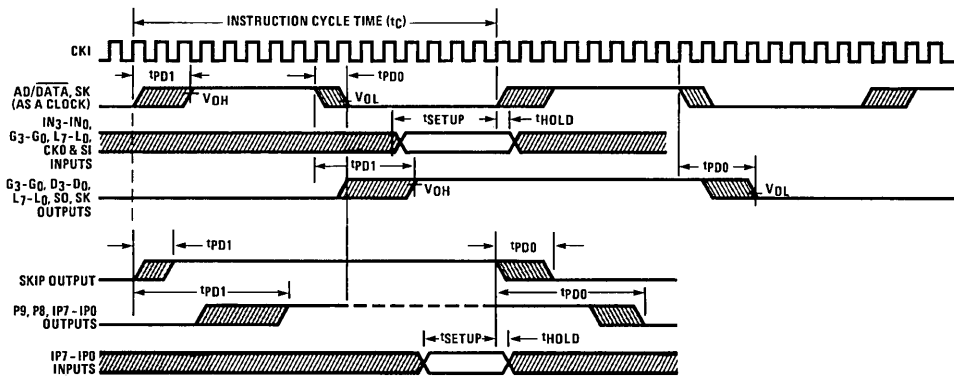


FIGURE 3a. Input/Output Timing Diagrams (Crystal ÷ 16 Mode)

TL/DD/6915-3

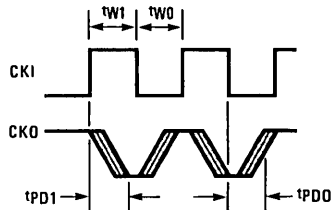


FIGURE 3b. CKO Output Timing

TL/DD/6915-4

Functional Description

A block diagram of the COP402 is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024-byte external memory (typically PROM). Words of this memory may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential **10-bit binary count** value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instruction based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit **A register** (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A **4-bit adder** performs the arithmetic and logic functions of the COP402, storing its results in A. It also outputs a carry bit to the 1-bit **C register**, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description.)

Four **general-purpose inputs**, IN_3 – IN_0 , are provided; IN_1 , IN_2 , and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select and Write Strobe inputs, respectively, for use in MICROBUS applications.

The **D register** provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The **G register** contents are outputs to 4 general-purpose bidirectional I/O ports. G_0 may be mask-programmed as a "ready" output for MICROBUS applications.

The **Q register** is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The **8 L drivers**, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The **SIO register** functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The **XAS instruction** copies C into the SKL latch. In the counter mode, SK is the output of SKL. In the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The **EN register** is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3 – EN_0).

1. The least significant bit of the enable register, EN_0 , selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3 . With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.

Functional Description (Continued)

3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3 , in conjunction with EN_0 , affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3 . With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0 .

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address ($PC + 1$) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level ($PC + 1 \rightarrow SA \rightarrow SB \rightarrow SC$). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.
- b. An interrupt will be acknowledged only after the following conditions are met:
 1. EN_1 has been set.
 2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
 3. A currently executing instruction has been completed.
 4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).
- c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon the popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt-routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the

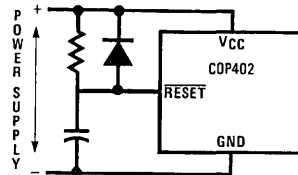
original ASC. At *this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and the LQID instruction should not be nested within the interrupt servicing routine since their popping of the stack enables any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

- d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.
- e. An LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below. The RESET pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC} . Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least two instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, G, and SO are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



$$RC \geq 5 \times \text{Power Supply Rise Time} \quad \text{TL/DD/6915-8}$$

FIGURE 4. Power-Up Clear Circuit

OSCILLATOR

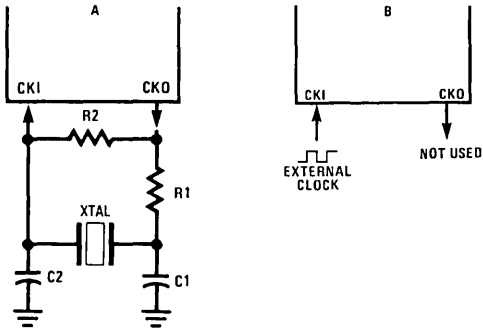
There are two basic clock oscillator configurations available as shown by Figure 5.

- a. **Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16.
- b. **External Oscillator.** CKI is driven by an external clock signal. The instruction cycle time is the clock frequency divided by 16.

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = SYNC If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = SYNC If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)



TL/DD/6915-9

Crystal Value	Component Values			
	R1	R2	C1 (pF)	C2 (pF)
4 MHz	4.7k	1M	22	22
3.58 MHz	3.3k	1M	22	27
2.09 MHz	8.2k	1M	47	33

FIGURE 5. COP402 Oscillator

EXTERNAL MEMORY INTERFACE

The COP402 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL = compatible inputs
4. access time = 1.0 μ s, max.

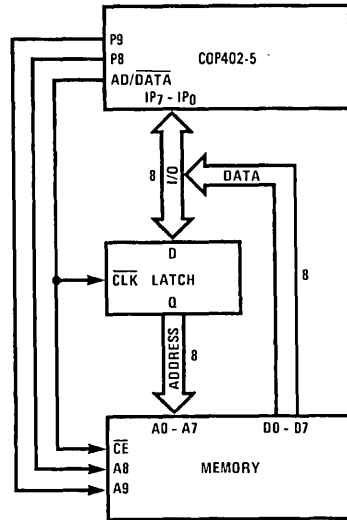
Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P9, P8, and IP7 through IP0 during the time that AD/DAT \bar{A} is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/DAT \bar{A} line; P9 and P8 are dedicated address outputs, and do not need to be latched. When AD/DAT \bar{A} is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/DAT \bar{A} output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input. A simplified block diagram of the external memory interface is shown in Figure 6.

INPUT/OUTPUT

COP402 outputs have the following configurations, illustrated in Figure 7.

- a. Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements.



TL/DD/6915-10

FIGURE 6. External Memory Interface to COP402

- a. High Drive**—same as a. except greater current sourcing capability.
- b. Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.
- c. LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display.
- d. TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers.
- f. Input with Load**—Inputs have an on-chip depletion load device to V_{CC}, as shown in Figure 7f.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1-6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in Figure 7 for each of these devices.

The SO, SK outputs are configured as shown in Figure 7c. The D and G outputs are configured as shown in Figure 7a.

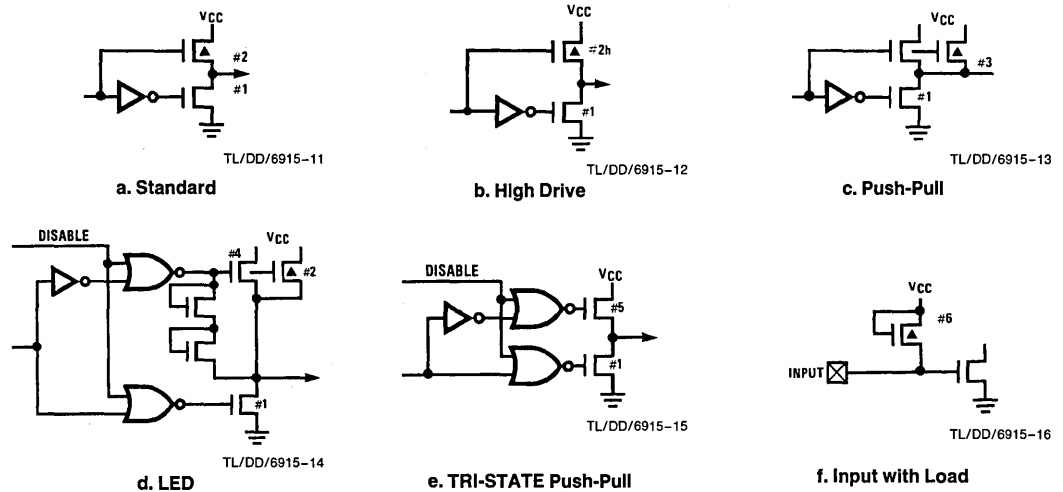
Functional Description (Continued)

Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs are configured as in Figure 7d on the COP402.

An important point to remember if using configuration d with the L drivers is that even when the L drivers are disabled,

the depletion load device will source a small amount of current. (See Figure 8.)

IP7 through IP0 outputs are configured as shown in Figure 7c; P9, P8, SKIP, and AD/DATA are configured as shown in Figure 8b.



(▲ is Depletion Device)

FIGURE 7. Input/Output Configurations

Typical Performance Characteristics

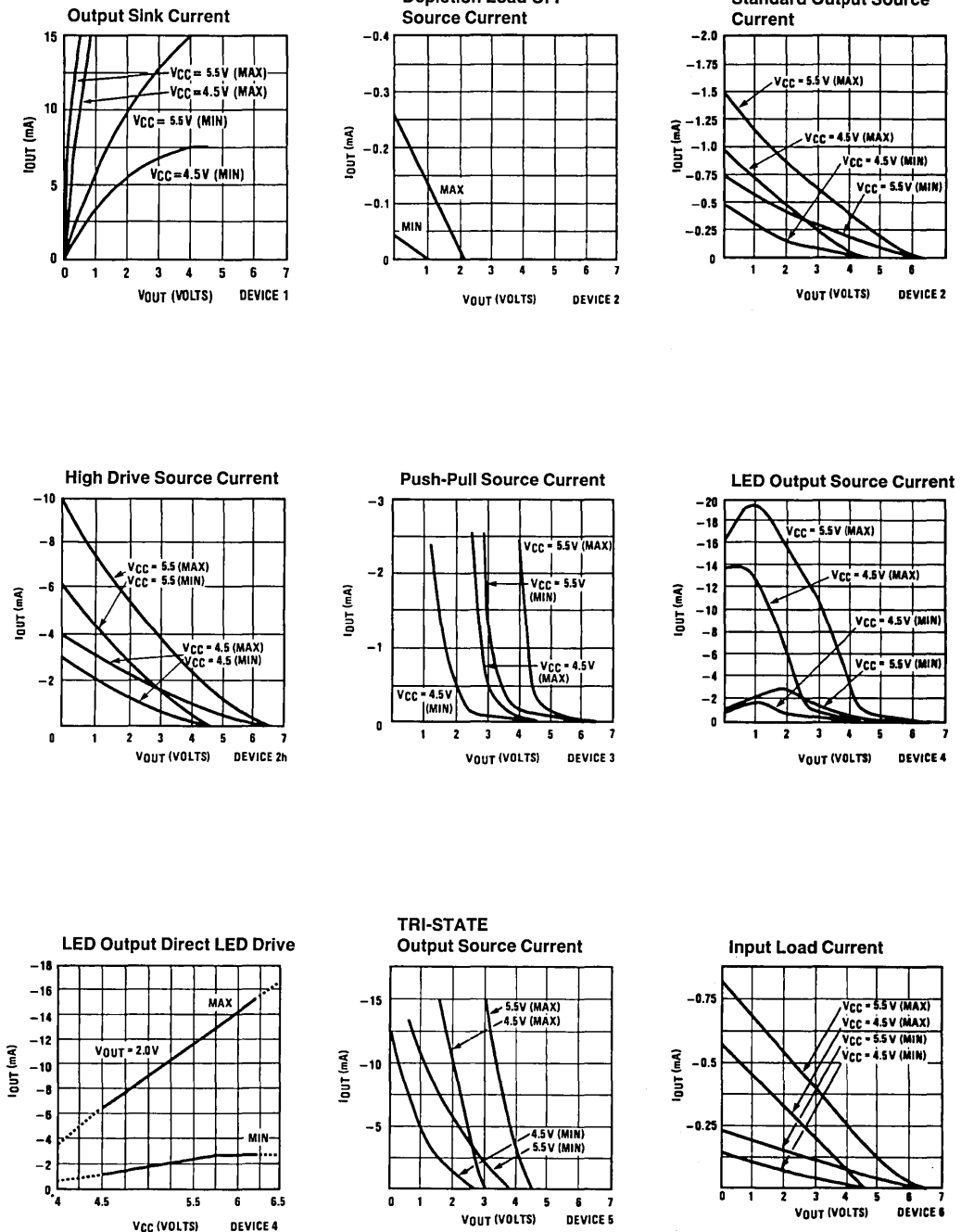


FIGURE 8. COP402 Input/Output Characteristics

TL/DD/6915-17

Typical Performance Characteristics (Continued)

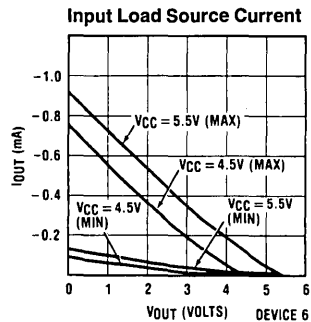
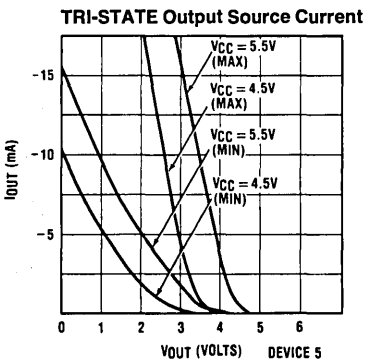
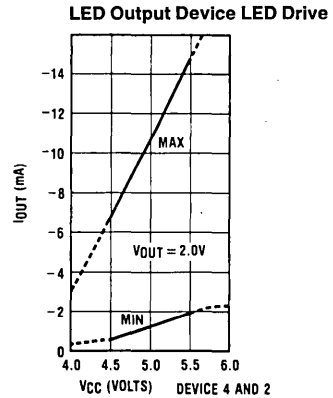
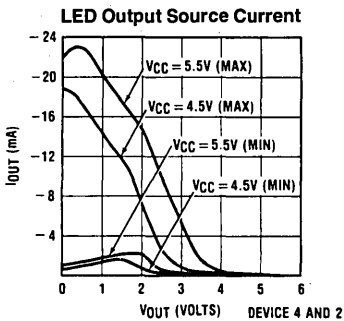
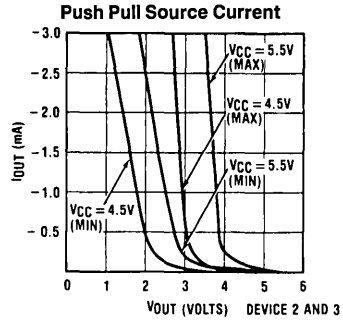
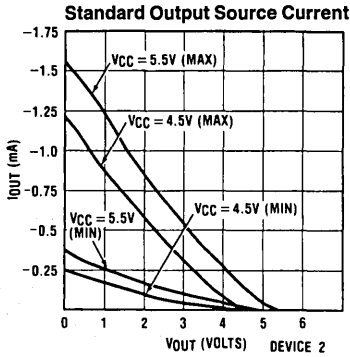
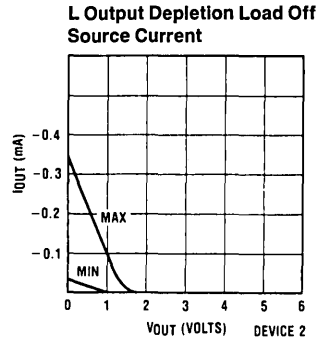
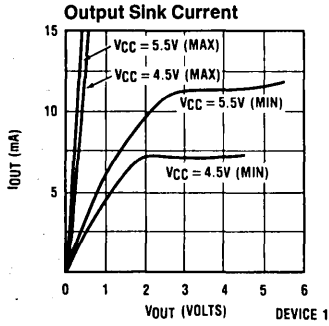


FIGURE 8a. COP302 Input/Output Characteristics

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP402 instruction set.

TABLE II. COP402 Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
B	6-bit RAM Address Register	r	2-bit Operand Field, 0–3 binary (RAM Register Select)
Br	Upper 2 bits of B (register address)	a	9-bit Operand Field, 0–511 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0–15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register	OPERATIONAL SYMBOLS	
G	4-bit Register to latch data for G I/O Port	+	Plus
IL	Two 1-bit Latches Associated with the IN ₃ or IN ₀ inputs	–	Minus
IN	4-bit Input port	→	Replaces
L	8-bit TRI-STATE I/O Port	↔	Is exchanged with
M	4-bit contents of RAM Memory pointed to by B Register	=	Is equal to
P	2-bit ROM Address Port	\bar{A}	The one's complement of A
PC	10-bit ROM Address Register (program counter)	⊕	Exclusive-OR
Q	8-bit Register to latch data for L I/O Port	:	Range of values
SA	10-bit Subroutine Save Register A		
SB	10-bit Subroutine Save Register B		
SC	10-bit Subroutine Save Register C		
SIO	4-bit Shift Register and Counter		
SK	Logic-Controlled Clock Output		

TABLE III. COP402 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
ADT		4A	<u>0100</u> <u>1010</u>	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5–	<u>0101</u> <u>y</u>	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	<u>0001</u> <u>0000</u>	$\bar{A} + \text{RAM}(B) + C \rightarrow A$ Carry $\rightarrow C$	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" $\rightarrow C$	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" $\rightarrow C$	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP402 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC _{9:8} , A, M) → PC _{7:0}	None	Jump Indirect (Note 3)
JMP	a	6--	0110 00 a _{9:8} a _{7:0}	a → PC	None	Jump
JP	a	--	1 a _{6:0} (pages 2,3 only) or 11 a _{5:0} (all other pages)	a → PC _{6:0} a → PC _{5:0}	None	Jump within Page (Note 4)
JSRP	a	--	10 a _{5:0}	PC + 1 → SA → SB → SC 0010 → PC _{9:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 5)
JSR	a	6--	0110 10 a _{9:8} a _{7:0}	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → Q _{7:4} RAM(B) → Q _{3:0}	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	Q _{7:4} → RAM(B) Q _{3:0} → A	None	Copy Q to RAM, A
LD	r	-5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 --	0010 0011 00 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM(PC _{9:8} , A, M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → RAM(B) ₀ 0 → RAM(B) ₁ 0 → RAM(B) ₂ 0 → RAM(B) ₃	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → RAM(B) ₀ 1 → RAM(B) ₁ 1 → RAM(B) ₂ 1 → RAM(B) ₃	None	Set RAM Bit
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 0011 10 r d	RAM(r,d) ↔ A	None	Exchange A with RAM pointed to directly by r,d

Instruction Set (Continued)

TABLE III. COP402 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	00 r 0111	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	00 r 0100	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	--	00 r (d-1) (d=0,9:15) or 33 10 r d (any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A ↔ Br (0,0 → A ₃ ,A ₂)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ	0 1 2 3	33 01 11 03 13	0011 0011 0000 0001 0001 0001 0000 0011 0001 0011	1st byte 2nd byte	G ₀ = 0 G ₁ = 0 G ₂ = 0 G ₃ = 0	Skip if G Bit is Zero
SKMBZ	0 1 2 3	01 11 03 13	0000 0001 0001 0001 0000 0011 0001 0011		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)



Instruction Set (Continued)

TABLE III. COP402 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description				
INPUT/OUTPUT INSTRUCTIONS										
ING		33 2A	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1010</td></tr></table>	0011	0011	0010	1010	G → A	None	Input G Ports to A
0011	0011									
0010	1010									
ININ		33 28	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1000</td></tr></table>	0011	0011	0010	1000	IN → A	None	Input IN Inputs to A (Note 2)
0011	0011									
0010	1000									
INIL		33 29	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1001</td></tr></table>	0011	0011	0010	1001	IL ₃ , "0", IL ₀ → A	None	Input IL Latches to A (Note 3)
0011	0011									
0010	1001									
INL		33 2E	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0010</td><td>1110</td></tr></table>	0011	0011	0010	1110	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM,A
0011	0011									
0010	1110									
OBD		33 3E	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0011</td><td>1110</td></tr></table>	0011	0011	0011	1110	Bd → D	None	Output Bd to D Outputs
0011	0011									
0011	1110									
OGI	y	33 5-	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0101</td><td>y</td></tr></table>	0011	0011	0101	y	y → G	None	Output to G Ports Immediate
0011	0011									
0101	y									
OMG		33 3A	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0011</td><td>1010</td></tr></table>	0011	0011	0011	1010	RAM(B) → G	None	Output RAM to G Ports
0011	0011									
0011	1010									
XAS		4F	<table border="1"><tr><td>0100</td><td>1111</td></tr></table>	0100	1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 3)		
0100	1111									

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit register.

Note 2: The ININ instruction is not available on the 24-pin COP421 since this device does not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register.

The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

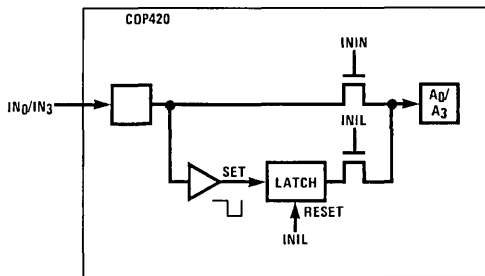
JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 10-bit word, PC_{9:8}, A, M. PC₉ and PC₈ are not affected by this instruction.

Note that JID requires 2 instruction cycles.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL₃ and IL₀ (see Figure 9) and CKO into A. The IL₃ and IL₀ latches are set if a low-going pulse ("1" to "0") has occurred on the IN₃ and IN₀ inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL₃ and IN₀ into A₃ and A₀ respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN₃ and IN₀ lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A₂. If CKO has not been so programmed, a "1" will be placed in A₂. A "0" is always placed in A₁ upon the execution of an INIL. The general purpose inputs IN₃–IN₀ are input to A upon the execution of an ININ instruction. (See Table III, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.



TL/DD/6915-19

FIGURE 9. IN₀/IN₃ Latches

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC₉, PC₈, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → PC_{7:4}, RAM(B) → PC_{3:0}, leaving PC₉ and PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip on Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the controller to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 131 kHz (crystal frequency ÷ 16) and the binary counter output pulse frequency will be 128 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 128 ticks.

INSTRUCTION SET NOTES

- The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed, except JID and LQID. LQID and JID take two cycle times if executed and one if skipped.
- The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, or 15 will access data in the next group of 4 pages.

Typical Application: PROM-Based System

The COP402 may be used to exactly emulate the COP420, *Figure 10* shows the interconnect to implement a COP420 hardware emulation. This connection uses two MM5204 EPROMs as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7-IP0 are bidirectional inputs and outputs. When the AD/DATA clocking output turns on, the EPROM drivers are disabled and IP7-IP0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

When AD/DATA turns off, the EPROMs are enabled and the IP7-IP0 pins will input the memory data. P8 and P9 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

The other 28 pins of the COP402 may be configured exactly the same as a COP420.

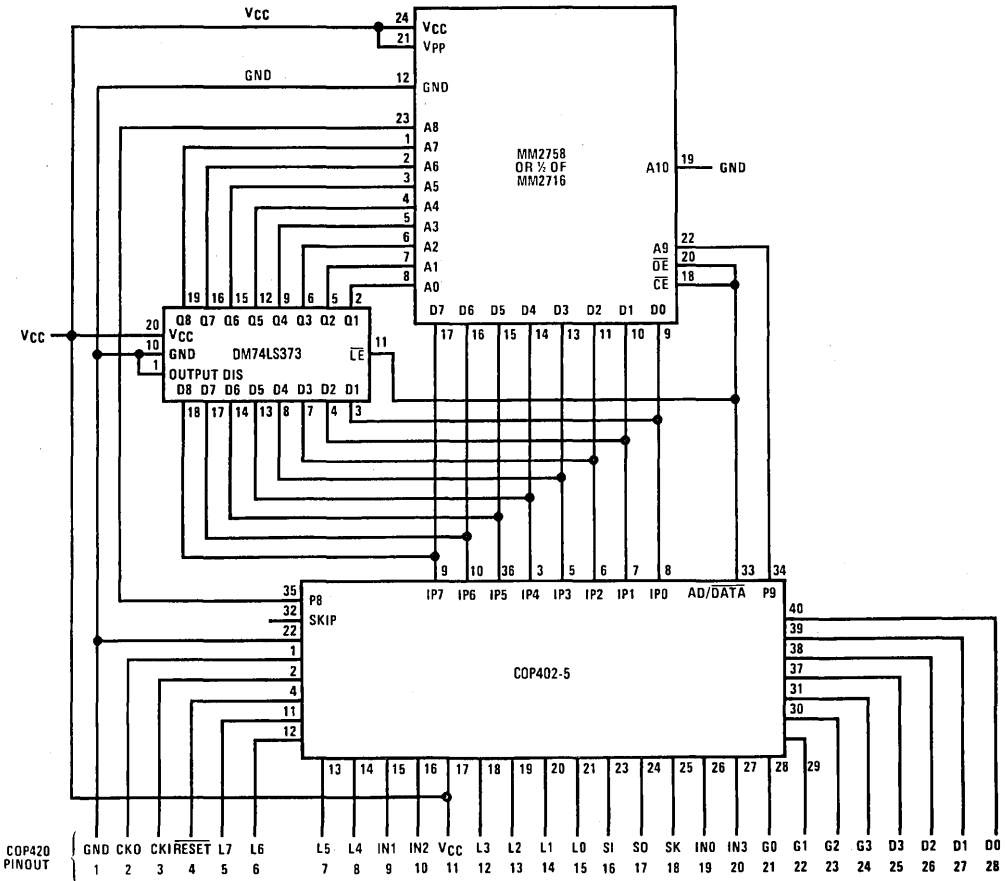


FIGURE 10. COP402 Used to Emulate a COP420

TL/DD/6915-20

Option List

COP402 MASK OPTIONS

The following COP420 options have been implemented in this basic version of the COP402. Subsequent versions of the COP402 will implement different combinations of available options; such versions will be identified as COP402-A, COP402-B, etc.

Option Value	Comment	Option Value	Comment
Option 1 = 0	Ground Pin—no option available	Option 15 = 2, 3	L0 same as L7
Option 2 = 0	CKO is clock generator output to crystal	Option 16 = 0	SI has load device to V_{CC}
Option 3 = 0	CKI is crystal input $\div 16$ (may be overridden externally)	Option 17 = 2	SO has push-pull output
Option 4 = 0	RESET pin has load device to V_{CC}	Option 18 = 2	SK has push-pull output
Option 5 = 2 (402)	L7 has LED direct-drive output	Option 19 = 0	IN0 has load device to V_{CC}
Option 6 = 2, 3	L6 same as L7	Option 20 = 0 (402)	IN3 has load device to V_{CC}
Option 7 = 2, 3	L5 same as L7	Option 21 = 0	G0 has standard output
Option 8 = 2, 3	L4 same as L7	Option 22 = 0	G1 same as G0
Option 9 = 0 (402)	IN1 has load device to V_{CC}	Option 23 = 0	G2 same as G0
Option 10 = 0 (402)	IN2 has load device to V_{CC}	Option 24 = 0	G3 same as G0
Option 11 = 0	V_{CC} pin—no option available	Option 25 = 0	D3 has standard output
Option 12 = 2, 3	L3 same as L7	Option 26 = 0	D2 same as D3
Option 13 = 2, 3	L2 same as L7	Option 27 = 0	D1 same as D3
Option 14 = 2, 3	L1 same as L7	Option 28 = 0	D0 same as D3
		Option 29 = 0 (402)	normal operation
		Option 30 = N/A	40-pin package



COP404C ROMless CMOS Microcontrollers

General Description

The COP404C ROMless Microcontroller is a member of the COPSTM family, fabricated using double-poly, silicon gate CMOS (microCMOS) technology. The COP404C contains CPU, RAM, I/O and is identical to a COP444C device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. The COP404C can be configured, by means of external pins, to function as a COP444C, a COP424C, or a COP410C. Pins have been added to allow the user to select the various functional options that are available on the family of mask-programmed CMOS parts. The COP404C is primarily intended for use in the development and debug of a COP program for the COP444C/445C, COP424C/425C, and COP410C/411C devices prior to masking the final part. The COP404C is also appropriate in low volume applications or when the program might be changing.

Features

- Accurate emulation of the COP444C, COP424C and COP410C
- Lowest Power Dissipation (50 μ W typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4 μ s instruction time, plus software selectable clocks
- 128 \times 4 RAM, addresses 2k \times 8 ROM
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (2.4V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible
- MICROBUS™ compatible
- Software/hardware compatible with other members of the COP400 family

Block Diagram

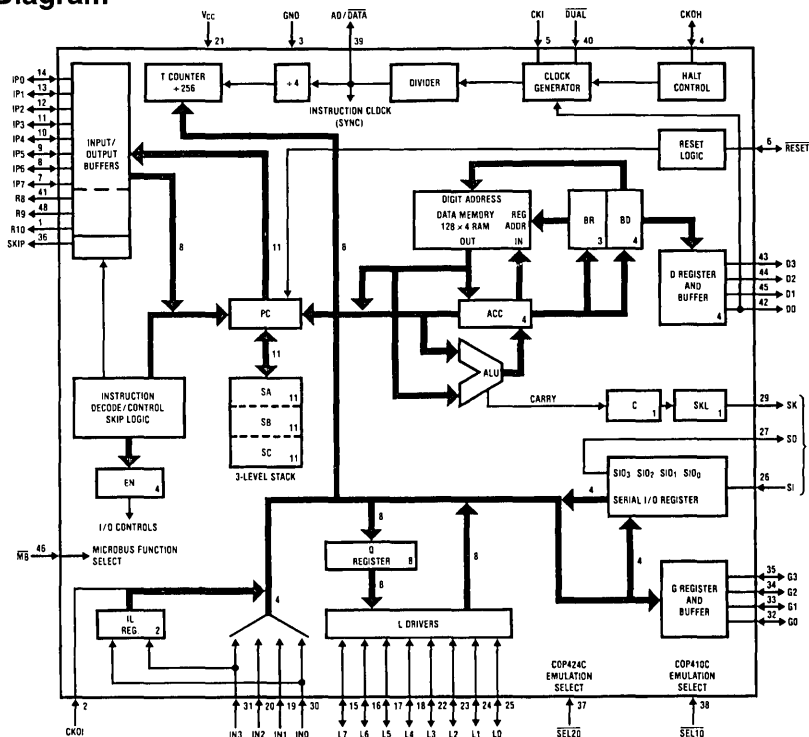


FIGURE 1. Block Diagram

TL/DD/5530-1

Absolute Maximum Ratings

Supply Voltage	6V	Operating temperature range	0° to +70°C
Voltage at any pin	-0.3V to $V_{CC} + 0.3V$	Storage temperature range	-65°C to +150°C
Total Allowable Source Current	25 mA	Lead temperature (soldering, 10 sec.)	300°C
Total Allowable Sink Current	25 mA		

DC Electrical Characteristics

0°C ≤ T_a ≤ 70°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple (Notes 4, 5)	peak to peak		0.1 V _{CC}	V
Supply Current (Note 1)	V _{CC} = 2.4V, t _c = 64 μs V _{CC} = 5.0V, t _c = 16 μs V _{CC} = 5.0V, t _c = 4 μs (T _c is instruction cycle time)		120 700 3000	μA μA μA
HALT Mode Current (Note 2)	V _{CC} = 5.0V, F _{IN} = 0 kHz, T _A = 25°C V _{CC} = 2.4V, F _{IN} = 0 kHz, T _A = 25°C		20 6	μA μA
Input Voltage Levels RESET, D0 (clock input) CKI				
Logic High		0.9 V _{CC}		V
Logic Low			0.1 V _{CC}	V
All other inputs (Note 7)				
Logic High		0.7 V _{CC}		V
Logic Low			0.2 V _{CC}	V
Input Pull-up current	V _{CC} = 4.5V, V _{IN} = 0	30	330	μA
Hi-Z input leakage		-1	+1	μA
Input capacitance (Note 4)			7	pF
Output Voltage Levels LSTTL Operation	Standard outputs V _{CC} = 5.0V ±10%			
Logic High	I _{OH} = -100 μA	2.7		V
Logic Low	I _{OL} = 400 μA		0.4	V
CMOS Operation				
Logic High	I _{OH} = -10 μA	V _{CC} - 0.2		V
Logic Low	I _{OL} = 10 μA		0.2	V
Output current levels Sink (Note 6)	V _{CC} = 4.5V, V _{OUT} = V _{CC} V _{CC} = 2.4V, V _{OUT} = V _{CC}	1.2 0.2		mA mA
Source (Standard option)	V _{CC} = 4.5V, V _{OUT} = 0V V _{CC} = 2.4V, V _{OUT} = 0V	0.5 0.1		mA mA
Source (Low current option)	V _{CC} = 4.5V, V _{OUT} = 0V V _{CC} = 2.4V, V _{OUT} = 0V	30 6	330 80	μA μA
Allowable Sink/Source current per pin (Note 6)			5	mA
Allowable Loading on CKOH			100	pF
Current needed to over-ride HALT (Note 3)				
To continue	V _{CC} = 4.5V, V _{IN} = 2V _{CC}		.7	mA
To halt	V _{CC} = 4.5V, V _{IN} = 7V _{CC}		1.6	mA
TRI-STATE leakage current		-2.5	+2.5	μA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP404C

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq 70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Max	Units									
Instruction Cycle Time (t_c)	$V_{CC} \geq 4.5\text{V}$	4	DC	μs									
	$4.5\text{V} > V_{CC} \geq 2.4\text{V}$	16	DC	μs									
Operating CKI Frequency	$V_{CC} \geq 4.5\text{V}$	DC	1.0	MHz									
	$4.5\text{V} > V_{CC} \geq 2.4\text{V}$	DC	250	kHz									
Duty Cycle (Note 4)	$f_1 = 4\text{ MHz}$	40	60	%									
Rise Time (Note 4)	$f_1 = 4\text{ MHz}$ external clock		60	ns									
Fall Time (Note 4)			40	ns									
Instruction Cycle Time using D0 as a RC Oscillator Dual-Clock Input (Note 4)	$R = 30\text{k}$, $V_{CC} = 5\text{V}$ $C = 82\text{ pF}$	8	16	μs									
INPUTS: (See Fig. 3) t_{SETUP}	<table style="border: none;"> <tr> <td style="border: none;">G Inputs</td> <td rowspan="5" style="border: none; vertical-align: middle;">} $V_{CC} \geq 4.5\text{V}$</td> <td rowspan="5" style="border: none; vertical-align: middle;">$T_c/4 + .7$</td> <td rowspan="5" style="border: none;"></td> <td rowspan="5" style="border: none;">μs</td> </tr> <tr> <td style="border: none;">SI Input</td> </tr> <tr> <td style="border: none;">IP Input</td> </tr> <tr> <td style="border: none;">All Others</td> </tr> <tr> <td style="border: none;">$V_{CC} \geq 4.5\text{V}$</td> </tr> </table>	G Inputs	} $V_{CC} \geq 4.5\text{V}$	$T_c/4 + .7$		μs	SI Input	IP Input	All Others	$V_{CC} \geq 4.5\text{V}$			
G Inputs		} $V_{CC} \geq 4.5\text{V}$					$T_c/4 + .7$		μs				
SI Input													
IP Input													
All Others													
$V_{CC} \geq 4.5\text{V}$													
t_{HOLD}	$4.5\text{V} > V_{CC} \geq 2.4\text{V}$	1.0		μs									
OUTPUT PROPAGATION DELAY	$V_{\text{OUT}} = 1.5\text{V}$, $C_L = 100\text{ pF}$, $R_L = 5\text{K}$												
IP7-IP0, A10-A8, SKIP t_{PD1} , t_{PD0}	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$		1.94	μs									
			7.75	μs									
AD/DATA t_{PD1} , t_{PD0}	$V_{CC} \geq 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$		375	ns									
			1.5	μs									
ALL OTHER OUTPUTS t_{PD1} , t_{PD0}	$V_{CC} > 4.5\text{V}$ $4.5\text{V} > V_{CC} \geq 2.4\text{V}$		1.0	μs									
			4.0	μs									
MICROBUS TIMING Read Operation (Fig. 4)	$C_L = 50\text{ pF}$, $V_{CC} = 5\text{V} \pm 5\%$												
Chip select stable before $\overline{\text{RD}}$ - t_{CSR}		65		ns									
Chip select hold time for $\overline{\text{RD}}$ - t_{RCS}		20		ns									
$\overline{\text{RD}}$ pulse width - t_{PR}		400		ns									
Data delay from $\overline{\text{RD}}$ - t_{RD}			375	ns									
$\overline{\text{RD}}$ to data floating - t_{DF} (Note 4)			250	ns									
Write Operation (Fig. 5)													
Chip select stable before $\overline{\text{WR}}$ - t_{CSW}		65		ns									
Chip select hold time for $\overline{\text{WR}}$ - t_{WCS}		20		ns									
$\overline{\text{WR}}$ pulse width - t_{WW}		400		ns									
Data set-up time for $\overline{\text{WR}}$ - t_{DW}		320		ns									
Data hold time for $\overline{\text{WR}}$ - t_{WD}		100		ns									
INTR transition time from $\overline{\text{WR}}$ - t_{WI}			700	ns									

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI and all other pins pulled up to V_{CC} with 20k resistors. See current drain equation on page 16.

Note 2: Test conditions: All inputs tied to V_{CC} ; L lines in TRI-STATE mode and tied to Ground; all outputs tied to Ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

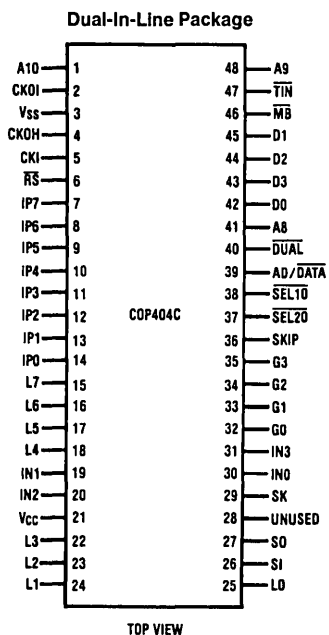
Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.1 V_{CC} in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} to prevent entering test mode.

Note 7: MB, TIN, DUAL, SELT0, SELZ0, input levels at V_{CC} or V_{SS} .

Connection Diagram



Order Number COP404CN
See NS Package Number N48A

TL/DD/5530-2

Pin Descriptions

Pin	Description
V _{CC}	Most positive voltage
V _{SS}	Ground
CKI	Clock input
\overline{RS}	Reset input
CKOI	General purpose input
L0-L7	8 TRI-STATE I/O
G0-G3	4 general purpose I/O
D1-D3	3 general purpose outputs
D0	Either general purpose output or Dual-Clock RC input
IN0-IN3	4 general purpose inputs
SO	Serial data output
SI	Serial data input
SK	Serial data clock output
IP0-IP7	I/O for ROM address and data
A8, A9, A10	3 address outputs
SKIP	Skip status output
AD/ \overline{DATA}	Clock output
\overline{MB}	MICROBUS select input
CKOH	Halt I/O pin
\overline{DUAL}	Dual-Clock select input
\overline{TIN}	Timer input select pin (should be connected to GND)
$\overline{SEL10}$	COP410C emulation select input
$\overline{SEL20}$	COP424C emulation select input
UNUSED	Ground

FIGURE 2

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program Memory consists of a 2048-byte external memory (typically PROM). Words of this memory may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by a 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

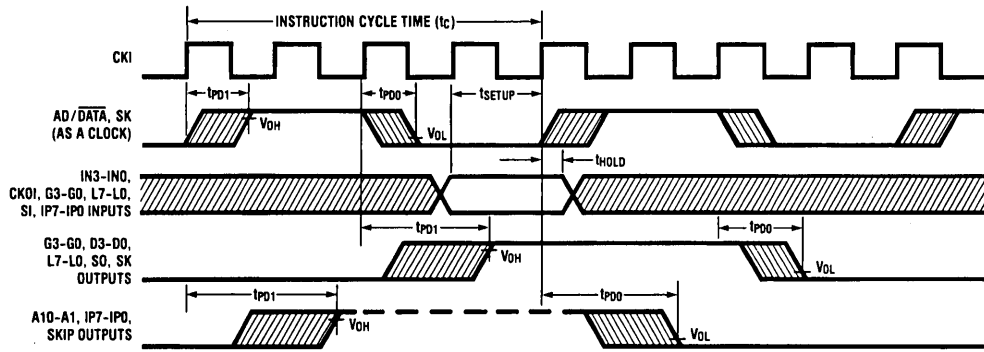
Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt

pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

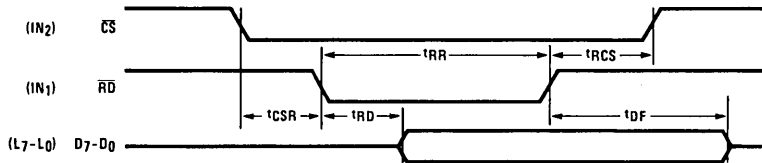
Data memory consists of a 512-bit RAM, organized as 8 data registers of 16×4 -bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (B_7) select 1 of 8 data registers and lower 4 bits (B_4) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions. The B_4 register also serves as a source register for 4-bit data sent directly to the D outputs.

Timing Diagrams



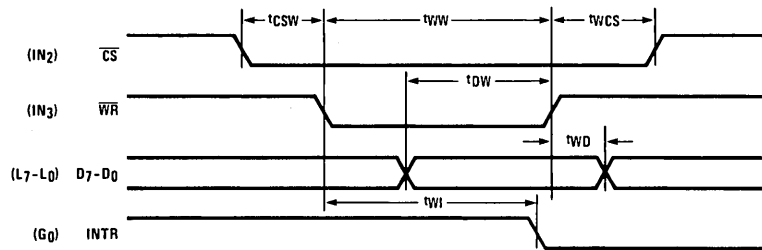
TL/DD/5530-3

FIGURE 3. Input/Output Timing



TL/DD/5530-4

FIGURE 4. MICROBUS Read Operation Timing



TL/DD/5530-5

FIGURE 5. MICROBUS Write Operation Timing

Functional Description

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the B_r and B_d portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter is operated as a time-base counter. When the T counter overflows, an overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 10a*.

Four general-purpose inputs, IN₃–IN₀, are provided. IN₁, IN₂ and IN₃ may be selected (by pulling \overline{MB} pin low) as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in MICROBUS application.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of B_d. In the dual clock mode, DO latch controls the clock selection (see dual oscillator below).

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G₀ may be selected as an output for MICROBUS applications.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register.

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE™ I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

0. The least significant bit of the enable register, EN₀, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN₃. With EN₀ reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN₁ set, interrupt is enabled. Immediately following an interrupt, EN₁ is reset to disable further interrupts.
2. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN₂ disables the L drivers, placing the L I/O port in a high-impedance input state.
3. EN₃, in conjunction with EN₀, affects the SO output. With EN₀ set (binary counter option selected) SO will output the value loaded into EN₃. With EN₀ reset (serial shift register option selected), setting EN₃ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN₃ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC + 1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN₁ is reset.
- An interrupt will be recognized only on the following conditions:
 - EN₁ has been set.
 - A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN₁ input.
 - A currently executing instruction has been completed.

TABLE I. ENABLE REGISTER MODES — BITS EN₀ AND EN₃

EN ₀	EN ₃	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).
- c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of an ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.
- d. The instruction at hex address 0FF must be a NOP.
- e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

MICROBUS INTERFACE

With \overline{MB} pin tied to Ground, the COP404C can be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μ P). IN1, IN2 and IN3 general purpose inputs become MICROBUS compatible read-strobe, chip-select, and write-strobe lines, respectively. IN1 becomes \overline{RD} — a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μ P. IN2 becomes \overline{CS} — a logic "0" on this line selects the COP404C and the μ P peripheral device by enabling the operation of the \overline{RD} and \overline{WR} lines and allows for the selection of one of several peripheral components. IN3 becomes \overline{WR} — a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP404C. G0 becomes INTR a "ready" output, reset by a write pulse from the μ P on the \overline{WR} line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP404C.

This option has been designed for compatibility with National's MICROBUS - a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS National Publication). The

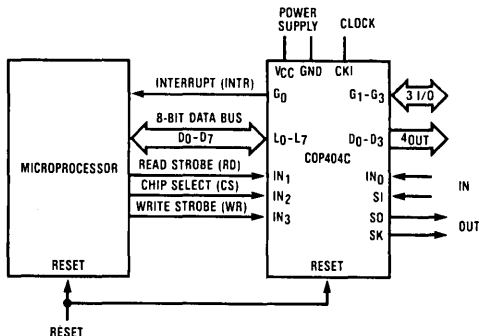


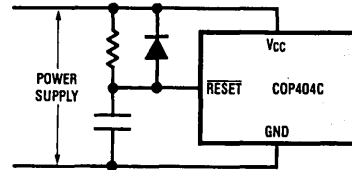
FIGURE 6. MICROBUS Option Interconnect

functioning and timing relationships between the signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams (Figures 4 and 5). Connection of the COP404C to the MICROBUS is shown in Figure 6.

INITIALIZATION

The external RC network shown in Figure 7 must be connected to the \overline{RESET} pin for the internal reset logic to initialize the device upon power-up. The \overline{RESET} pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC} . Initialization will occur whenever a logic "0" is applied to the \overline{RESET} input, providing it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



$RC \geq 5X$ POWER SUPPLY RISE TIME
AND $RC \geq 100X$ CKI PERIOD.

TL/DD/5530-8

FIGURE 7. Power-Up Circuit

TIMER

The timer is operated as a time-base counter. The instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset. For example, using a 1MHz crystal, the instruction cycle frequency of 250 kHz (divide by 4) increments the 10-bit timer every 4 μ S. By presetting the counter and detecting overflow, accurate timeouts between 16 μ S (4 counts) and 4.096 mS (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

HALT MODE

The COP404C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by two other ways (see Figure 8):

- Software HALT: by using the HALT instruction.
- Hardware HALT: by using the HALT I/O port CKOH. It is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing CKOH high the

Functional Description (Continued)

chip will stop as soon as CKI is high and CKOH output will stay high to keep the chip stopped if the external driver returns to high impedance state.

Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing.

The chip may be awakened by one of two different methods:

- Continue function: by forcing CKOH low, the system clock will be re-enabled and the circuit will continue to operate from the point where it was stopped. CKOH will stay low.
- Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization)

The HALT mode is the minimum power dissipation state.

Note: if the user has selected dual-clock (DUAL pin tied to Ground) AND is forcing an external clock on D0 pin AND the COP404C is running from the D0 clock, the HALT mode - either hardware or software - will NOT be entered. Thus, the user should switch to the CKI clock to HALT. Alternatively, the user may stop the D0 clock to minimize power.

Oscillator Options

There are two basic clock oscillator configurations available as shown by Figure 9.

- CKI oscillator: CKI is configured as a LSTTL compatible input external clock signal. The external frequency is divided by 4 to give the instruction cycle time.
- Dual oscillator. By tying DUAL pin to Ground, pin D0 is now a single pin RC controlled Schmitt trigger oscillator input. The user may software select between the

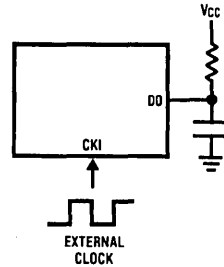
D0 oscillator (the instruction cycle time equals the D0 oscillation frequency divided by 4) by setting the D0 latch high or the CKI oscillator by resetting D0 latch low.

Note that even in dual clock mode, the counter, if used as a time-base counter, is always connected to the CKI oscillator.

For example, the user may connect up to a 1 MHz RC circuit to D0 for faster processing and a 32 kHz external clock to CKI for minimum current drain and time keeping.

Note: CTMA instruction is not allowed when the chip is running from D0 clock.

Figures 10a and 10b show the timer and clock diagrams with and without Dual-Clock.



TL/DD/5530-9

R	C	Cycle Time	Vcc
15k	82 pF	4-9 μs	≥ 4.5V
30k	82 pF	8-16 μs	≥ 4.5V
60k	100 pF	16-32 μs	2.4-4.5V

Note: $15k \leq R \leq 150k$

$50 \text{ pF} \leq C \leq 150 \text{ pF}$

FIGURE 9. Dual-Oscillator Component Values

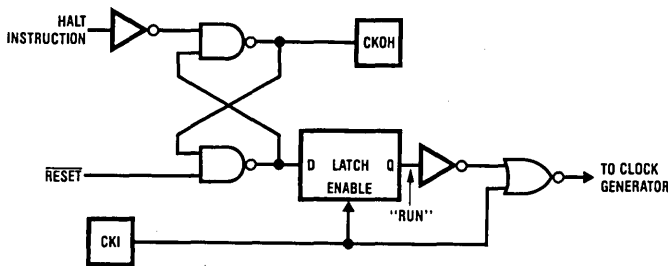


FIGURE 8. HALT Mode

TL/DD/5530-10

Functional Description (Continued)

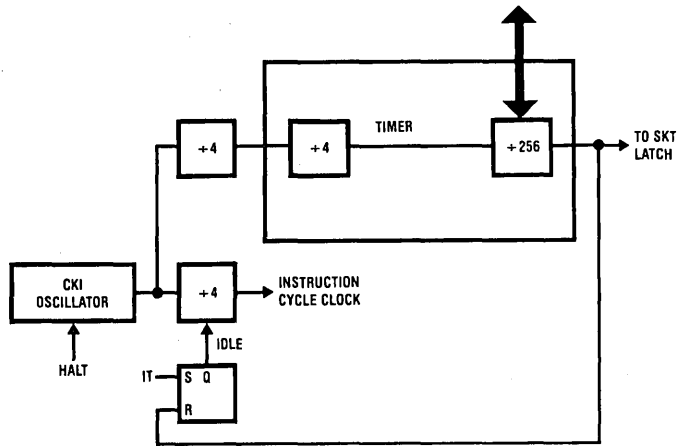


FIGURE 10a. Clock and Timer Block Diagram without Dual-Clock

TL/DD/5530-11

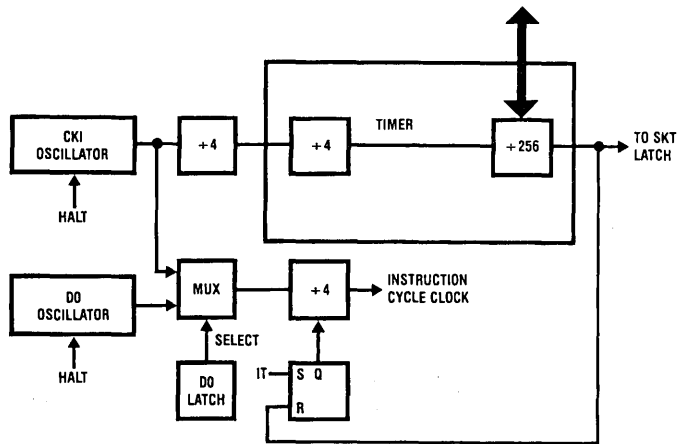


Figure 10b. Clock and Timer Block Diagram with Dual-Clock

TL/DD/5530-12

External Memory Interface

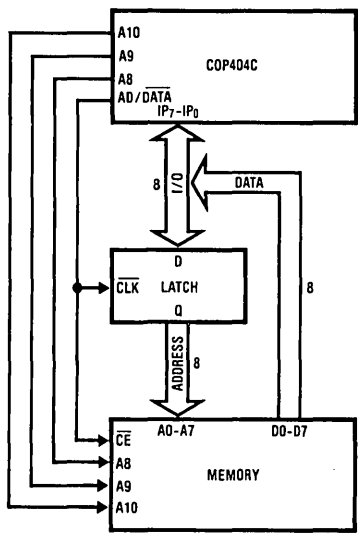
The COP404C is designed for use with an external Program Memory.

This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. LSTTL or CMOS-compatible TRI-STATE outputs
3. LSTTL or CMOS-compatible inputs
4. access time = 1.0 μ s max.

Typically, these requirements are met using bipolar PROMs or MOS/CMOS PROMs, EPROMs or E²PROMs.

During operation, the address of the next instruction is sent out on A10, A9, A8 and IP7 through IP0 during the time that AD/DATA is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/DATA line; A10, A9 and A8 are dedicated address outputs, and do not need to be latched. When AD/DATA is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that AD/DATA output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or data input. A simplified block diagram of the external memory interface is shown in Figure 11.



TL/DD/5530-13

FIGURE 11. External Memory Interface to COP404C

COP404C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

Table II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	7-bit RAM address register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry register
D	4-bit Data output port
EN	4-bit Enable register
G	4-bit General purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Instruction operand symbols

- d 4-bit operand field, 0-15 binary (RAM digit select)
- r 3-bit operand field, 0-7 binary (RAM register select)
- a 11-bit operand field, 0-2047
- y 4-bit operand field, 0-15 (immediate data)

- RAM(x) RAM addressed by variable x
- ROM(x) ROM addressed by variable x

Operational Symbols

- + Plus
- Minus
- > Replaces
- <-> is exchanged with
- = Is equal to
- one's complement of A
- ⊕ exclusive-or
- :

Instruction Set (Continued)

TABLE III. COP404C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	$A+C+RAM(B) \rightarrow A$ Carry $\rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	$A+RAM(B) \rightarrow A$	None	Add RAM to A
ADT		4A	0011 0001	$A+10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5-	0101 y	$A+y \rightarrow A$	Carry	Add Immediate. Skip on Carry ($y \neq 0$)
CASC		10	0001 0000	$\bar{A}+RAM(B)+C \rightarrow A$ Carry $\rightarrow C$	Carry	Compliment and Add with Carry, Skip on Carry
CLRA		00	0000 0000	$0 \rightarrow A$	None	Clear A
COMP		40	0100 0000	$\bar{A} \rightarrow A$	None	Ones complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" $\rightarrow C$	None	Reset C
SC		22	0010 0010	"1" $\rightarrow C$	None	Set C
XOR		02	0000 0010	$A \oplus RAM(B) \rightarrow A$	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{10:8}, A, M$) $\rightarrow PC_{7:0}$	None	Jump Indirect (note 2)
JMP	a	6-	0110 0 a _{10:8}	$a \rightarrow PC$	None	Jump
JP	a	-	1 a _{7:0}	$a \rightarrow PC_{6:0}$	None	Jump within Page (Note 3)
			(pages 2,3 only)			
			or			
			11 a _{5:0}	$a \rightarrow PC_{5:0}$		
			(all other pages)			
JSRP	a	-	10 a _{5:0}	$PC+1 \rightarrow SA \rightarrow SB \rightarrow SC$ $00010 \rightarrow PC_{10:6}$ $a \rightarrow PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6-	0110 1 a _{10:8}	$PC+1 \rightarrow SA \rightarrow SB \rightarrow SC$ $a \rightarrow PC$	None	Jump to Subroutine
RET		48	0100 1000	$SC \rightarrow SB \rightarrow SA \rightarrow PC$	None	Return from Subroutine
RETSK		49	0100 1001	$SC \rightarrow SB \rightarrow SA \rightarrow PC$	Always Skip on Return	Return from Subroutine then Skip
HALT		33	0011 0011		None	HALT processor
		38	0011 1000			
IT		33	0011 0011			IDLE till timer overflows then continues
		39	0011 1001		None	
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33	0011 0011	$A \rightarrow T_{7:4}$ $RAM(B) \rightarrow T_{3:0}$	None	Copy A, RAM to T
CTMA		33	0011 0011	$T_{7:4} \rightarrow RAM(B)$		
		2F	0010 1111	$T_{3:0} \rightarrow A$	None	Copy T to RAM, A
CAMQ		33	0011 0011	$A \rightarrow Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 1100	$RAM(B) \rightarrow Q_{3:0}$		
CQMA		33	0011 0011	$Q_{7:4} \rightarrow RAM(B)$	None	Copy Q to RAM, A
		2C	0010 1100	$Q_{3:0} \rightarrow A$		
LD	r	-5	00 r 0101	$RAM(B) \rightarrow A$ $Br \oplus r \rightarrow Br$	None	Load RAM into A, Exclusive-OR Br with r
		(r=0:3)				
LDD	r,d	23	0010 0011	$RAM(r,d) \rightarrow A$	None	Load A with RAM pointed to direct by r,d
		-	0 r d			
LQID		BF	1011 1111	ROM($PC_{10:8}, A, M$) $\rightarrow Q$ $SB \rightarrow SC$	None	Load Q Indirect (Note 2)
RMB	0	4C	0100 1100	$0 \rightarrow RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	$0 \rightarrow RAM(B)_1$		
	2	42	0100 0010	$0 \rightarrow RAM(B)_2$		
	3	43	0100 0011	$0 \rightarrow RAM(B)_3$		

Instruction Set (Continued)

TABLE III. COP404C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → RAM(B) ₀ 1 → RAM(B) ₁ 1 → RAM(B) ₂ 1 → RAM(B) ₃	None	Set RAM Bit
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110 (r=0:3)	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	0010 0011 1 r d	RAM(r,d) ↔ A	None	Exchange A with RAM pointed to directly by r,d
XDS	r	-7	00 r 0111 (r=0:3)	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	-4	00 r 0100 (r=0:3)	RAM(B) ↔ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	—	00 r (d-1) (r=0:3; d=0,9:15) or 33 — 1 r d (any r, any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 6)
XABR		12	0001 0010	A ↔ Br	None	Exchange A with Br (Note 7)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		21	0010 0001	1st byte		Skip if G Bit is Zero
	0	01	0000 0001		G ₀ = 0	
	1	11	0001 0001		G ₁ = 0	
	2	03	0000 0011	2nd byte	G ₂ = 0	
	3	13	0001 0011		G ₃ = 0	
SKMBZ	0	01	0000 0001		RAM(B) ₀ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001		RAM(B) ₁ = 0	
	2	03	0000 0011		RAM(B) ₂ = 0	
	3	13	0001 0011		RAM(B) ₃ = 0	
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 2)

Instruction Set (Continued)

TABLE III. COP404C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
ININ		33	0011 0011	IN → A	None	Input IN Inputs to A
		28	0010 1000			
INIL		33	0011 0011	IL ₃ , CKO, "0", IL ₀ → A	None	Input IL Latches to A (Note 2)
		29	0010 1001			
INL		33	0011 0011	L _{7:4} → RAM(B) L _{3:0} → A	None	Input L Ports to RAM,A
		2E	0010 1110			
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OGI	y	33	0011 0011	y → G	None	Output to G Ports Immediate
		5-	0101 y			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data minus 1, e.g., to load the lower four bits of B(Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 7: If $\overline{\text{SELZ0}} = 1$, A ↔ Br (0 → A3)

If $\overline{\text{SELZ0}} = 0$, A ↔ Br (0,0 → A3, A2).

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10: PC8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC (7:4), RAM(B) → PC(3:0), leaving PC(10), PC(9) and PC(8) unchanged. The ROM data pointed to by the

new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10: B, A, M. PC10, PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

Description of Selected Instructions (Continued)

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT      ; load T counter
SKT       ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKOI and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. The state of CKOI is input into A2. A 0 is input into A1. IL latches are cleared on reset.

Instruction Set Notes

- The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.
- The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, for minimum power dissipation, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. For example, an RC oscillator on D0 will draw more current than a square wave clock input since it is a slow rising signal.

If using an external square wave oscillator, the following equation can be used to calculate the COP404C operating current drain:

$$I_{co} = I_q + V \times 40 \times F_i + V \times 1400 \times F_i / 4$$

where:

I_{co} = chip operating current drain in microamps

I_q = quiescent leakage current (from curve)

F_i = CKI frequency in MegaHertz

V = chip V_{CC} in volts

For example at 5 volts V_{CC} and 400 kHz:

$$I_{co} = 20 + 5 \times 40 \times .4 + 5 \times 1400 \times .4 / 4$$

$$I_{co} = 20 + 80 + 700 = 800 \mu A$$

at 2.4 volts V_{CC} and 30 kHz:

$$I_{co} = 6 + 2.4 \times 40 \times .03 + 2.4 \times 1400 \times .03 / 4$$

$$I_{co} = 6 + 2.88 + 25.2 = 34.08 \mu A$$

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$I_{ci} = I_q + V \times 40 \times F_i$$

For example, at 5 volts V_{CC} and 400 kHz

$$I_{ci} = 20 + 5 \times 40 \times .4 = 100 \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$I_{ta} = I_{co} \times \frac{T_o}{T_o + T_i} + I_{ci} \times \frac{T_i}{T_o + T_i}$$

where:

I_{ta} = total average current

I_{co} = operating current

I_{ci} = idle current

T_o = operating time

T_i = idle time

I/O OPTIONS

COP404C outputs have the following configurations, illustrated in *Figure 12*.

- Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC} , compatible with CMOS and LSTTL. (Used on SO, SK, AD/DATA, SKIP, A10:8 and D outputs.)
- Low Current — This is the same configuration as a. above except that the sourcing current is much less. (Used on G outputs.)
- Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control. (Used on L outputs.)

All inputs have the following configuration:

- Input with on chip load device to V_{CC} . (Used on CKOI.)
- Hi-Z input which must be driven by the users logic. (Used on CKI, RESET, IN, SI, DUAL, MB, SELT0 and SEL20 inputs.)

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 13* for each of these devices to allow the designer to effectively use these I/O configurations.

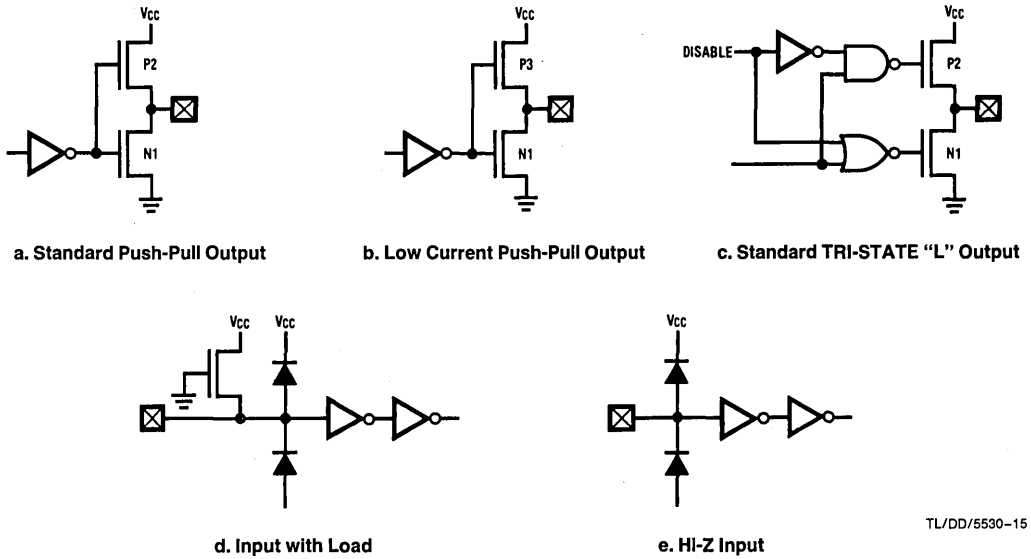


FIGURE 12. Input/Output Configurations

TL/DD/5530-15

Typical Performance Characteristics

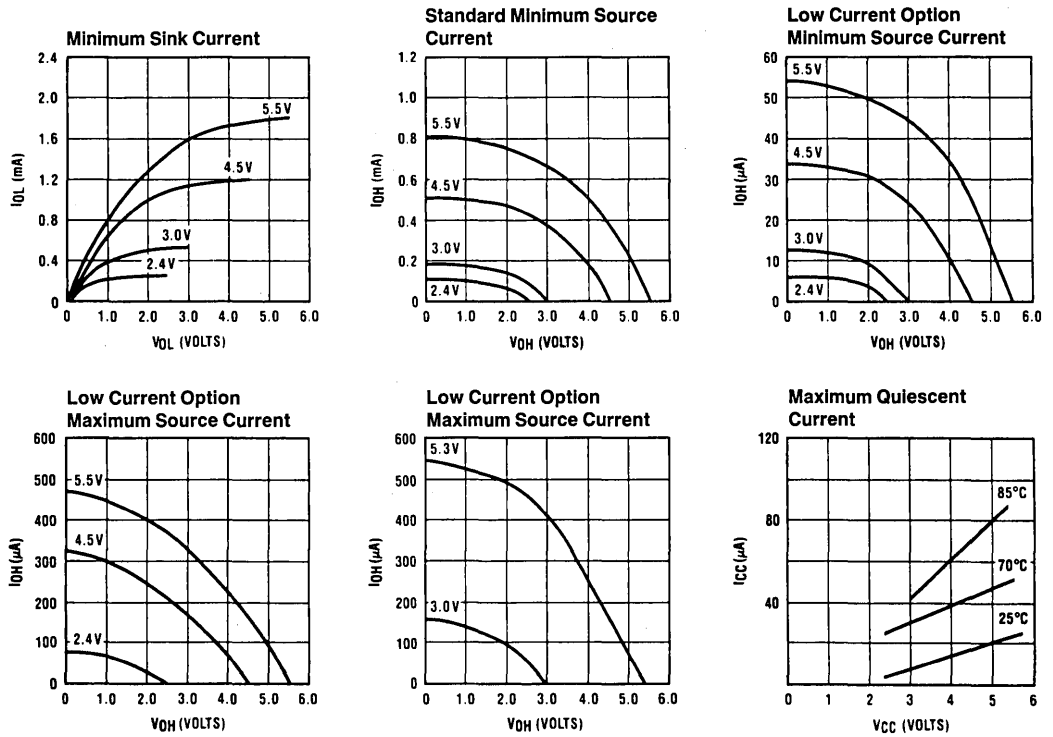


FIGURE 13. Input/Output Characteristics

TL/DD/5530-16

Emulation

The COP404C may be used to exactly emulate the COP444C/445C, COP424C/425C, and COP410C/411C. However, the Program Counter always addresses 2k of external ROM whatever chip is being emulated. *Figure 14* shows the interconnect to implement a hardware emulation. This connection uses a NMC27C16 EPROM as external

memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7-IP0 are bidirectional inputs and outputs. When the AD/DAT \bar{A} clocking output turns on, the EPROM drivers are disabled and IP7-IP0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

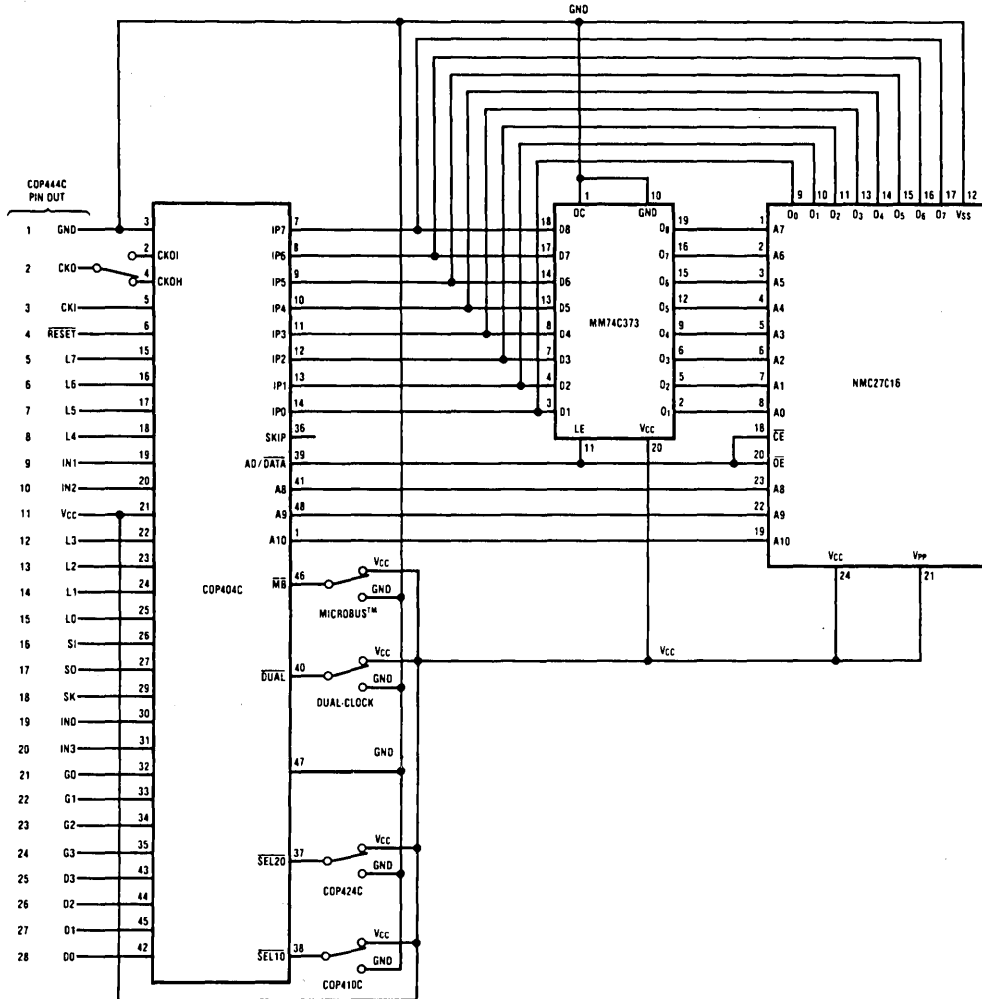


FIGURE 14. COP404C Used To Emulate A COP444C

TL/DD/5530-14

Emulation (Continued)

When AD/DATA turns off, the EPROM is enabled and the IP7-IP0 pins will input the memory data. A10, A9 and A8 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

- CKI is divided by 4. Other divide-by are emulated by external divider.
- CKO can be emulated as a general purpose input by using CKOI or as a Halt I/O port by using CKOH.
- MB pin can be pulled low if the MICROBUS feature of the COP444C and COP424C is needed. Otherwise it should be high.
- DUAL pin can be pulled low if the Dual-Clock feature of the COP444C and COP424C is needed. Otherwise it should be high.
- The SEL10 and SEL20 inputs are used to emulate the COP444C/445C, COP424C/425C, or COP410C/411C.
 - When emulating the COP444C/445C, the user must configure SEL20=1 and SEL10=1.
 - When emulating the COP424C/425C, the user must configure SEL20=0 and SEL10=1. In this mode, the user RAM is physically halved. As in the COP424C/425C, the user has 64 digits (256 bits) of RAM available. Pin A10 should not be connected to the program memory (most significant address bit of the program memory should be grounded if using a 2k×8 memory).
 - When emulating the COP410C/411C, the user must configure SEL20=0 and SEL10=0. In this mode, the user has 32 digits (128 bits) of RAM available organized in the same way as the COP410C/411C - 4 registers of 8 digits each. Pins A10 and A9 should not be connected to the program memory (the 2 most signifi-

cant address bits of the program memory should be grounded).

Furthermore, the subroutine stack is decreased from 3 levels to 2 levels.

The pins SEL10 and SEL20 change the internal logic of the device to accurately emulate the devices as indicated above. However, the user must remember that the COP424C/425C is a subset of the COP444C/COP445C with respect to memory size. The COP410C/411C is a subset both in memory size and in function. The user must take care not to use features and instructions which are not available on the COP410C/411C (see table IV. below) when using the COP404C to emulate the COP410C/411C.

TABLE IV. FEATURES AND INSTRUCTIONS NOT AVAILABLE ON COP410C/411C.

Timer	ADT		
Dual-clock	CASC		
Interrupt	CAMT		
Microbus	CTMA		
	IT		
	LDD	r, d	
	XAD	r, d	(except 3, 15)
	XABR		
	SKT		
	ININ		
	INIL		
	OGI	y	

Option Table

COP404C MASK OPTIONS

The following COP444C options have been implemented in the COP404C:

Option value	Comment
Option 1 = 0	Ground Pin — no option available
Option 2 = 1, 2	CKO is replaced by CKOI and CKOH
Option 3 = 5	CKI is external clock input divided by 4
Option 4 = 1	RESET is Hi-Z input
Option 5-8 = 0	L outputs are standard TRI-STATE
Option 9 = 1	IN1 is a Hi-Z input
Option 10 = 1	IN2 is a Hi-Z input
Option 11 = 0	VCC pin — no option available
Option 12-15 = 0	L outputs are standard TRI-STATE
Option 16 = 1	SI is a Hi-Z input
Option 17 = 0	SO is a standard output
Option 18 = 0	SK is a standard output
Option 19 = 1	IN0 is a Hi-Z input
Option 20 = 1	IN3 is a Hi-Z input
Option 21-24 = 1	G outputs are low-current
Option 25-28 = 0	D outputs are standard
Option 29 = 1	No internal initialization logic
Option 30 = 0, 1	DUAL-CLOCK is pin selectable
Option 31 = 0	TIMER time-base counter
Option 32 = 0, 1	MICROBUS is pin selectable
Option 33 = N/A	48-pin package

COP404LSN-5 ROMless N-Channel Microcontrollers

General Description

The COP404LSN-5 ROMless Microcontroller is a member of the COPSM family, fabricated using N-channel, silicon gate MOS technology. The COP404LSN-5 contains CPU, RAM, I/O and is identical to a COP444L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP404LSN-5 will perform exactly as the COP444L. This important benefit facilitates development and debug of a COP program prior to masking the final part. The COP404LSN-5 is also appropriate in low volume applications, or when the program might be changing. The COP404LSN-5 may be used to emulate the COP444L, COP445L, COP420L, and the COP421L.

Use COP404LSN-5 in volume applications. For extended temperature range (-40°C to +85°C), COP304L is available on a special order basis.

Features

- Exact circuit equivalent of COP444L
- Low cost
- Powerful instruction set
- 128 x 4 RAM, addresses 2048 x 8 ROM
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 16 μ s instruction time
- Single supply operation (4.5V-5.5V)
- Low current drain (16 mA max)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRETM compatible serial I/O
- General purpose outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family

Block Diagram

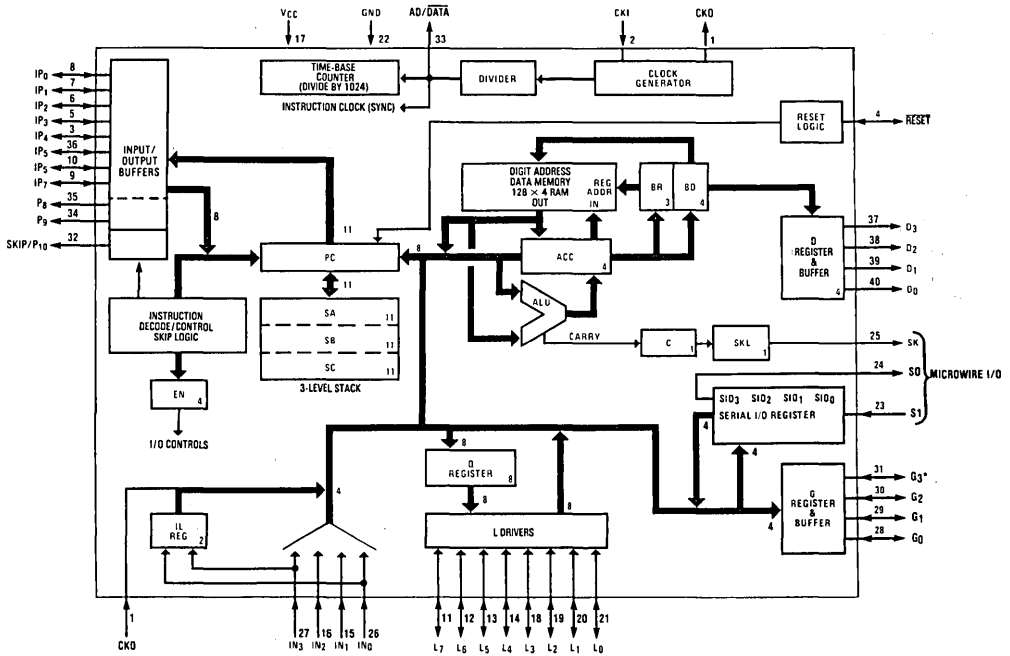


FIGURE 1

TL/DD/8817-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C

Total Source Current 120 mA

Total Sink Current 140 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

4.5V ≤ V_{CC} ≤ 5.5V; 0°C ≤ T_A ≤ 70°C

Parameter	Conditions	Min	Max	Units
Operating Voltage (V _{CC})		4.5	5.5	V
Power Supply Ripple (Notes 2, 3)	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		16	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input	V _{CC} = Max			
Logic High (V _{IH})		2.0		V
Logic Low (V _{IL})		-0.3	0.4	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
IP0-IP7, SI Input Levels				
Logic High	V _{CC} = 5.5V	2.4		V
Logic High	V _{CC} = 5V ± 5%	2.0		V
Logic Low		-0.3	0.8	V
All Other Inputs				
Logic High	High Trip Level Options	3.6		V
Logic Low	Selected	-0.3	1.2	V
Input Capacitance (Note 3)			7	pF
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ± 10%			
Logic High (V _{OH})	I _{OH} = -25 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 mA		0.4	V
IP0-IP7, P8, P9, SKIP/P10	(Note 1)			
Logic High	I _{OH} = -80 μA	2.4		V
Logic Low	I _{OL} = 720 μA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	V _{CC} = 4.5V, V _{OL} = 0.4V	0.9		mA
L0-L7 Outputs	V _{CC} = 4.5V, V _{OL} = 0.4V	0.4		mA
G0-G3 and D0-D3 Outputs	V _{CC} = 4.5V, V _{OL} = 1.0V	7.5		mA
CKO	V _{CC} = 4.5V, V _{OL} = 0.4V	0.2		mA
Output Source Current				
D0-D3, G0-G3 Outputs (I _{OH})	V _{CC} = 4.5V, V _{OH} = 2.0V	-30	-250	μA
SO and SK Outputs (I _{OH})	V _{CC} = 4.5V, V _{OH} = 1.0V	-1.2		mA
L0-L7 Outputs	V _{CC} = 5.5V, V _{OH} = 2.0V	-1.4	-25	mA

DC Electrical Characteristics (Continued)0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

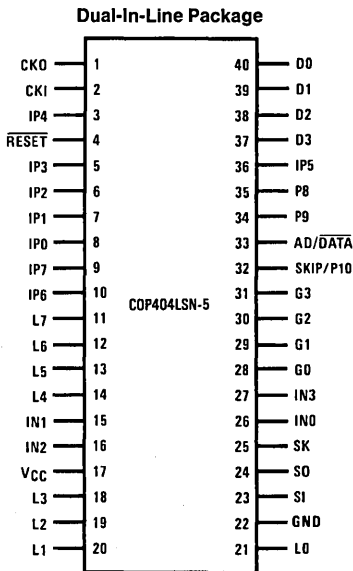
Parameter	Conditions	Min	Max	Units
Input Load Source Current (I _{IL})	V _{CC} = 5.0V, V _{IL} = 0V	-10	-140	μA
Total Sink Current Allowed				
All Outputs Combined			140	mA
D, G Ports			120	mA
L ₇ -L ₄			4	mA
L ₃ -L ₀			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L ₇ -L ₄			60	mA
L ₃ -L ₀			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		16	40	μs
CKI				
Input Frequency, f	(÷32 Mode)	0.8	2	MHz
Duty Cycle		30	60	%
Rise Time (Note 3)	f _I = 2.0 MHz		120	ns
Fall Time (Note 3)			80	ns
INPUTS:				
SI, IP7-IP0				
t _{SETUP}		2.0		μs
t _{HOLD}		1.0		μs
IN3-IN0, G3-G0, L7-L0				
t _{SETUP}		8.0		μs
t _{HOLD}		1.3		μs
OUTPUT PROPAGATION DELAY	Test Condition: C _L = 50 pF, V _{OUT} = 1.5V			
SO, SK Outputs	R _L = 20 kΩ		4.0	μs
t _{pd1} , t _{pd0}				
D3-D0, G3-G0, L7-L0	R _L = 20 kΩ		5.6	μs
t _{pd1} , t _{pd0}				
IP7-IP0, P8, P9, SKIP	R _L = 5 kΩ		7.2	μs
t _{pd1} , t _{pd0}				
P10	R _L = 5 kΩ		6.0	μs
t _{pd1} , t _{pd0}				

Note 1: COP404LSN-5 has Push-Pull drivers on these outputs.**Note 2:** V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.**Note 3:** This parameter is only sampled and not 100% tested. Variation due to the device included.

Connection Diagram



Top View

FIGURE 2

Order Number COP404LSN-5
See NS Package Number N40A

TL/DD/8817-2

Pin Descriptions

Pin	Description
L7-L0	8 bidirectional I/O ports with TRI-STATE®
G3-G0	4 bidirectional I/O ports
D3-D0	4 general purpose outputs
IN3-IN0	4 general purpose outputs
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/DATA	Address out/data in flag
CKI	System oscillator input
CKO	System oscillator output (COP404LSN-5)
RESET	System reset input
VCC	Power supply
GND	Ground
IP7-IP0	8 bidirectional ROM address and data ports
P8, P9	2 ROM address outputs
SKIP/P10	Instruction skip output and most significant ROM address bit output

Timing Diagram

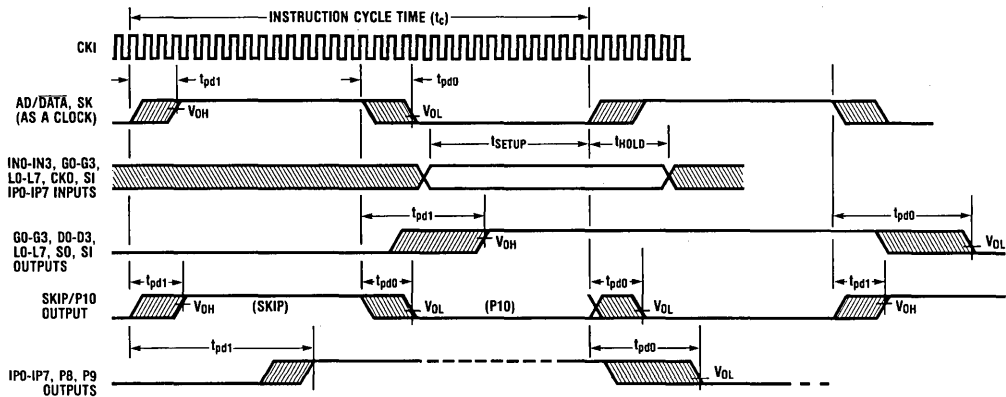


FIGURE 3. Input/Output

TL/DD/8817-3

Functional Description

A block diagram of the COP404LSN-5 is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 2048 byte external memory. As can be seen by an examination of the COP404LSN-5 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by an 11-bit PC register. Its binary value selects one of the 2048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value. Three levels of subroutine nesting are implemented by the 11-bit subroutine saves registers, SA, SB, and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

Four general-purpose inputs, IN₃–IN₀, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN₃–EN₀).

1. The least significant bit of the enable register, EN₀, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN₀ set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN₃. With EN₀ reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN₁ set the IN₁ input is enabled as an interrupt input. Immediately following an interrupt, EN₁ is reset to disable further interrupts.
3. With EN₂ set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN₂ disables the L drivers, placing the L I/O ports in a high-impedance input state.

Functional Description (Continued)

4. EN_3 , in conjunction with EN_0 , affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3 . With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0 .

INTERRUPT

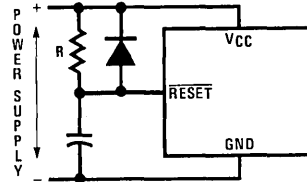
The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

- The interrupt, once acknowledged as explained below, pushes the next sequential program counter address ($PC+1$) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level ($PC+1 \rightarrow SA \rightarrow SB \rightarrow SC$). Any previous contents of SC are lost. The program counter is set to hex address OFF (the last word of page 3) and EN_1 is reset.
- An interrupt will be acknowledged only after the following conditions are met:
 - EN_1 has been set.
 - A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
 - A currently executing instruction has been completed.
 - All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).
- Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address OFF. At the end of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

- The first instruction of the interrupt routine at hex address OFF must be a NOP.
- A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the RESET pin as shown below. The RESET pin is configured as a Schmitt trigger input. If the RC network is not used, the RESET pin should be left open. Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.



TL/DD/8817-4

$$RC \geq 5 \times \text{Power Supply Rise Time (R} > 40k)$$

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP404LSN-5 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

- random addressing
- TTL-compatible TRI-STATE outputs
- TTL-compatible inputs
- access time = 5 μ s max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P10, P9, P8, and IP7 through IP0 during the time that $\overline{AD/DATA}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the $\overline{AD/DATA}$ line; P9 and P8 are

Enable Register Modes — Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

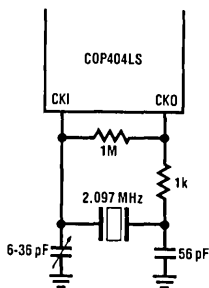
Functional Description (Continued)

dedicated address outputs, and do not need to be latched. SKIP/P10 outputs address data when AD/DATA is low. When AD/DATA is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/DATA output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

The basic clock oscillator configurations is shown in *Figure 4*.

Crystal Controlled Oscillator—CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32.



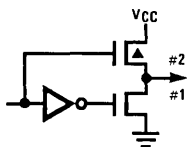
TL/DD/8817-5

FIGURE 4. Oscillator

INPUT/OUTPUT CONFIGURATIONS

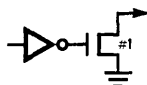
COP404LSN-5 outputs have the following configurations, illustrated in *Figure 5*:

a. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to VCC, compatible with LSTTL and CMOS input requirements. (Used on D and G outputs.)



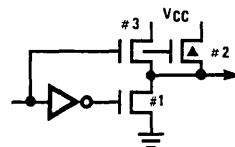
TL/DD/8817-6

a. Standard Output



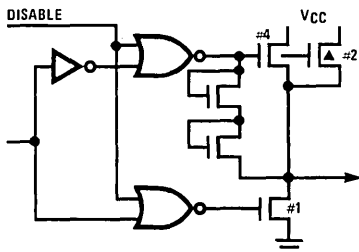
TL/DD/8817-7

b. Open-Drain Output



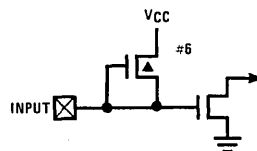
TL/DD/8817-8

c. Push-Pull Output



TL/DD/8817-9

d. L Output (LED)



TL/DD/8817-10

e. Input with Load

(Δ is Depletion Device)

FIGURE 5. Output Configurations

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application.

c. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to VCC. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

d. LED Direct Drive—an enhancement-mode device to ground and to VCC, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. (Used on L outputs.)

COP404LSN-5 inputs have an on-chip depletion load device to VCC.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to a logic "1".

Typical Performance Characteristics

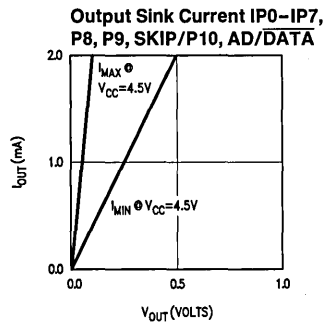
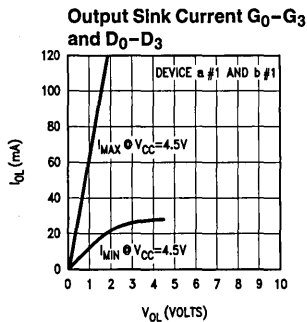
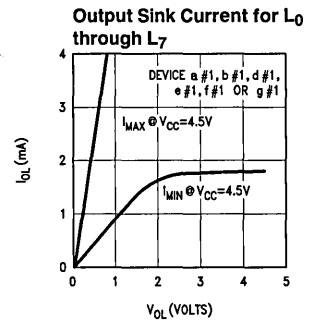
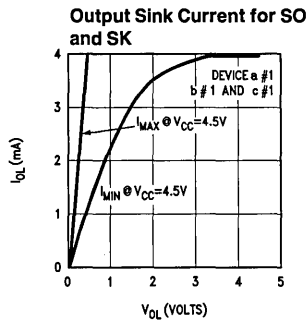
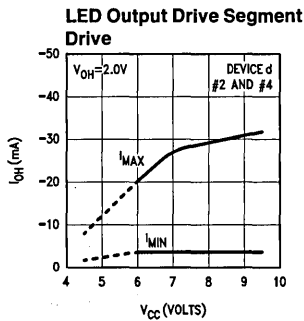
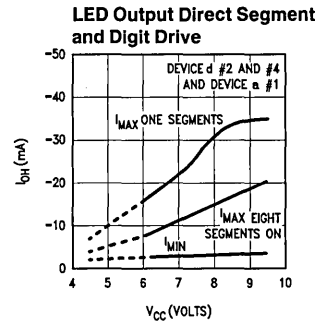
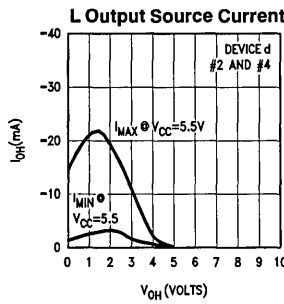
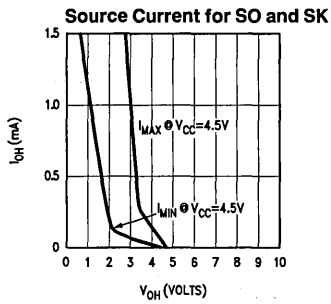
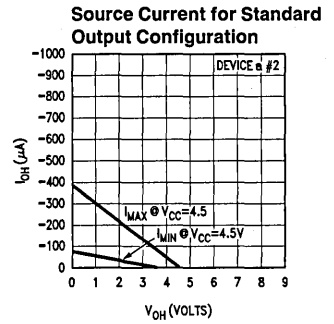
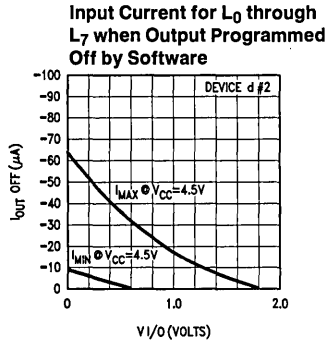
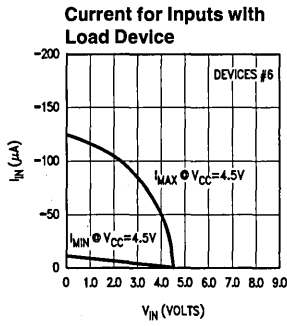


FIGURE 6. COP404LSN-5 I/O Characteristics

COP404LSN-5 Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP404LSN-5 instruction set.

TABLE I. COP404LSN-5 Instruction Set Table Symbols

Symbol	Definition	Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS		INSTRUCTION OPERAND SYMBOLS	
A	4-bit Accumulator	d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
B	10-bit RAM Address Register	r	3-bit Operand Field, 0–7 binary (RAM Register Select)
Br	Upper 3 bits of B (register address)	a	11-bit Operand Field, 0–2047 binary (ROM Address)
Bd	Lower 4 bits of B (digit address)	y	4-bit Operand Field, 0–15 binary (Immediate Data)
C	1-bit Carry Register	RAM(s)	Contents of RAM location addressed by s
D	4-bit Data Output Port	ROM(t)	Contents of ROM location addressed by t
EN	4-bit Enable Register		
G	4-bit Register to latch data for G I/O Port	OPERATIONAL SYMBOLS	
IL	Two 1-bit latches associated with the IN ₃ or IN ₀ inputs	+	Plus
IN	4-bit Input Port	–	Minus
IP	8-bit bidirectional ROM address and Data Port	→	Replaces
L	8-bit TRI-STATE I/O Port	↔	Is exchanged with
M	4-bit contents of RAM Memory pointed to by B Register	=	Is equal to
P	3-bit ROM Address Register Port	\bar{A}	The one's complement of A
PC	11-bit ROM Address Register (program counter)	⊕	Exclusive-OR
Q	8-bit Register to latch data for L I/O Port	:	Range of values
SA	11-bit Subroutine Save Register A		
SB	11-bit Subroutine Save Register B		
SC	11-bit Subroutine Save Register C		
SIO	4-bit Shift Register and Counter		
SK	Logic-Controlled Clock Output		

TABLE II. COP404LSN-5 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	<u>0011</u> <u>0000</u>	$A + C + \text{RAM}(B) \rightarrow A$ $\text{Carry} \rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	<u>0011</u> <u>0001</u>	$A + \text{RAM}(B) \rightarrow A$	None	Add RAM to A
ADT		4A	<u>0100</u> <u>1010</u>	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5–	<u>0101</u> y	$A + y \rightarrow A$	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	<u>0001</u> <u>0000</u>	$\bar{A} + \text{RAM}(B) + C \rightarrow A$ $\text{Carry} \rightarrow C$	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	<u>0000</u> <u>0000</u>	$0 \rightarrow A$	None	Clear A
COMP		40	<u>0100</u> <u>0000</u>	$\bar{A} \rightarrow A$	None	One's complement of A to A
NOP		44	<u>0100</u> <u>0100</u>	None	None	No Operation
RC		32	<u>0011</u> <u>0010</u>	"0" → C	None	Reset C
SC		22	<u>0010</u> <u>0010</u>	"1" → C	None	Set C
XOR		02	<u>0000</u> <u>0010</u>	$A \oplus \text{RAM}(B) \rightarrow A$	None	Exclusive-OR RAM with A

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description	
TRANSFER OF CONTROL INSTRUCTIONS							
JID		FF	1111 1111	ROM (PC _{10:8} , A, M) → PC _{7:0}	None	Jump Indirect (Note 2)	
JMP	a	6--	0110 0 a _{10:8} a _{7:0}	a → PC	None	Jump	
JP	a	--	1 a _{6:0} (pages 2,3 only)	a → PC _{6:0}	None	Jump within Page (Note 4)	
			11 a _{5:0} (all other pages)	a → PC _{5:0}			
JSRP	a	--	10 a _{5:0}	PC + 1 → SA → SB → SC 00010 → PC _{10:6} a → PC _{5:0}	None	Jump to Subroutine Page (Note 5)	
JSR	a	6--	0110 1 a _{10:8} a _{7:0}	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine	
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine	
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip	
MEMORY REFERENCE INSTRUCTIONS							
CAMQ		33	0011 0011	A → Q _{7:4}	None	Copy A, RAM to Q	
		3C	0011 1100	RAM(B) → Q _{3:0}			
CQMA		33	0011 0011	Q _{7:4} → RAM(B)	None	Copy Q to RAM, A	
		2C	0010 1100	Q _{3:0} → A			
LD	r	-5	00 r 0101 (r = 0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r	
LDD	r,d	23	0010 0011 0 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d	
LQID		BF	1011 1111	ROM(PC _{10:8} , A, M) → Q SB → SC	None	Load Q Indirect (Note 3)	
RMB	0	4C	0100 1100	0 → RAM(B) ₀	None	Reset RAM Bit	
		1	45	0100 0101			0 → RAM(B) ₁
		2	42	0100 0010			0 → RAM(B) ₂
		3	43	0100 0011			0 → RAM(B) ₃
SMB	0	4D	0100 1101	1 → RAM(B) ₀	None	Set RAM Bit	
		1	47	0100 0111			1 → RAM(B) ₁
		2	46	0100 0110			1 → RAM(B) ₂
		3	4B	0100 1011			1 → RAM(B) ₃
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd	
X	r	-6	00 r 0110 (r = 0:3)	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r	
XAD	r,d	23	0010 0011 1 r d	RAM(r,d) ↔ A	None	Exchange A with RAM pointed to directly by (r,d)	
XDS	r	-7	00 r 0111 (r = 0:3)	RAM(B) ↔ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r	

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XIS	r	-4	$\begin{array}{ c c } \hline 00 & r 0100 \\ \hline \end{array}$ (r = 0:3)	RAM(B) \leftrightarrow A Bd + 1 \rightarrow Bd Br \oplus r \rightarrow Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	$\begin{array}{ c c } \hline 0101 & 0000 \\ \hline \end{array}$	A \rightarrow Bd	None	Copy A to Bd
CBA		4E	$\begin{array}{ c c } \hline 0100 & 1110 \\ \hline \end{array}$	Bd \rightarrow A	None	Copy Bd to A
LBI	r,d	--	$\begin{array}{ c c } \hline 00 & r (d-1) \\ \hline \end{array}$ (r = 0:3; d = 0, 9:15) or 33 -- $\begin{array}{ c c c } \hline 1 & r & d \\ \hline \end{array}$ (any r, any d)	r,d \rightarrow B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6-	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0110 & y \\ \hline \end{array}$	y \rightarrow EN	None	Load EN Immediate (Note 7)
XABR		12	$\begin{array}{ c c } \hline 0001 & 0010 \\ \hline \end{array}$	A \leftrightarrow Br (0 \rightarrow A ₃)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	$\begin{array}{ c c } \hline 0010 & 0000 \\ \hline \end{array}$		C = "1"	Skip if C is True
SKE		21	$\begin{array}{ c c } \hline 0010 & 0001 \\ \hline \end{array}$		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0010 & 0001 \\ \hline \end{array}$		G _{3:0} = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$	1st byte	G ₀ = 0	Skip if G Bit is Zero
	0	01	$\begin{array}{ c c } \hline 0000 & 0001 \\ \hline \end{array}$	} 2nd byte	G ₁ = 0	
	1	11	$\begin{array}{ c c } \hline 0001 & 0001 \\ \hline \end{array}$		G ₂ = 0	
	2	03	$\begin{array}{ c c } \hline 0000 & 0011 \\ \hline \end{array}$		G ₃ = 0	
	3	13	$\begin{array}{ c c } \hline 0001 & 0011 \\ \hline \end{array}$			
SKMBZ		01 1 2 3	$\begin{array}{ c c } \hline 0000 & 0001 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0001 & 0001 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0000 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0001 & 0011 \\ \hline \end{array}$		RAM(B) ₀ = 0 RAM(B) ₁ = 0 RAM(B) ₂ = 0 RAM(B) ₃ = 0	Skip if RAM Bit is Zero
SKT		41	$\begin{array}{ c c } \hline 0100 & 0001 \\ \hline \end{array}$		A time-base counter carry has occurred since last test	Skip on Timer (Note 2)
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0010 & 1010 \\ \hline \end{array}$	G \rightarrow A	None	Input G Ports to A
ININ		33 28	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0010 & 1000 \\ \hline \end{array}$	IN \rightarrow A	None	Input IN Inputs to A
INIL		33 29	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0010 & 1001 \\ \hline \end{array}$	IL ₃ , CKO, "0", IL ₀ \rightarrow A	None	Input IL Latches to A (Note 2)
INL		33 2E	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0010 & 1110 \\ \hline \end{array}$	L _{7:4} \rightarrow RAM(B) L _{3:0} \rightarrow A	None	Input L Ports to RAM, A
OBD		33 3E	$\begin{array}{ c c } \hline 0011 & 0011 \\ \hline \end{array}$ $\begin{array}{ c c } \hline 0011 & 1110 \\ \hline \end{array}$	Bd \rightarrow D	None	Output Bd to D Outputs

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description				
INPUT/OUTPUT INSTRUCTIONS (Continued)										
OGI	y	33 5-	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0101</td><td>y</td></tr></table>	0011	0011	0101	y	$y \rightarrow G$	None	Output to G Ports Immediate
0011	0011									
0101	y									
OMG		33 3A	<table border="1"><tr><td>0011</td><td>0011</td></tr><tr><td>0011</td><td>1010</td></tr></table>	0011	0011	0011	1010	$RAM(B) \rightarrow G$	None	Output RAM to G Ports
0011	0011									
0011	1010									
XAS		4F	<table border="1"><tr><td>0100</td><td>1111</td></tr></table>	0100	1111	$A \leftrightarrow SIO, C \rightarrow SKL$	None	Exchange A with SIO (Note 2)		
0100	1111									

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A₃ indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: LBI is a single-byte instruction if $d = 0, 9, 10, 11, 12, 13, 14,$ or 15 . The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001₂), the lower 4 bits of the LBI instruction equal 8 (1000₂). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111₂).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds to the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selection Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP404LSN-5 programs.

XAS INSTRUCTIONS

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, PC_{10:8}, A, M. PC₁₀, PC₉ and PC₈ are not affected by this instruction.

Note: JID requires 2 instruction cycles to execute.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL₃ and IL₀ (see Figure 7) and CKO into A. The IL₃ and IL₀ latches are set if a low-going pulse ("1" to "0") has occurred on the IN₃ and IN₀ inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL₃ and IL₀ into A₃ and A₀ respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN₃ and IN₀ lines. INIL will input "1" into A₂ on the COP404LSN-5. A "0" is always placed in A₁ upon the execution of an INIL. The general purpose inputs IN₃-IN₀ are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is use-

ful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC₁₀, PC₉, PC₈, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → PC_{7:4}, RAM(B) → PC_{3:0}, leaving PC₁₀, PC₉ and PC₈ unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC).

Note: LQID takes two instruction cycle times to execute.

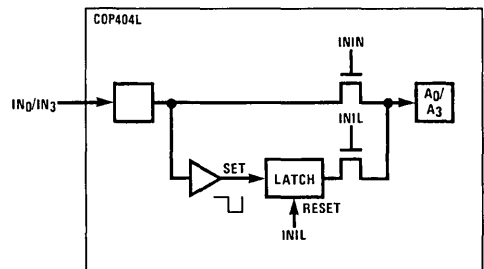


FIGURE 7. INIL Hardware Implementation

TL/DD/8617-12

Description of Selected Instructions (Continued)

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP404LSN-5 to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz oscillator as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency \div 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

- The first word of a COP404LSN-5 program (ROM address 0) must be a CLRA (Clear A) instruction.
- Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

- The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23 or 27 will access data in the next group of four pages.

Typical Applications

PROM-BASED SYSTEM

The COP404LSN-5 may be used to exactly emulate the COP444L. *Figure 8* shows the interconnect to implement a COP444L hardware emulation. This connection uses a MM2716 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/DAT \bar{A} clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74LS373) latches the addresses to drive the memory.

When AD/DAT \bar{A} turns off, the EPROM is enabled and the IP7–IP0 pins will input the memory data. P8, P9 and SKIP/P10 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

The other 28 pins of the COP404LSN-5 may be configured exactly the same as a COP444L. The COP404LSN-5 V_{CC} can vary from 4.5V to 5.5V. However, 5V is used for the memory.

For In-Circuit emulation, see also COP444LP.

COP404LSN-5 Mask Options

The following COP444L options have been implemented on the COP404LSN-5.

Option Value	Comment	Option Value	Comment
Option 1 = 0	Ground, no option available	Option 18 = 2	SK has push-pull output
Option 2 = 0	CKO is clock generator output to crystal/resonator	Option 19 = 0	IN0 has load device to V_{CC}
Option 3 = 0	CKI is oscillator input (divide by 32)	Option 20 = 0	IN3 has load device to V_{CC}
Option 4 = 0	\overline{RESET} pin has load device to V_{CC}	Option 21 = 0	G ₀ } have high current
Option 5 = 2	L ₇ } have LED direct-drive	Option 22 = 0	G ₁ } standard output
Option 6 = 2	L ₆ } output	Option 23 = 0	G ₂ }
Option 7 = 2	L ₅ }	Option 24 = 0	G ₃ }
Option 8 = 2	L ₄ }	Option 25 = 0	D ₃ } have high current
Option 9 = 0	IN1 has load device to V_{CC}	Option 26 = 0	D ₂ }
Option 10 = 0	IN2 has load device to V_{CC}	Option 27 = 0	D ₁ } standard output
Option 11 = 1	V_{CC} 4.5V to 5.5V operation	Option 28 = 0	D ₀ }
Option 12 = 2	L ₃ } have LED direct-drive	Option 29 = 1	L } have higher voltage
Option 13 = 2	L ₂ } output	Option 30 = 1	IN } input levels
Option 14 = 2	L ₁ }	Option 31 = 1	G } SI has standard input level
Option 15 = 2	L ₀ }	Option 32 = 0	\overline{RESET} has Schmitt trigger input
Option 16 = 0	SI has load to V_{CC}	Option 33 = 0	CKO has standard input levels
Option 17 = 2	SO has push-pull output	Option 34 = 0	40-pin package
		Option 35 = N/A	

Typical Applications (Continued)

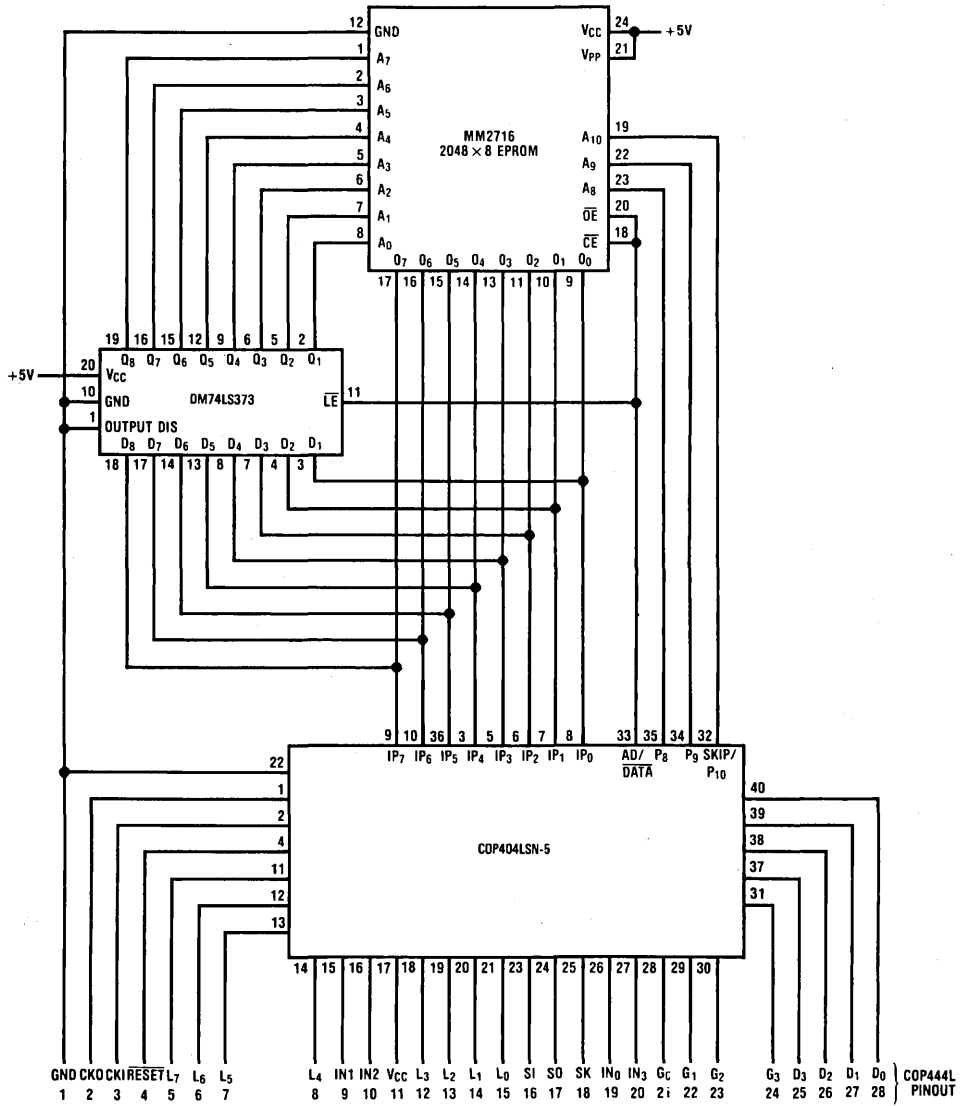


FIGURE 8. COP404LSN-5 System Diagram

TL/DD/8817-13

COP420P/COP444CP/COP444LP Piggyback EPROM Microcontrollers

General Description

The COP420P, COP444CP, and COP444LP are piggyback versions of the COPSM microcontroller families. These devices are identical to their respective device except the program ROM has been removed. The device package incorporates the circuitry and socket on top of package to accommodate the piggyback EPROM—MM2716, NMC27C16 or other appropriate EPROMs. With the addition of an EPROM, the device performs exactly as its masked equivalent.

The device is a complete microcontroller system with CPU, RAM, I/O and EPROM socket in a 28-lead package. The completed package allows field test of the system in the final electrical and mechanical configuration. This important benefit facilitates development and debug of the COP400 program prior to masking of a production part.

These devices are also economical in low and medium volume applications or when the program may require changing.

Device Selection	Device Emulated	Piggyback Device
Low Power NMOS	COP420L, COP444L	COP444LP
High Speed NMOS	COP420	COP420P
Low Power CMOS	COP424C, COP444C	COP444CP

Features

COP444LP

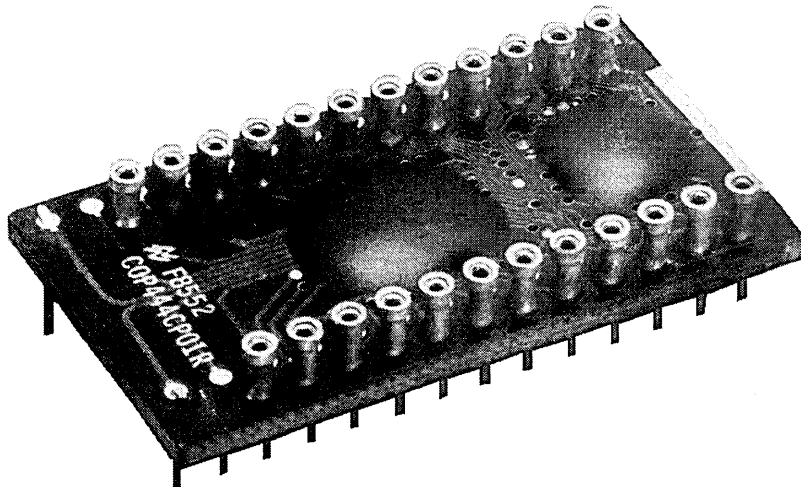
- 16 μ s instruction time
- Same Specification as COP404LSN-5

COP420P

- 4 μ s instruction time
- Same Specification as COP402N

COP444CP

- 4 μ s instruction time
- Fully static (can turn off clock)
- Power-saving IDLE state and Halt mode
- Same Specification as COP404C



TL/DD/8705-10

COP420P Absolute Maximum Ratings

Voltage at Any Pin	-0.3V to +7V
Operating Temperature Range COP420P	0°C to 70°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Total Sink Current	50 mA
Total Source Current	70 mA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP420P DC Electrical Characteristics

0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	5.5	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	All Outputs Open		81	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High	V _{CC} = 5.5V	3.0		V
Logic High	V _{CC} = 4.5V	2.0		V
Logic Low		-0.3	0.4	V
Schmitt Trigger Input				
RESET				
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
All Other Inputs				
Logic High	V _{CC} = Max	3.0		V
Logic High	V _{CC} = 5V ± 10%	2.0		V
Logic Low		-0.3	0.8	V
Input Load Source Current	V _{CC} = 5V, V _{IN} = 0V	-100	-800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage		-1	+1	μA
Output Voltage Levels				
D, G, L, SK, SO Outputs				
TTL Operation	V _{CC} = 5V ± 10%			
Logic High	I _{OH} = -100 μA	2.0		V
Logic Low	I _{OL} = 1.6 mA	-0.3	0.4	V
IP0-IP7, P8, P9, SKIP, CKO, AD/DATA				
Logic High	I _{OH} = -75 μA	2.4		V
Logic Low	I _{OL} = 400 μA	-0.3	0.4	V
CMOS Operation (Note 2)				
Logic High	I _{OH} = -10 μA	V _{CC} - 1		V
Logic Low	I _{OL} = 10 μA	-0.3	0.2	V
Output Current Levels				
LED Direct Drive (Note 3)	V _{CC} = 5.0V			
Logic High	V _{OH} = 2.0V	1.0	14	mA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			-15	mA
Per Pin (All Others)			-1.5	mA

COP420P AC Electrical Characteristics0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	÷ 16 Mode	1.6	4.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time	Frequency = 4 MHz		60	ns
Fall Time	Frequency = 4 MHz		40	ns
Inputs				
SI				
t _{SETUP}		0.3		μs
t _{HOLD}		250		ns
All Other Inputs				
t _{SETUP}		1.7		μs
t _{HOLD}		300		ns
Output Propagation Delay	R _L = 5k, C _L = 50 pF, V _{OUT} = 1.5V			
SO and SK				
t _{pd1}			1.0	μs
t _{pd0}			1.0	μs
CKO				
t _{pd1}			0.25	μs
t _{pd0}			0.25	μs
AD/DATA, SKIP				
t _{pd1}			0.6	μs
t _{pd0}			0.6	μs
All Other Outputs				
t _{pd1}			1.4	μs
t _{pd0}			1.4	μs

Note 1: Duty cycle = t_{W1} / (t_{W1} + t_{W0}).**Note 2:** Voltage change must be less than 0.5V in a 1 ms period.**Note 3:** Exercise great care not to exceed maximum device power dissipation limits when direct driving LEDs (or sourcing similar loads) at high temperature.

COP444CP Absolute Maximum Ratings

Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to 70°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP444CP DC Electrical Characteristics

0°C < T_A < 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Power Supply Ripple (Note 3)	Peak to Peak		0.1 V_{CC}	V
Supply Current (Note 1)	$V_{CC} = 5V, t_C = 4 \mu s$		15	mA
Input Voltage Levels				
RESET, D0				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	$V_{CC} = 4.5V, V_{IN} = 0$	30	330	μA
Hi-Z Input Leakage		-1	+1	μA
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	Standard Outputs			
Logic High	$V_{CC} = 5.0V \pm 5\%$	2.7		V
Logic Low	$I_{OH} = -100 \mu A$		0.4	V
CMOS Operation	$I_{OL} = 400 \mu A$			
Logic High	$I_{OH} = -10 \mu A$	$V_{CC} - 0.2$		V
Logic Low	$I_{OL} = 10 \mu A$		0.2	V
Output Current Levels				
Sink (Note 6)	$V_{CC} = 4.5V, V_{OUT} = V_{CC}$	1.2		mA
Source (Standard Option)	$V_{CC} = 4.5V, V_{OUT} = 0V$	0.5		mA
Source (Low Current Option)	$V_{CC} = 4.5V, V_{OUT} = 0V$	30	330	μA
Allowable Sink/Source Current Per Pin (Note 4)			5	mA
Allowable Loading on CKOH			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	$V_{CC} = 4.5V, V_{IN} = 2 V_{CC}$		0.7	mA
To Halt	$V_{CC} = 4.5V, V_{IN} = 7 V_{CC}$		1.6	mA
TRI-STATE Leakage Current		-2.5	+2.5	μA

COP444CP AC Electrical Characteristics0°C < T_A < 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t _C)	V _{CC} ≥ 4.5V	4	DC	μs
Operating CKI Frequency	V _{CC} ≥ 4.5V	DC	1.0	MHz
Inputs				
t _{SETUP}	G Inputs }	t _C /4 + 0.7		μs
	SI Input } V _{CC} ≥ 4.5V	0.3		μs
	IP Input }	1.0		μs
	All Others }	1.7		μs
t _{CLOCK}	V _{CC} ≥ 4.5V	0.25		μs
Output Propagation Delay	V _{OUT} = 1.5V, C _L = 100 pF, R _L = 5k			
IP7–IP0, A10–A8, SKIP				
t _(pd1) , T _(pd0)	V _{CC} ≥ 4.5V		1.94	μs
AD/DATA				
t _(pd1) , t _(pd0)	V _{CC} ≥ 4.5V		375	μs
All Other Outputs				
t _(pd1) , t _(pd0)	V _{CC} > 4.5V		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI and all other pins pulled up to V_{CC} with 20k resistors.

Note 2: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 3: Voltage change must be less than 0.5V in a 1 ms period.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} (i.e., 0.1 mA at 2.4V V_{CC} and 0.5 mA at 4.5V V_{CC}).

COP444LP Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	-0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C

Total Source Current	120 mA
Total Sink Current	140 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

COP444LP DC Electrical Characteristics

0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V _{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current			66	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V _{IH})	V _{CC} = 5.5V	3.0		V
Logic High (V _{IH})	V _{CC} = 4.5V	2.0		V
Logic Low (V _{IL})		-0.3	0.4	V
RESET Input Levels	Schmitt Trigger Input			
Logic High		0.7 V _{CC}		V
Logic Low		-0.3	0.6	V
IP0-IP7, SI Input Levels				
Logic High	*V _{CC} = 5.5V	2.4		V
Logic High	V _{CC} = 5V ± 5%	2.0		V
Logic Low		-0.3	0.8	V
All Other Inputs				
Logic High	High Trip Level Options	3.6		V
Logic Low		-0.3	1.2	V
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	V _{CC} = 5V ± 5%			
Logic High (V _{OH})	I _{OH} = 25 μA	2.7		V
Logic Low (V _{OL})	I _{OL} = 0.36 mA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I _{OL})	*V _{CC} = 4.5V, V _{OL} = 0.4V	0.9		mA
L0-L7 Outputs	*V _{CC} = 4.5V, V _{OL} = 0.4V	0.4		mA
G0-G3 and D0-D3 Outputs	*V _{CC} = 4.5V, V _{OL} = 1.0V	7.5		mA
CKO	*V _{CC} = 4.5V, V _{OL} = 0.4V	0.2		mA
Output Source Current				
D0-D3, G0-G3 Outputs (I _{OH})	*V _{CC} = 4.5V, V _{OH} = 2.0V	-30	-250	μA
SO and SK Outputs (I _{OH})	*V _{CC} = 4.5V, V _{OH} = 1.0V	1.2		mA
L0-L7 Outputs	*V _{CC} = 4.5V, V _{OH} = 2.0V	-1.4	-20	mA
Input Load Source Current (I _{IL})	V _{CC} = 5.0V, V _{IL} = 0V	-10	-140	μA
Total Sink Current Allowed				
All Outputs Combined			140	mA
D, G Ports			120	mA
L7-L4			4	mA
L3-L0			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L7-L4			60	mA
L3-L0			60	mA
Each L Pin			30	mA
All Other Pins			1.4	mA

COP444LP AC Electrical Characteristics0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		16	40	μs
CKI				
Input Frequency f _i	÷ 32 mode	0.8	2.0	MHz
Duty Cycle		30	60	%
Rise Time	f _i = 2.0 MHz		120	ns
Fall Time			80	ns
Inputs				
SI, IP7–IP0			2.0	μs
t _{SETUP}			1.0	μs
t _{HOLD}				
IN3–IN0, G3–G0, L7–L0			8.0	μs
t _{SETUP}			1.3	μs
t _{HOLD}				
Output Propagation Delay	C _L = 50 pF, V _{OUT} = 1.5V			
SO, SK Outputs	R _L = 20 kΩ		4.0	μs
t _{pd1} , t _{pd0}				
D3–D0, G3–G0, L7–L0	R _L = 20 kΩ		5.6	μs
t _{pd1} , t _{pd0}				
A0–A7			7.5	μs
t _{pd1} , t _{pd0}			11.5	μs
A8, A9				
t _{pd1} , t _{pd0}			6.0	μs
A10				
t _{pd1} , t _{pd0}				

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

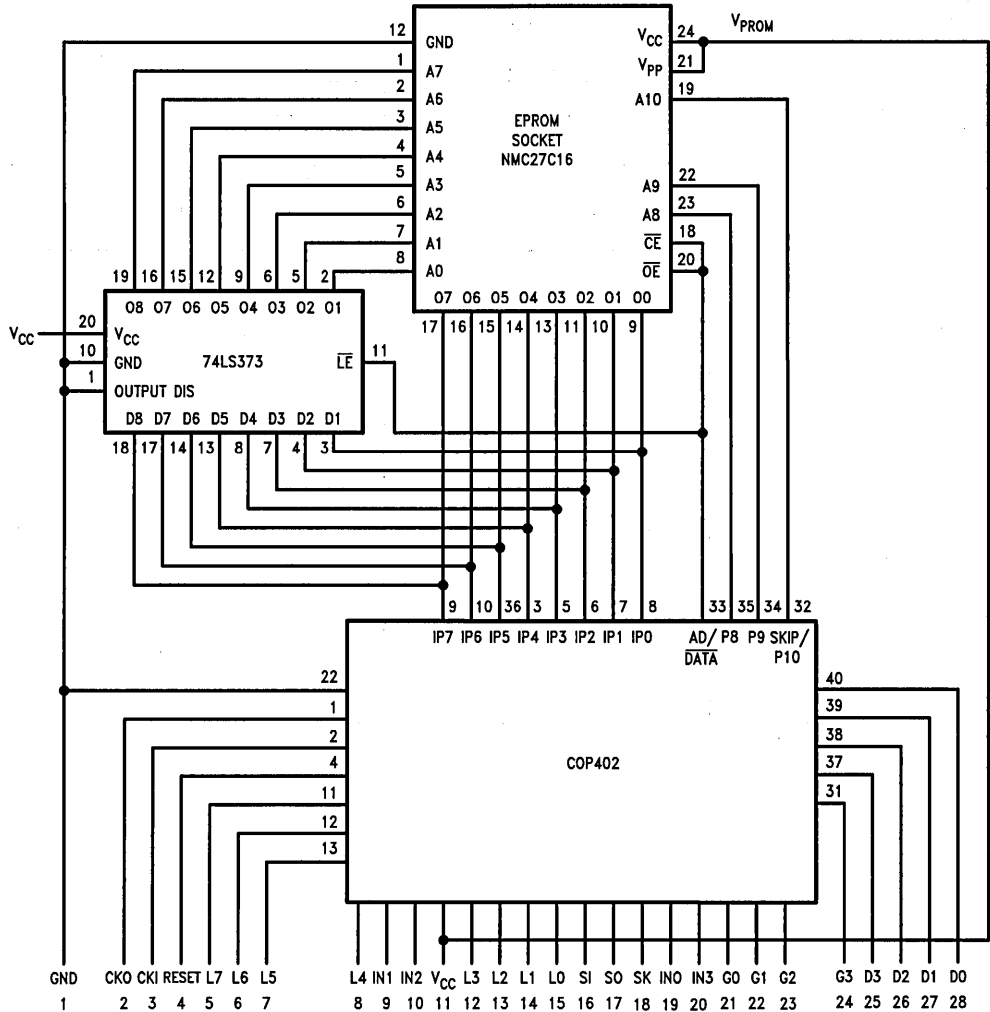


FIGURE 1. COP420P Block Diagram

TL/DD/8705-1

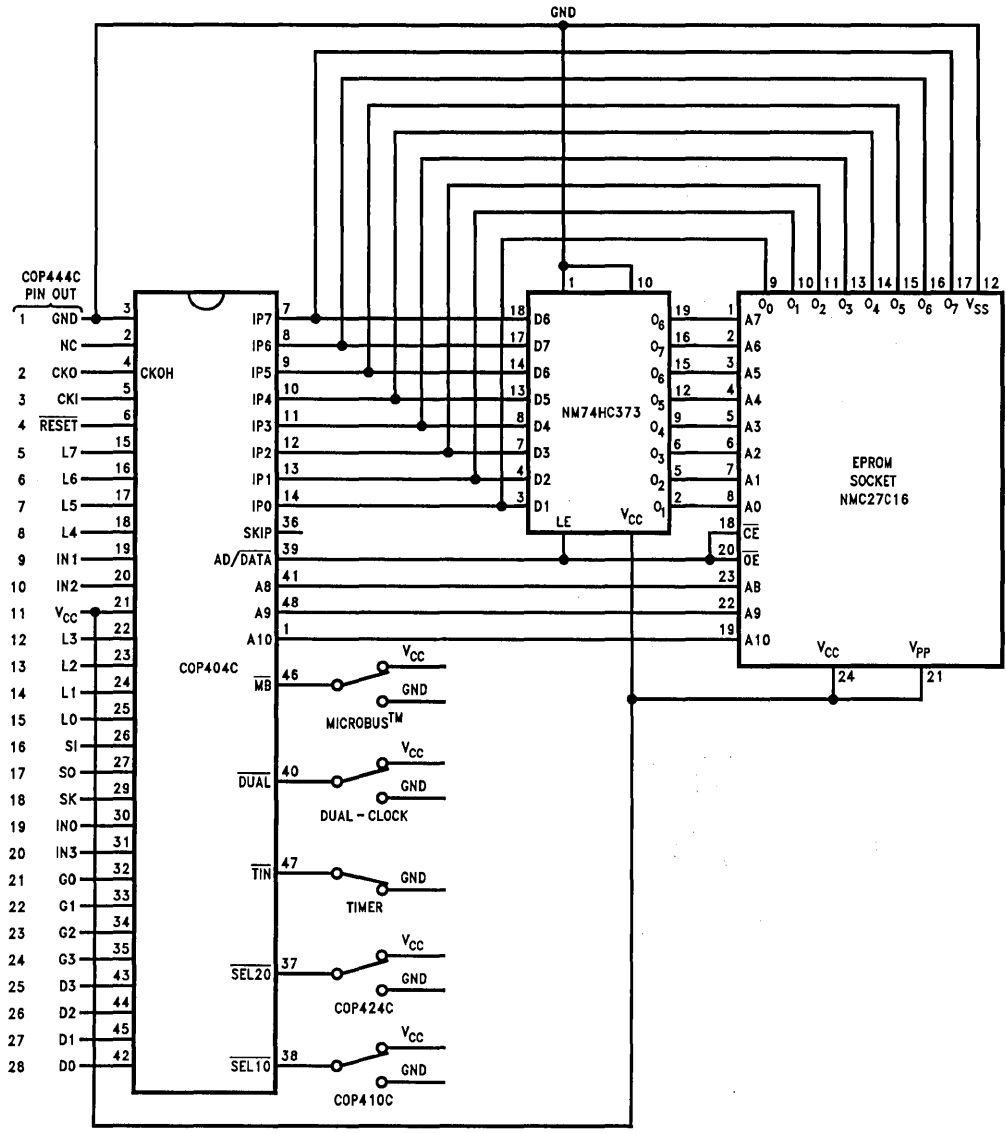


FIGURE 2. COP444CP Block Diagram

TL/DD/8705-2

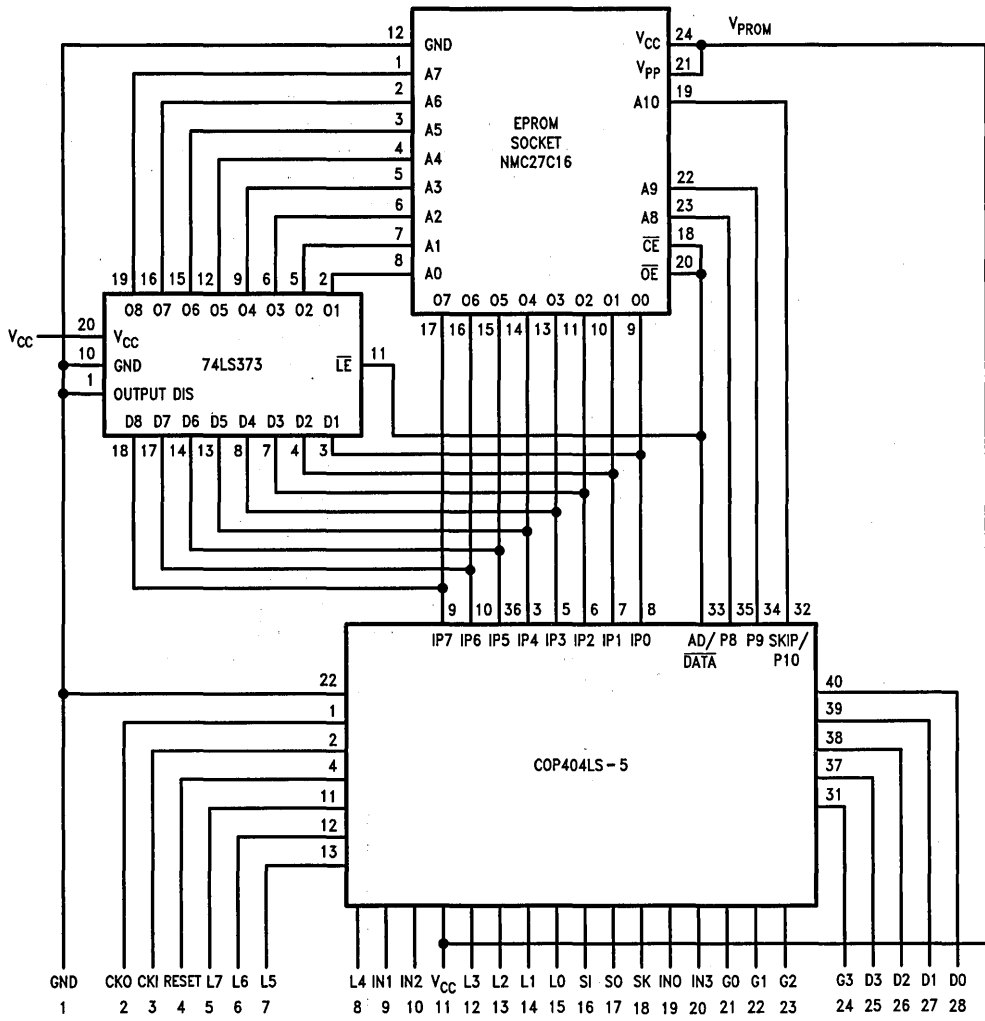
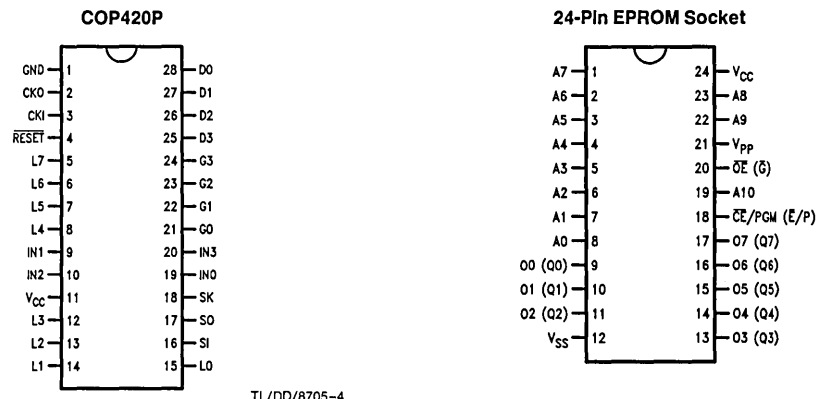


FIGURE 3. COP444LP Block Diagram

TL/DD/8705-3

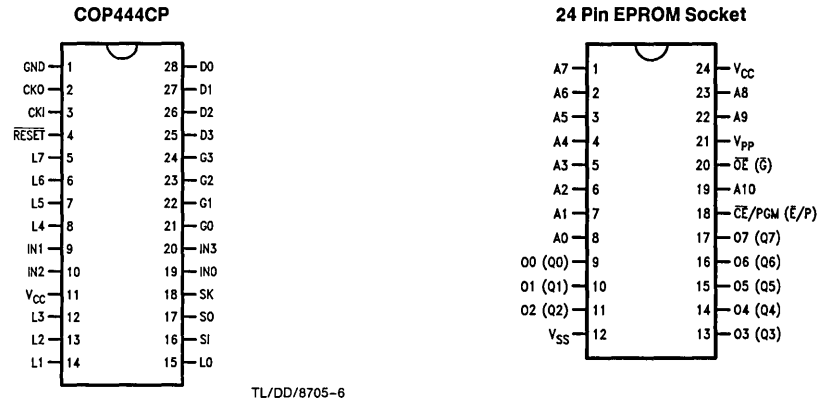
Connection Diagrams



TL/DD/8705-4
FIGURE 4. COP420P Connection Diagrams

Pin	Description
L7-L0	8 Bidirectional I/O Ports with TRI-STATE
G3-G0	4 Bidirectional I/O Ports
D3-D0	4 General Purpose Outputs
IN3-IN0	4 General Purpose Inputs
SI	Serial Input (or Counter Input)
SO	Serial Output (or General Purpose Output)
SK	Logic-Controlled Clock (or General Purpose Output)

Pin	Description
AD/DATA	Address Out/Data In Flag
CKI	System Oscillator Input
CKO	Clock Generator Output to Crystal/Resonator
RESET	System Reset Input
V _{CC}	Power Supply
GND	Ground
O7-O0	PROM Data Lines
A9-A0	PROM Address Outputs



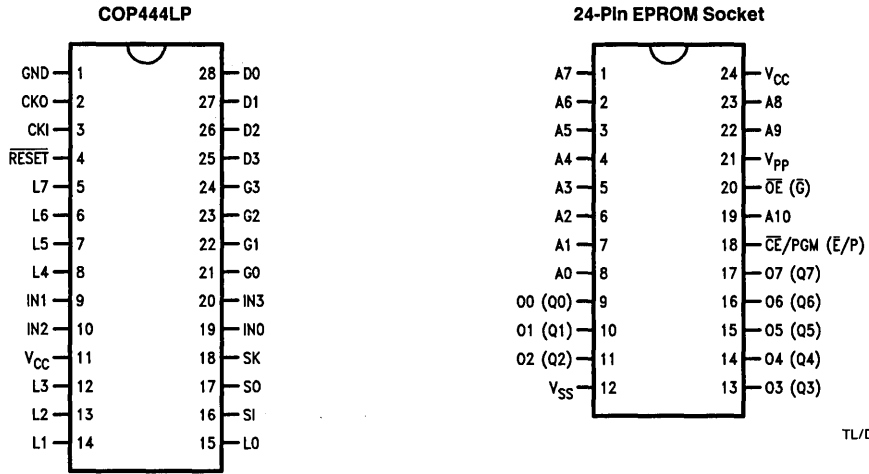
TL/DD/8705-6
FIGURE 5. COP444CP Connection Diagrams

Pin	Description
L7-L0	8 Bidirectional I/O Ports with TRI-STATE
G3-G0	4 Bidirectional Very High Current Standard Output
D3-D0	4 General Very High Current Standard Output
IN3-IN0	4 General Purpose Inputs
SI	Serial Input (or Counter Input)
SO	Serial Output (or General Purpose Output)
SK	Logic-Controlled Clock (or General Purpose Output)

Pin	Description
AD/DATA	Address Out/Data In Flag
CKI	System Oscillator Input
CKO	Clock Generator Output to Crystal/Resonator
RESET	System Reset Input
V _{CC}	Power Supply
GND	Ground
O7-O0	PROM Data Lines
A10-A0	PROM Address Outputs



Connection Diagrams (Continued)



TL/DD/8705-8

FIGURE 6. COP444LP Connection Diagrams

TL/DD/8705-9

Pin	Description	Pin	Description
L ₇ -L ₀	8 LED Direct Drive	AD/ <u>DATA</u>	Address Out/Data In Flag
G ₃ -G ₀	4 Bidirectional Low Current I/O Ports	CKI	System Oscillator Input
D ₃ -D ₀	4 General Purpose Outputs	CKO	Clock Generator Output to Crystal/Resonator
IN ₃ -IN ₀	4 General Purpose Inputs	RESET	System Reset Input
SI	Serial Input (or Counter Input)	V _{CC}	Power Supply
SO	Serial Output (or General Purpose Output)	GND	Ground
SK	Logic-Controlled Clock (or General Purpose Output)	O ₇ -O ₀	PROM Data Lines
		A ₁₀ -A ₀	PROM Address Outputs

COP420 (COP444LP) Mask Options

The following COP420 (COP444L) options have been implemented in the COP420P (COP444LP):

Option Value	Comment
Option 1 = 0	GND pin—no option available
Option 2 = 0	CKO is clock generator output to crystal
Option 3 = 0	CKI is crystal input ÷ 16 (÷ 32 COP444LP)
Option 4 = 0	RESET pin has load device to V _{CC}
Option 5–8 = 2	L outputs have LED direct-drive
Option 9 = 0	IN1 has load device to V _{CC}
Option 10 = 0	IN2 has load device to V _{CC}
Option 11 = 0 (COP420P)	V _{CC} pin—no option available
(Option 11 = 1 COP444LP)	V _{CC} pin—4.5V–5.5V operation
Option 12–15 = 2	L outputs have LED direct-drive
Option 16 = 0	SI has load device to V _{CC}
Option 17 = 2	SO has push-pull output
Option 18 = 2	SK has push-pull output
Option 19 = 0	IN0 has load device to V _{CC}
Option 20 = 0	IN3 has load device to V _{CC}
Option 21–24 = 0	G outputs are standard (COP420P). G outputs have very high current standard output (COP444LP)
Option 25–28 = 0	D outputs are standard (COP420P). D outputs have very high current standard output (COP444LP)
Option 29 = 0 (COP420P)	Normal operation
(Option 29 = 1 COP444LP)	L has higher voltage input levels
Option 30 = 0 (COP420P)	28-pin package
(Option 30 = 1 COP444LP)	IN has higher voltage input levels
Option 31 = 0 (COP420P)	IN has standard input levels
(Option 31 = 1 COP444LP)	G has higher voltage input levels
Option 32 = 0	G has standard input levels (COP420P). SI has standard input levels (COP444LP)
Option 33 = 0	L has standard input levels (COP420P). RESET has Schmitt trigger input (COP444LP)
Option 34 = 0	No option
Option 35 = 0	SI has standard input levels (COP420P). 28-pin package (COP444LP)

COP444CP Mask Options

The following COP444C options have been implemented in the COP444CP:

Option Value	Comment
Option 1 = 0	GND pin—no option available
Option 2 = 1	CKO is HALT I/O
Option 3 = 5	CKI is external clock input ÷ 4
Option 4 = 1	RESET is Hi-Z input
Option 5–8 = 0	L outputs are standard TRI-STATE
Option 9 = 1	IN1 is a Hi-Z input
Option 10 = 1	IN2 is a Hi-Z input
Option 11 = 0	V _{CC} pin (4.5V–5.5V)
Option 12–15 = 0	L outputs are standard TRI-STATE
Option 16 = 1	SI is a Hi-Z input
Option 17 = 0	SO is a standard output
Option 18 = 0	SK is a standard output
Option 19 = 1	IN0 is a Hi-Z input
Option 20 = 1	IN3 is a Hi-Z input
Option 21–24 = 1	G outputs are low current
Option 25–28 = 0	D outputs are standard
Option 29 = 1	No internal initialization logic
Option 30 = 0	Normal operation
Option 31 = 0	Time-base counter
Option 32 = 0	Normal
Option 33 = 0	28-pin package

Note: The COP404C die used for the COP444CP is configured to support 2K ROM, 128 x 4 RAM, D0 as a normal output, and CKO as the HALT restart.



Section 2
COP800 Family



Section 2 Contents

COP800 Family	2-3
COP620C/COP622C/COP640C/COP642C/COP820C/COP822C/COP840C/COP842C/ COP920C/COP922C/COP940C/COP942C Single-Chip microCMOS Microcontrollers	2-5
COP820CJ/COP822CJ/COP823CJ Single-Chip microCMOS Microcontrollers	2-27
COP8640C/COP8642C/COP8620C/COP8622C/COP86L20C/COP86L22C/COP86L40C/ COP86L42C Single-Chip microCMOS Microcontrollers	2-53
COP680C/COP681C/COP880C/COP881C/COP980C/COP981C Microcontrollers	2-75
COP688CL/COP684CL/COP888CL/COP884CL/COP988CL/COP984CL Single-Chip microCMOS Microcontrollers	2-98
COP888CF/COP884CF/COP988CF/COP984CF Single-Chip microCMOS Microcontrollers ..	2-133
COP888CG/COP884CG Single-Chip microCMOS Microcontrollers	2-167
COP688EG/COP684EG/COP888EG/COP884EG Single-Chip microCMOS Microcontrollers .	2-203
COP688CS/COP684CS/COP888CS/COP884CS/COP988CS/COP984CS Single-Chip microCMOS Microcontrollers	2-243
COP8780C/COP8781C/COP8782C Single-Chip EPROM/OTP Microcontrollers	2-283
COP842CMH Microcontroller Emulator	2-300
COP880CMH/COP881CMH Microcontroller Emulators	2-307
COP8640CMH/COP8642CMH Microcontroller Emulators	2-316
COP888CLMH Single-Chip microCMOS Microcontroller Emulator	2-325
COP888CFMH Single-Chip microCMOS Microcontroller Emulator	2-334
COP888CGMH/COP884CGMH/COP888EGMH Single-Chip microCMOS Microcontroller Emulators	2-344
COP820CJMH/COP822CJMH Single-Chip microCMOS Microcontrollers	2-354
COP888CSMH microCMOS Microcontroller Emulator	2-363

The 8-Bit COP800 Family: Optimized for Value

National's COP800 family provides cost-effective solutions for feature-rich, 8-bit microcontroller applications.

Key Features

- High-performance 8-bit microcontroller
- Full 8-bit architecture and implementation
- 1 μ s instruction-cycle time
- High code efficiency with single-byte, multiple-function instructions
- UART
- A/D converter
- WATCHDOG™/clock monitor
- Brown Out Detect
- On-chip ROM from 1 kbyte
- On-chip RAM to 192 bytes
- EEPROM
- M²C^{MOS}™ fabrication
- MICROWIRE/PLUS™ serial interface
- ROMless versions available
- Wide operating voltage range: +2.5V to +6V
- Military temp range available: -55°C to +125°C
- MIL-STD-883C versions available
- 20- to 44-pin packages

The COP800 combines a powerful single-byte, multiple-function instruction set with a memory-mapped core architecture similar to the HPC™.

And like the HPC, the COP800 family supports a wide variety of ROM, RAM, I/O and peripheral functions.

The COP800 has an instruction-cycle time of only 1 μ s, and because over 70% of its instruction set is composed of single-cycle, single-byte instructions, the COP800 can deliver exceptional performance for an 8-bit engine.

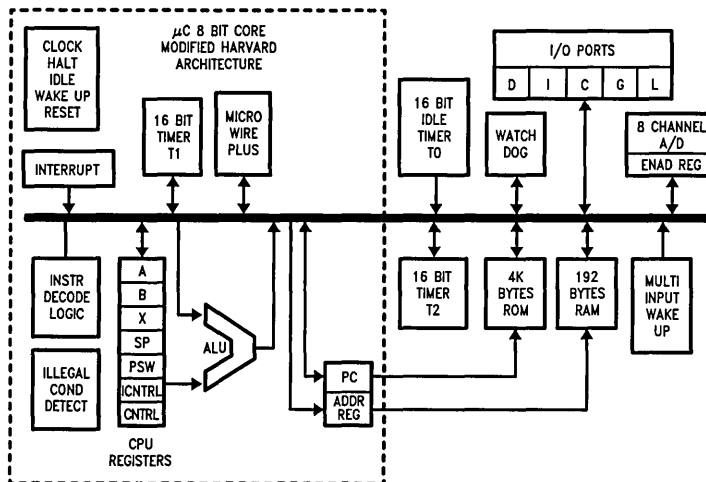
And since it's fabricated in National's advanced M²C^{MOS} process, the COP800 has low current drain, low heat dissipation, and a wide operating voltage range.

Key Applications

- Automotive systems
- Process control
- Robotics
- Telecommunications
- AC-motor control
- DC-motor control
- Keyboard controllers
- Modems
- RS232C controllers

The COP800 family offers high performance in a low-cost, easy-to-design-in package.

COP888CF Block Diagram



TL/XX/0073-3

COP800 Family of Microcontrollers

Commercial Temp Version 0°C to +70°C	Industrial Temp Version -40°C to +85°C	Military Temp Version -55°C to +125°C	Memory		I/O		Features				
			ROM (Bytes)	RAM (Bytes)	I/O Pins	Serial I/O	Interrupt	Stack	Timer Base Counters	Size (Pins)	Other
COP920C COP922C	COP820C COP822C	COP620C COP622C	1.0k 1.0k	64 64	24 16	Yes Yes	3 Sources 3 Sources	EEPROM EEPROM	1 1	28 20	
	COP820CJ COP822CJ COP823CJ		1.0k 1.0k 1.0k	64 64 64	24 16 12	Yes Yes No	3 Sources 3 Sources 3 Sources	In RAM In RAM In RAM	3 3 3	28 20 16	Brown Out, MIWU, & Comp
	COP8640 COP8642 COP8620 COP8622		2.0k 2.0k 1.0k 1.0k	64 64 64 64	24 16 24 16	Yes Yes Yes Yes	3 Sources 3 Sources 3 Sources 3 Sources	In RAM In RAM In RAM In RAM	1 1 1 1	28 20 28 20	64 x 8 EEPROM 64 x 8 EEPROM
COP980C COP981C	COP880C COP881C		4k 4k	128 128	36 24	Yes	3 Sources	In RAM	1	40/44 28	
COP940C COP942C	COP840C COP842C	COP640C COP642C	2.0k 2.0k	128 128	24 16	Yes Yes	3 Sources 3 Sources	In RAM In RAM	1 1	28 20	
	COP8780C COP8781C COP8782C		4.0k 4.0k 4.0k	64 or 128 64 or 128 64 or 128	36 24 16	Yes Yes Yes	3 Sources 3 Sources 3 Sources	In RAM In RAM In RAM	1 1 1	40/44 28 20	OTP or UV Erasable
	COP884CF COP884CG COP884EG COP884CS COP884CL	COP684CF COP684CG COP684EG COP684CS COP684CL	4.0k 4.0k 8.0k 4.0k 4.0k	128 192 256 192 128	21 23 23 23 23	Yes Yes Yes Yes Yes	10 Sources 14 Sources 14 Sources 10 Sources 10 Sources	In RAM In RAM In RAM In RAM In RAM	2 3 3 1 2	28 28 28 28 28	2 PWM & A/D 3 PWM & UART 3 PWM & UART 3 PWM & UART 2 PWM
	COP888CF COP888CG COP888EG COP888CS COP888CL	COP688CF COP688CG COP688EG COP688CS COP688CL	4.0k 4.0k 8.0k 4.0k 4.0k	128 192 256 192 128	33/37 35/39 35/39 35/39 33/39	Yes Yes Yes Yes Yes	10 Sources 14 Sources 14 Sources 10 Sources 10 Sources	In RAM In RAM In RAM In RAM In RAM	2 3 3 1 2	40/44 40/44 40/44 40/44 40/44	2 PWM & A/D 3 PWM & UART 3 PWM & UART 1 PWM & UART 2 PWM

COP620C/COP622C/COP640C/COP642C/ COP820C/COP822C/COP840C/COP842C/ COP920C/COP922C/COP940C/COP942C Single-Chip microCMOS Microcontrollers

General Description

The COP820C and COP840C are members of the COPSTM microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP820C and COP840C to the specific application. The part operates over a voltage range of 2.5 to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate.

Features

- Low Cost 8-bit microcontroller
- Fully static CMOS
- 1 μ s instruction time (10 MHz clock)
- Low current drain (2.2 mA at 3 μ s instruction rate)
Low current static HALT mode (Typically < 1 μ A)
- Single supply operation: 2.5 to 6.0V
- 1024 bytes ROM/64 Bytes RAM—COP820C
- 2048 bytes ROM/128 Bytes RAM—COP840C
- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - Reset master clear
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE/PLUS serial I/O
- 28 pin package (optionally 20 pin package)
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature ranges: 0°C to +70°C, -40°C to +85°C, -55°C to +125°C
- Form Factor emulation devices
- Fully supported by National's development system

Block Diagram

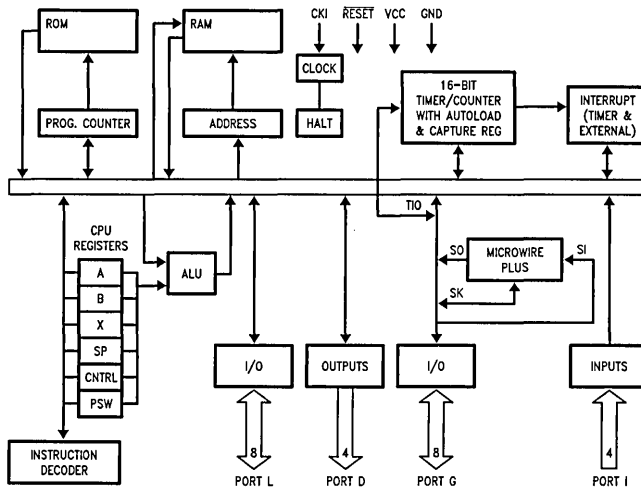


FIGURE 1

TL/DD/9103-1

COP920C/COP922C/COP940C/COP942C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 7V
 Voltage at any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 50 mA

Total Current out of GND Pin (Sink) 60 mA
 Storage Temperature Range $-65^{\circ}C$ to $+140^{\circ}C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP92XC, COP94XC; $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage COP9XXC		2.3		4.0	V
COP9XXCH		4.0		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			$0.1 V_{CC}$	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			7.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			4.0	mA
CKI = 4 MHz	$V_{CC} = 4V, t_c = 2.5 \mu s$			2.0	mA
CKI = 1 MHz	$V_{CC} = 4V, t_c = 10 \mu s$			1.2	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$		<0.7	8.0	μA
	$V_{CC} = 4V, CKI = 0 MHz$		<0.4	5.0	μA
Input Levels					
RESET, CKI					
Logic High		$0.9 V_{CC}$			V
Logic Low				$0.1 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-1		+1	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
G Port Input Hysteresis				$0.35 V_{CC}$	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.3V, V_{OH} = 1.6V$	0.2			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
	$V_{CC} = 2.3V, V_{OL} = 0.4V$	2			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
	$V_{CC} = 2.3V, V_{OH} = 1.6V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.3V, V_{OH} = 1.6V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.3V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage	$V_{CC} = 6.0V$	-1.0		+1.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 4) Without Latchup (Room Temp)	Room Temp			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

COP920C/COP922C/COP940C/COP942C**DC Electrical Characteristics** (Continued)

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Except pin G7: +100 mA, -25 mA (COP920C only). Sampled and not 100% tested. Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

AC Electrical Characteristics 0°C < T_A < +70°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext., Crystal/Resonator (Div-by 10)	$V_{CC} \geq 4.0V$ $2.3V \leq V_{CC} \leq 4.0V$	1 2.5		DC DC	μs μs
R/C Oscillator Mode (Div-by 10)	$V_{CC} \geq 4.0V$ $2.3V \leq V_{CC} \leq 4.0V$	3 7.5		DC DC	μs μs
CKI Clock Duty Cycle (Note 5)	fr = Max	40		60	%
Rise Time (Note 5)	fr = 10 MHz Ext Clock			12	ns
Fall Time (Note 5)	fr = 10 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.0V$ $2.3V \leq V_{CC} \leq 4.0V$	200 500			ns ns
t_{HOLD}	$V_{CC} \geq 4.0V$ $2.3V \leq V_{CC} \leq 4.0V$	60 150			ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK	$C_L = 100 \text{ pF}$, $R_L = 2.2 \text{ k}\Omega$ $V_{CC} \geq 4.0V$ $2.5V \leq V_{CC} \leq 4.0V$			0.7 1.75	μs μs
All Others	$V_{CC} \geq 4.0V$ $2.5V \leq V_{CC} \leq 4.0V$			1 2.5	μs μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_C			
Interrupt Input Low Time		t_C			
Timer Input High Time		t_C			
Timer Input Low Time		t_C			
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).

COP820C/COP822C/COP840C/COP842C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	50 mA

Total Current out of GND Pin (Sink) 60 mA
Storage Temperature Range -65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP82XC, COP84XC: -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage Power Supply Ripple (Note 1)	Peak to Peak	2.5		6.0 0.1 V_{CC}	V V
Supply Current (Note 2) CKI = 10 MHz CKI = 4 MHz CKI = 4 MHz CKI = 1 MHz HALT Current (Note 3)	$V_{CC} = 6V, t_c = 1 \mu s$ $V_{CC} = 6V, t_c = 2.5 \mu s$ $V_{CC} = 4.0V, t_c = 2.5 \mu s$ $V_{CC} = 4.0V, t_c = 10 \mu s$ $V_{CC} = 6V, CKI = 0 MHz$			7.5 4.0 2.0 1.2 10	mA mA mA mA μA
Input Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		0.9 V_{CC} 0.7 V_{CC}		0.1 V_{CC} 0.2 V_{CC}	V V V V
Hi-Z Input Leakage Input Pullup Current	$V_{CC} = 6.0V$ $V_{CC} = 6.0V$	-2 40		+2 250	μA μA
G Port Input Hysteresis			0.35 V_{CC}		V
Output Current Levels D Outputs Source Sink All Others Source (Weak Pull-Up) Source (Push-Pull Mode) Sink (Push-Pull Mode) TRI-STATE Leakage	$V_{CC} = 4.5V, V_{OH} = 3.8V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$ $V_{CC} = 4.5V, V_{OL} = 1.0V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$ $V_{CC} = 4.5V, V_{OH} = 3.2V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$ $V_{CC} = 4.5V, V_{OH} = 3.8V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$ $V_{CC} = 4.5V, V_{OL} = 0.4V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	0.4 0.2 10 2 10 2.5 0.4 0.2 1.6 0.7 -2.0		110 33	mA mA mA mA μA μA mA mA mA μA
Allowable Sink/Source Current Per Pin D Outputs (Sink) All Others				15 3	mA mA
Maximum Input Current (Note 4) Without Latchup (Room Temp)	Room Temp			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Except pin G7: +100 mA, -25 mA (COP820C only). Sampled and not 100% tested. Pins G6 and \overline{RESET} are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

COP820C/COP822C/COP840C/COP842C

AC Electrical Characteristics $-40^{\circ}\text{C} < T_A < +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext. or Crystal/Resonator (Div-by 10) R/C Oscillator Mode (Div-by 10)	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	2.5		DC	μs
	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	7.5		DC	μs
CKI Clock Duty Cycle (Note 6) Rise Time (Note 6) Fall Time (Note 6)	$f_r = \text{Max}$	40		60	%
	$f_r = 10\text{ MHz Ext Clock}$			12	ns
	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs t_{SETUP} t_{HOLD}	$V_{CC} \geq 4.5\text{V}$	200			ns
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	500			ns
	$V_{CC} \geq 4.5\text{V}$	60			ns
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	150			ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$C_L = 100\text{ pF}, R_L = 2.2\text{ k}\Omega$				
	$V_{CC} \geq 4.5\text{V}$			0.7	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$			1.75	μs
	$V_{CC} \geq 4.5\text{V}$			1	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$			2.5	μs
MICROWIRE Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20			ns
		56			ns
				220	ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		t_c			
		t_c			
		t_c			
		t_c			
		t_c			
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).

Timing Diagram

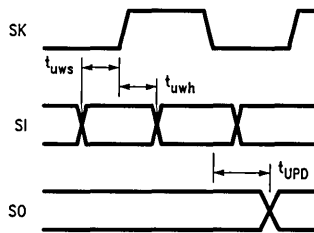


FIGURE 2. MICROWIRE/PLUS Timing

TL/DD/9103-19

COP620C/COP622C/COP640C/COP642C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	40 mA

Total Current out of GND Pin (Sink)	48 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP62XC, COP64XC: -55°C ≤ T_A ≤ +125°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current (Note 2)				7.5	mA
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			4	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu s$			30	μA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0 MHz$		<10		
Input Levels					
RESET, CKI					
Logic High		0.9 V_{CC}		0.1 V_{CC}	V
Logic Low					V
All Other Inputs					
Logic High		0.7 V_{CC}		0.2 V_{CC}	V
Logic Low					V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-5		+5	μA
Input Pullup Current	$V_{CC} = 4.5V$	35		300	μA
G Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.35			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	9			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	9		120	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.35			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.4			mA
TRI-STATE Leakage		-5.0		+5.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				12	mA
All Others				2.5	mA
Maximum Input Current (Room Temp) Without Latchup (Note 5)	Room Temp			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.5			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

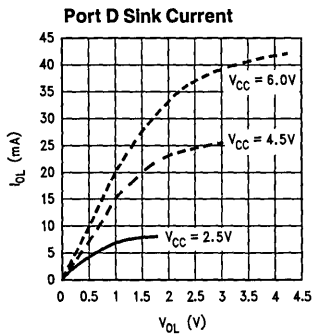
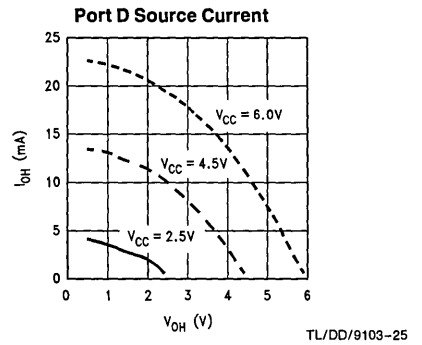
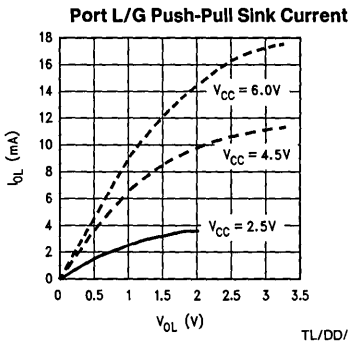
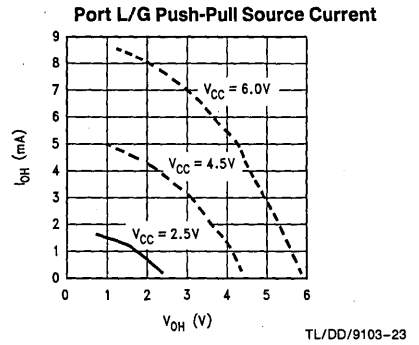
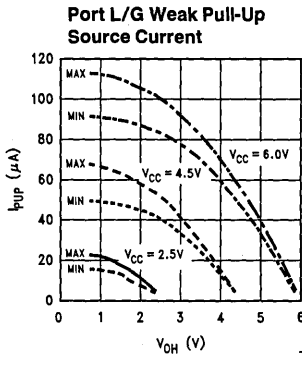
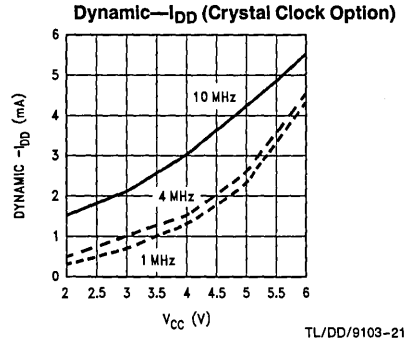
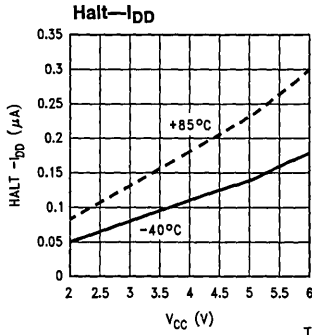
Note 4: Except pin G7: +100 mA, -25 mA (COP620C only). Sampled and not 100% tested. Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

COP620C/COP622C/COP640C/COP642C**AC Electrical Characteristics** $-55^{\circ}\text{C} < T_A < +125^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext. or Crystal/Resonant (Div-by 10)	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
CKI Clock Duty Cycle (Note 6)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 6)	$f_r = 10\text{ MHz Ext Clock}$			12	ns
Fall Time (Note 6)	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5\text{V}$	220			ns
t_{HOLD}	$V_{CC} \geq 4.5\text{V}$	66			ns
Output Propagation Delay	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$				
$t_{\text{PD1}}, t_{\text{PD0}}$	$V_{CC} \geq 4.5\text{V}$			0.8	μs
SO, SK	$V_{CC} \geq 4.5\text{V}$			1.1	μs
All Others	$V_{CC} \geq 4.5\text{V}$				
MICROWIRE Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1			μs

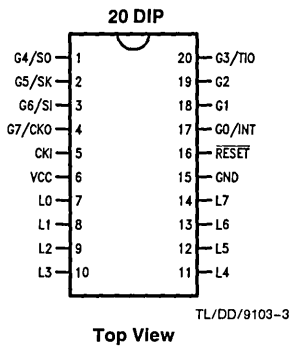
Note 5: Parameter sampled (not 100% tested).

Typical Performance Characteristics ($-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$)

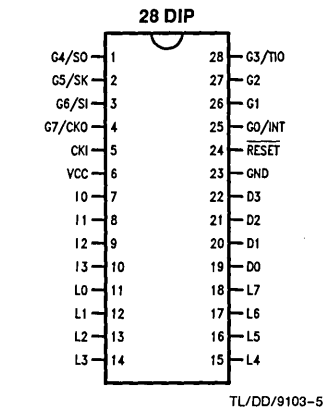


Connection Diagrams

DUAL-IN-LINE PACKAGE

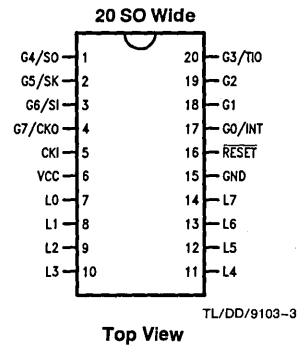


Top View
 Order Number COP622C-XXX/N,
 COP642C-XXX/N, COP822C-XXX/N,
 COP842C-XXX/N, COP922C-XXX/N
 or COP942C-XXX/N
 See NS Package Number N20A

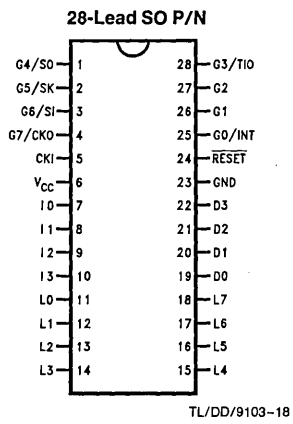


Top View
 Order Number COP620C-XXX/N,
 COP640C-XXX/N, COP820C-XXX/N,
 COP840C-XXX/D, COP920C-XXX/N
 or COP940C-XXX/N
 See NS Package Number N28B

SURFACE MOUNT



Top View
 Order Number COP822C-XXX/WM,
 COP842C-XXX/WM,
 COP922C-XXX/WM or
 COP942C-XXX/WM
 See NS Package Number M20B



Top View
 Order Number COP820C-XXX/WM,
 COP840C-XXX/WM,
 COP920C-XXX/WM or
 COP940C-XXX/WM
 See NS Package Number M28A

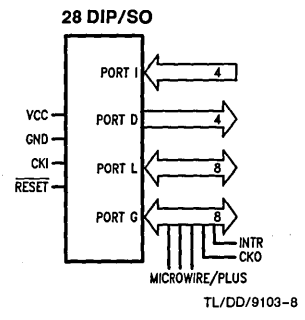
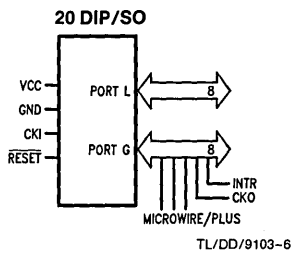


FIGURE 3

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below.

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

- G0 INTR (an external interrupt)
- G3 TIO (timer/counter input/output)
- G4 SO (MICROWIRE serial data output)
- G5 SK (MICROWIRE clock I/O)
- G6 SI (MICROWIRE serial data input)
- G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when **RESET** goes low.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 8-bit Accumulator register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns.

PROGRAM MEMORY

Program memory for the COP820C consists of 1024 bytes of ROM (2048 bytes of ROM for the COP840C). These bytes may hold program instructions or constant data. The program memory is addressed by the 15-bit program counter (PC). ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

The data memory address space includes on chip RAM, I/O and registers. Data memory is addressed directly by the instruction or indirectly by the B, X and SP registers.

The COP820C has 64 bytes of RAM and the COP840C has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" that can be loaded immediately, decremented or tested. Three specific registers: B, X and SP are mapped into this space, the other bytes are available for general usage.

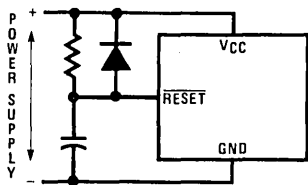
The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except the A & PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested.

RESET

The **RESET** input when pulled low initializes the microcontroller. Initialization will occur whenever the **RESET** input is pulled low. Upon initialization, the ports L and G are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L & G are cleared.

The external RC network shown in Figure 4 should be used to ensure that the **RESET** pin is held low until the power supply to the chip stabilizes.

Functional Description (Continued)



TL/DD/9103-9

$RC \geq 5X$ Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

OSCILLATOR CIRCUITS

Figure 5 shows the three clock oscillator configurations available for the COP820C and COP840C.

A. CRYSTAL OSCILLATOR

The COP820C/COP840C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

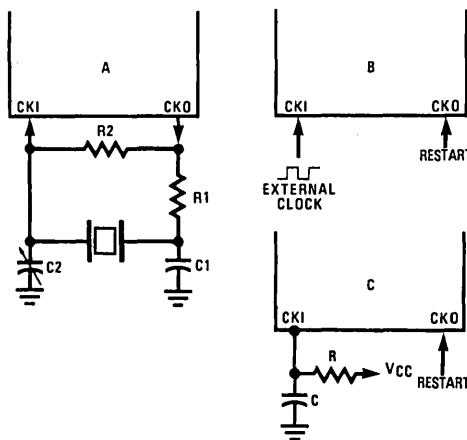
B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9103-10

FIGURE 5. Crystal and R-C Connection Diagrams

OSCILLATOR MASK OPTIONS

The COP820C and COP840C can be driven by clock inputs between DC and 10 MHz.

TABLE I. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4	$V_{CC} = 5V$
0	1	200	100-150	0.455	$V_{CC} = 5V$

TABLE II. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Note: $3k \leq R \leq 200k$, $50 \text{ pF} \leq C \leq 200 \text{ pF}$

Functional Description (Continued)

The COP820C and COP840C microcontrollers have three mask options for configuring the clock input. The CKI and CKO pins are automatically configured upon selecting a particular option.

- Crystal (CKI/10) CKO for crystal configuration
- External (CKI/10) CKO available as G7 input
- R/C (CKI/10) CKO available as G7 input

G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

- 1) Oscillator operating mode—I1
- 2) Internal switching current—I2
- 3) Internal leakage current—I3
- 4) Output source current—I4
- 5) DC current caused by external input not at V_{CC} or GND—I5

Thus the total current drain, It is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

Where

C = equivalent capacitance of the chip.

V = operating voltage

f = CKI frequency

HALT MODE

The COP820C and COP840C support a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to V_r (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the \overline{RESET} or by the CKO pin. A low on the \overline{RESET} line reinitializes the microcontroller and starts executing from the address

0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction. This also resets the G7 data bit.

INTERRUPTS

The COP820C and COP840C have a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer underflow or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

Functional Description (Continued)

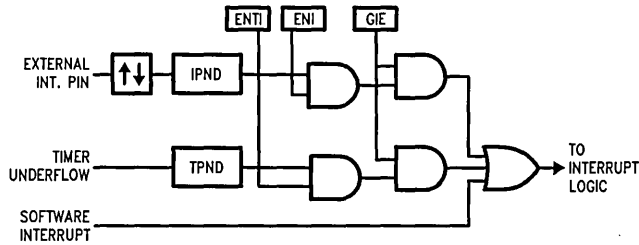


FIGURE 6. Interrupt Block Diagram

TL/DD/9103-11

DETECTION OF ILLEGAL CONDITIONS

The COP820C and COP840C incorporate a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP820C and COP840C grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the COP820C and COP840C to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMS, etc.) and with other microcontrollers which support the MICROWIRE/PLUS interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 7 shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS interface with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS interface with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	2t _C
0	1	4t _C
1	x	8t _C

where,

t_C is the instruction cycle clock.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP820C and COP840C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. Figure 8 shows how two COP820C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP820C. The MICROWIRE/PLUS Master always initiates all data exchanges. (See Figure 8). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The S0 and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summarizes the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by appropriately setting up the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See Figure 8.)

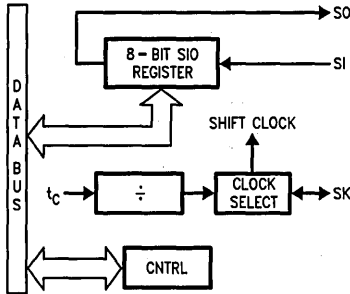
Functional Description (Continued)

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The COP820C and COP840C have a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table V details various timer operating modes and their requisite control settings.



TL/DD/9103-12

FIGURE 7. MICROWIRE/PLUS Block Diagram

MODE 1. TIMER WITH AUTO-LOAD REGISTER

In this mode of operation, the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allows the generation of square-wave outputs or pulse width modulated outputs under software control. (See Figure 9)

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See Figure 9)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See Figure 10.)

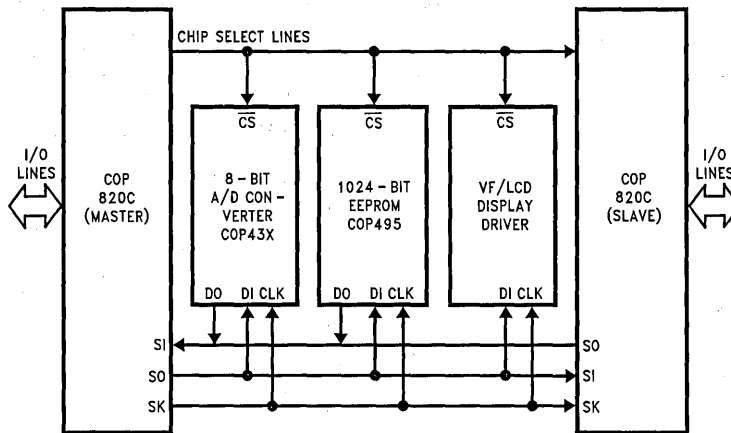


FIGURE 8. MICROWIRE/PLUS Application

TL/DD/9103-13

Functional Description (Continued)

TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t_c
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t_c
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t_c
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t_c

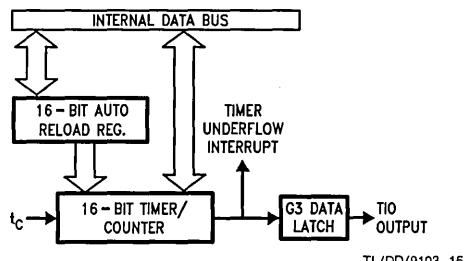


FIGURE 9. Timer/Counter Auto Reload Mode Block Diagram

TL/DD/9103-15

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.

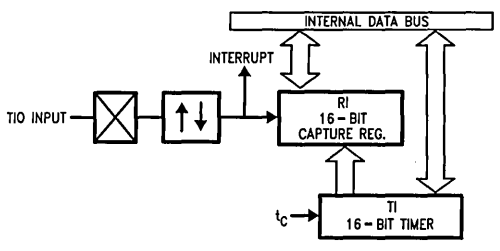


FIGURE 10. Timer Capture Mode Block Diagram

TL/DD/9103-14

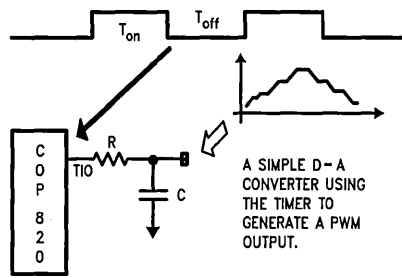


FIGURE 11. Timer Application

TL/DD/9103-16

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE/PLUS control register contains the following bits:

- S1 & S0 Select the MICROWIRE/PLUS clock divide-by
 IEDG External interrupt edge polarity select
 (0 = rising edge, 1 = falling edge)
 MSEL Enable MICROWIRE/PLUS functions SO and SK
 TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)
 TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)
 TC2 Selects the capture mode
 TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0
						BIT 7	BIT 0

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable
 ENI External interrupt enable
 BUSY MICROWIRE/PLUS busy shifting
 IPND External interrupt pending
 ENTI Timer interrupt enable
 TPND Timer interrupt pending
 C Carry Flag
 HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE
						Bit 7	Bit 0

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for COP820C and COP840C. The operand is the memory addressed by the B register or X register.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from -31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
COP820C	
00 to 2F	On Chip RAM Bytes
30 to 7F	Unused RAM Address Space (Reads as all Ones)
COP840C	
00 to 6F	On Chip RAM Bytes
70 to 7F	Unused RAM Address Space (Reads as all Ones)
COP820C and COP840C	
80 to BF	Expansion Space for on Chip EERAM
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8-DB	Reserved for Port C
DC	Port D Data Register
DD-DF	Reserved for Port D
E0 to EF	On Chip Functions and Registers
E0-E7	Reserved for Future Parts
E8	Reserved
E9	MICROWIRE/PLUS Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

- A 8-bit Accumulator register
- B 8-bit Address register
- X 8-bit Address register
- SP 8-bit Stack pointer register
- PC 15-bit Program counter register
- PU upper 7 bits of PC
- PL lower 8 bits of PC
- C 1-bit of PSW register for carry
- HC Half Carry
- GIE 1-bit of PSW register for global interrupt enable

Symbols

- [B] Memory indirectly addressed by B register
- [X] Memory indirectly addressed by X register
- Mem Direct address memory or [B]
- Meml Direct address memory or [B] or Immediate data
- Imm 8-bit Immediate data
- Reg Register memory: addresses F0 to FF (Includes B, X and SP)
- Bit Bit number (0 to 7)
- ← Loaded with
- ↔ Exchanged with

Instruction Set

ADD ADC	add add with carry	A ← A + Meml A ← A + Meml + C, C ← Carry HC ← Half Carry
SUBC	subtract with carry	A ← A + Meml + C, C ← Carry HC ← Half Carry
AND OR XOR	Logical AND Logical OR Logical Exclusive-OR	A ← A and Meml A ← A or Meml A ← A xor Meml
IFEQ IFGT IFBNE DRSZ SBIT	IF equal IF greater than IF B not equal Decrement Reg., skip if zero Set bit	Compare A and Meml, Do next if A = Meml Compare A and Meml, Do next if A > Meml Do next if lower 4 bits of B ≠ Imm Reg ← Reg - 1, skip if Reg goes to 0
RBIT IFBIT	Reset bit If bit	1 to bit, Mem (bit = 0 to 7 immediate) 0 to bit, Mem If bit, Mem is true, do next instr.
X LD A LD mem LD Reg	Exchange A with memory Load A with memory Load Direct memory Immed. Load Register memory Immed.	A ↔ Mem A ← Meml Mem ← Imm Reg ← Imm
X X LD A LD A LD M	Exchange A with memory [B] Exchange A with memory [X] Load A with memory [B] Load A with memory [X] Load Memory Immediate	A ↔ [B] (B ← B ± 1) A ↔ [X] (X ← X ± 1) A ← [B] (B ← B ± 1) A ← [X] (X ← X ± 1) [B] ← Imm (B ← B ± 1)
CLRA INCA DECA LAID DCORA RRCA SWAPA SC RC IFC IFNC	Clear A Increment A Decrement A Load A indirect from ROM DECIMAL CORRECT A ROTATE A RIGHT THRU C Swap nibbles of A Set C Reset C If C If not C	A ← 0 A ← A + 1 A ← A - 1 A ← ROM(PU,A) A ← BCD correction (follows ADC, SUBC) C → A7 → ... → A0 → C A7...A4 ↔ A3...A0 C ← 1, HC ← 1 C ← 0, HC ← 0 If C is true, do next instruction If C is not true, do next instruction
JMPL JMP JP JSRL JSR JID RET RETSK RETI INTR NOP	Jump absolute long Jump absolute Jump relative short Jump subroutine long Jump subroutine Jump indirect Return from subroutine Return and Skip Return from Interrupt Generate an interrupt No operation	PC ← ii (ii = 15 bits, 0 to 32k) PC11..0 ← i (i = 12 bits) PC ← PC + r (r is -31 to +32, not 1) [SP] ← PL, [SP-1] ← PU, SP-2, PC ← ii [SP] ← PL, [SP-1] ← PU, SP-2, PC11..0 ← i PL ← ROM(PU,A) SP+2, PL ← [SP], PU ← [SP-1] SP+2, PL ← [SP], PU ← [SP-1], Skip next instruction SP+2, PL ← [SP], PU ← [SP-1], GIE ← 1 [SP] ← PL, [SP-1] ← PU, SP-2, PC ← 0FF PC ← PC + 1

Bits 7-4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	OPCODE LIST
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0, [B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A, [B]	IFBIT 1, [B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A, [B]	IFBIT 2, [B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X-]	X A, [B-]	IFGT A, #i	IFGT A, [B]	IFBIT 3, [B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A, [B]	IFBIT 4, [B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	AND A, #i	AND A, [B]	IFBIT 5, [B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A, [B]	IFBIT 6, [B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	OR A, #i	OR A, [B]	IFBIT 7, [B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0, [B]	RBIT 0, [B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1, [B]	RBIT 1, [B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2, [B]	RBIT 2, [B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X-]	LD A, [B-]	LD [B-], #i	DECA	SBIT 3, [B]	RBIT 3, [B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	*	SBIT 4, [B]	RBIT 4, [B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP + 13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5, [B]	RBIT 5, [B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP + 14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00-0EFF	JMP 0E00-0EFF	JP + 31	JP + 15	E
JP -0	JP -16	LD 0FF, #1	DRSZ 0FF	*	*	*	RETI	SBIT 7, [B]	RBIT 7, [B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

OPCODE LIST

Bits 3-0

2-22

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr	
	[B]	[X]			[B+, B-]	[X+, X-]
X A,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B,Imm				1/1		
LD B,Imm				2/3		
LD Mem,Imm	2/2		3/3		2/2	
LD Reg,Imm				2/3		

(If B < 16)
(If B > 15)

* => Memory location addressed by B or X or directly.

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Option List

The COP820C/COP840C mask programmable options are listed out below. The options are programmed at the same time as the ROM pattern to provide the user with hardware flexibility to use a variety of oscillator configuration.

OPTION 1: CKI INPUT

- = 1 Crystal (CKI/10) CKO for crystal configuration
- = 2 External (CKI/10) CKO available as G7 input
- = 3 R/C (CKI/10) CKO available as G7 input

OPTION 2: COP820C/COP840C BONDING

- = 1 28 pin package
- = 2 N.A.
- = 3 20 pin package
- = 4 20 SO package
- = 5 28 SO package

The following option information is to be sent to National along with the EPROM.

Option Data

Option 1 Value__is: CKI Input

Option 2 Value__is: COP Bonding

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system.

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™—COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code

address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefineable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information:

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink Base Unit In-circuit Emulator for all COP8 Devices, Symbolic Debugger Software and RS-232 Serial Interface Cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-880C20D5PC	20 DIP	4.5V-5.5V	COP822C, 842C, 8782C
MHW-880C20DWPC	20 DIP	2.5V-6.0V	COP822C, 842C, 8782C
MHW-880C28D5PC	28 DIP	4.5V-5.5V	COP820C, 840C, 881C, 8781C
MHW-880C28DWPC	28 DIP	2.5V-6.0V	COP820C, 840C, 881C, 8781C

Development Support (Continued)

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 Macro Cross Assembler for IBM PC-XT®, PC-AT® or Compatible	424410527-001

SIMULATOR

The COP8 Designer's is Tool Kit is available for evaluating National Semiconductor's COP8 microcontroller family. The kit contains programmer's manuals, device datasheets, pocket reference guide, assembler and simulator, which allows the user to write, test, debug and run code on an industry standard compatible PC. The simulator has a windowed user interface and can handle script files that simulate hard-

ware inputs, interrupts and automatic command processing. The capture file feature enables the user to record to a file current cycle count and output port changes which are caused by the program under test.

Simulator Ordering Information

Part Number	Description	Manual
COP8-TOOL-KIT	COP8 Designer's Tool Kit Assembler and Simulator	420420270-001 424420269-001

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit, and function emulators. Two types of single chip emulators are available: Multi-Chip Module (MCM) emulators, which combine the microcontroller-die and an EPROM-die in one package, and emulators where the microcontroller's standard ROM is replaced with an on-chip EPROM. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP881CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	28 DIP	Multi-Chip Module (MCM), UV Erasable	COP840C, COP820C
COP881CMHEA-X	X = 1: Crystal X = 2: External X = 3: R/C	28 LCC	MCM, (Same Footprint as 28 SO), UV Erasable	COP840C, COP820C
COP842CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	20 DIP	MCM, UV Erasable	COP842C
COP822CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	20 DIP	MCM, UV Erasable	COP822C
COP8781CN	Programmable	28 DIP	One Time Programmable (OTP)	COP840C, COP820C
COP8781CJ	Programmable	28 DIP	UV Erasable	COP840C, COP820C
COP8781CWM	Programmable	28 SO	OTP	COP840C, COP820C
COP8781CMC	Programmable	28 SO	UV Erasable	COP840C, COP820C
COP8782CN	Programmable	20 DIP	OTP	COP842C, COP822C
COP8782CJ	Programmable	20 DIP	UV Erasable	COP842C, COP822C
COP8782CWM	Programmable	20 SO	OTP	COP842C, COP822C
COP8782CMC	Programmable	20 SO	UV Erasable	COP842C, COP822C

Development Support (Continued)

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa.

Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 FAX: (206) 882-1043

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP and for Use with Scrambler Boards	COP881CMHD
COP8-SCRM-DIP	Multi-Chip Module (MCM) Scrambler Board for 20 DIP Socket	COP842CMHD, COP822CMHD
COP8-SCRM-SBX	MCM Scrambler Board for 28 LCC Sockets	COP881CMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP881CMHD, 842CMHD, 822CMHD
COP8-PRGM-87A	Duplicator Board with COP87XX Scrambler for 28 DIP and 28 SO	COP8781CN, 8781CJ, 8781CWM, 8781CMC
COP8-PRGM-87B	Duplicator Board with COP87XX Scrambler for 20 DIP and 20 SO	COP8782CN, 8782CJ, 8782CWM, 8782CMC

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:

Dial-A-Helper Users Manual

Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 baud

Setup: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs. 7 Days

COP820CJ/COP822CJ/COP823CJ

Single-Chip microCMOS Microcontroller

General Description

The COP820CJ is a member of the COPSM 8-bit Microcontroller family. It is a fully static Microcontroller, fabricated using double-metal silicon gate microCMOS technology. This low cost Microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRETM serial I/O, a 16-bit timer/counter with capture register, a multi-sourced interrupt, Comparator, WATCHDOGTM Timer, Modulator/Timer, Brown out protection and Multi-Input Wakeup. Each I/O pin has software selectable options to adapt the device to the specific application. The device operates over a voltage range of 2.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 μ s per instruction rate.

Features

- Low cost 8-bit Microcontroller
- Fully static CMOS
- 1 μ s instruction time
- Low current drain
 - Low current static HALT mode
- Single supply operation: 2.5V to 6.0V
- 1024 x 8 on-chip ROM
- 64 bytes on-chip RAM
- WATCHDOG Timer
- Comparator
- Modulator/Timer (High speed PWM Timer for IR Transmission)
- Multi-Input Wakeup (on the 8-bit Port L)
- Brown Out Protection
- 4 high current I/O pins with 15 mA sink capability
- MICROWIRE/PLUSTSM serial I/O
- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- 28- and 20-pin DIP/SO package or 16-pin SO package
- Software selectable I/O options (TRI-STATE[®], push-pull, weak pull-up)
- Schmitt trigger inputs on Port G and Port L
- Fully supported by National's development system
- Form Factor Emulator

Block Diagram

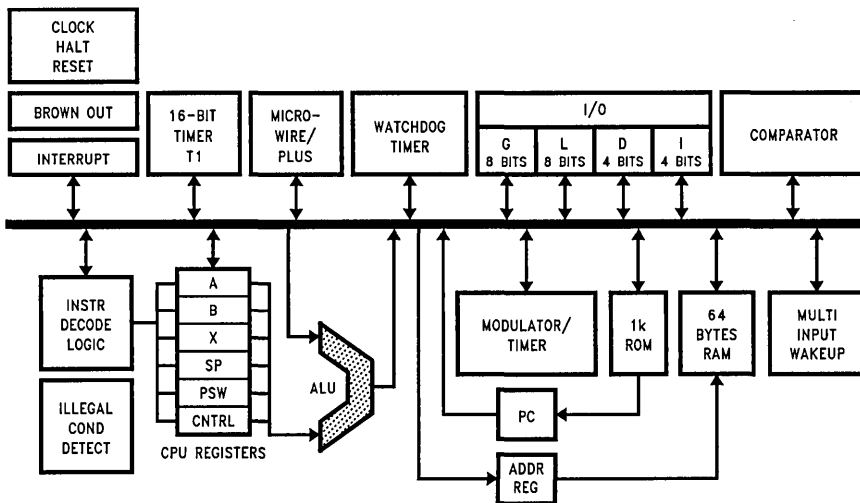


FIGURE 1. COP820CJ Block Diagram

TL/DD/11208-1

COP820CJ/COP822CJ/COP823CJ**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7.0V
Voltage at any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} pin (Source)	80 mA

Total Current out of GND pin (sink) 80 mA
Storage Temperature Range -65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur.

DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage	Brown Out Disabled	2.5		6.0	V
Power Supply Ripple 1 (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			8.0	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			6.0	mA
CKI = 4 MHz	$V_{CC} = 4.0V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 4.0V, t_c = 10 \mu s$			1.5	mA
HALT Current with Brown Out Disabled (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$		<1	10	μA
HALT Current with Brown Out Enabled	$V_{CC} = 6V, CKI = 0 MHz$		<50	100	μA
Brown Out Trip Level (Brown Out Enabled)		1.8	3.1	4.2	V
INPUT LEVELS (V_{IH}, V_{IL})					
Reset, CKI:					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
L- and G-Port Hysteresis (Note 5)			0.05 V_{CC}	0.35 V_{CC}	V
Output Current Levels					
D Outputs:					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
	$V_{CC} = 2.5V, V_{OH} = 0.4V$	2			mA
L4-L7 Output Sink	$V_{CC} = 4.5V, V_{OL} = 2.5V$	15			mA
All Others					
Source (Weak Pull-up Mode)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
Source (Push-pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink (Push-pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs				15	mA
L4-L7 (Sink)				20	mA
All Others				3	mA

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Maximum Input Current without Latchup (Note 4)	Room Temperature			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 10 V/mS.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. HALT test conditions: L, and G0..G5 ports configured as outputs and set high. The D port set to zero. All inputs tied to V_{CC} . The comparator and the Brown Out circuits are disabled.

Note 4: Pins G6 and RESET are designed with a high voltage input network. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

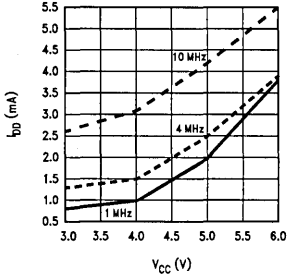
AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal/Resonator	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	1		DC	μs
	$2.5\text{V} \leq V_{CC} \leq 4.5\text{V}$	2.5		DC	μs
R/C Oscillator	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} \leq 4.5\text{V}$	7.5		DC	μs
V_{CC} Rise Time when Using Brown Out Frequency at Brown Out Reset	$V_{CC} = 0\text{V}$ to 6V	50		4	μs
CKI Frequency For Modular Output				4	MHz
CKI Clock Duty Cycle (Note 5)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 5)	$f_r = 10\text{ MHz ext. Clock}$			12	ns
Fall Time (Note 5)	$f_r = 10\text{ MHz ext. Clock}$			8	ns
Inputs					
t_{Setup}	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	200			ns
	$2.5\text{V} \leq V_{CC} \leq 4.5\text{V}$	500			ns
t_{Hold}	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	60			ns
	$2.5\text{V} \leq V_{CC} \leq 4.5\text{V}$	150			ns
Output Propagation Delay	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$				
$t_{\text{PD1}}, t_{\text{PD0}}$	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$			0.7	μs
SQ, SK	$2.5\text{V} \leq V_{CC} \leq 4.5\text{V}$			1.75	μs
All Others	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$			1	μs
	$2.5\text{V} \leq V_{CC} \leq 4.5\text{V}$			5	μs
Input Pulse Width					
Interrupt Input High Time		1			tc
Interrupt Input Low Time		1			tc
Timer Input High Time		1			tc
Timer Input Low Time		1			tc
MICROWIRE Setup Time ($t_{\mu\text{WS}}$)		20			ns
MICROWIRE Hold Time ($t_{\mu\text{WH}}$)		56			ns
MICROWIRE Output Propagation Delay ($t_{\mu\text{PD}}$)				220	ns
Reset Pulse Width		1.0			μs

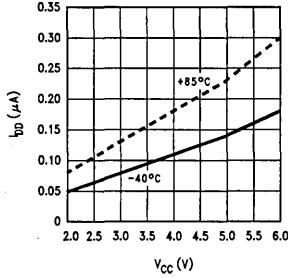
Note 5: Parameter sampled but not 100% tested.

Typical Performance Characteristics

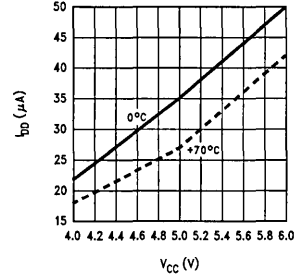
**Dynamic— I_{DD} vs V_{CC}
(Crystal Clock Option)**



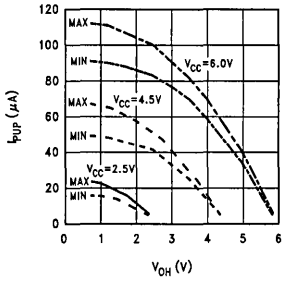
**Halt— I_{DD} vs V_{CC}
(Brown Out Disabled)**



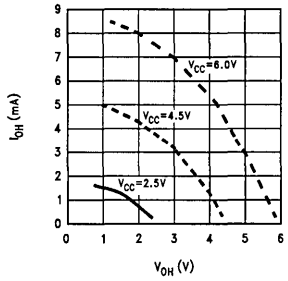
**Halt— I_{DD} vs V_{CC}
(Brown Out Enabled)**



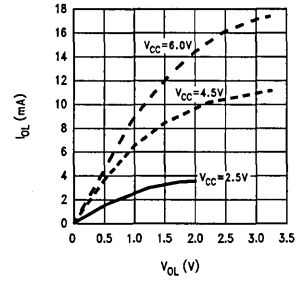
**Ports L/G Weak
Pull-Up Source Current**



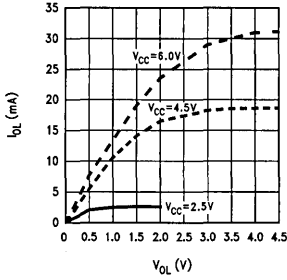
**Ports L/G Push-Pull
Source Current**



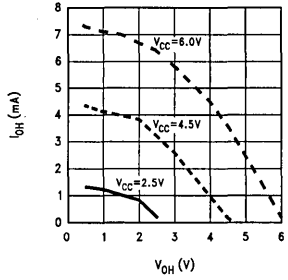
**Ports L/G Push-Pull
Sink Current**



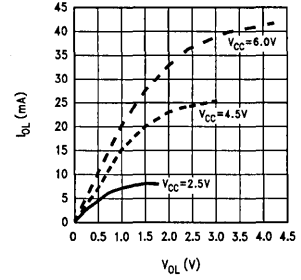
**Ports L4-L7
Sink Current**



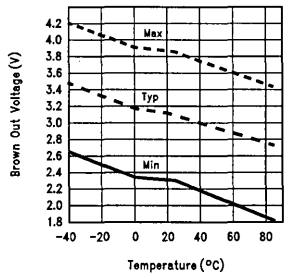
Port D Source Current



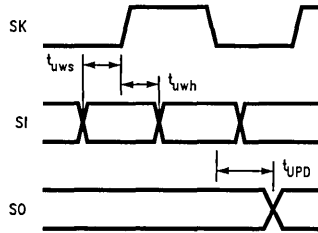
Port D Sink Current



**Brown Out Voltage
vs Temperature**



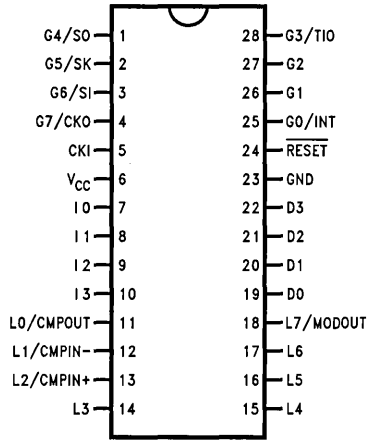
AC Electrical Characteristics (Continued)



TL/DD/11208-2

FIGURE 2. MICROWIRE/PLUS Timing

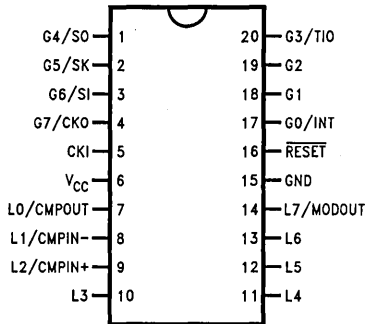
COP820CJ Pinout



TL/DD/11208-3

Top View

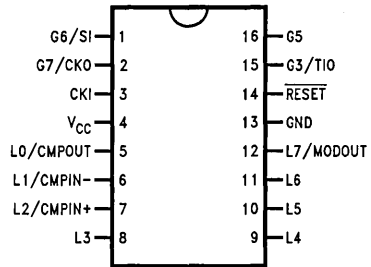
Order Number COPCJ820-XXX/N or COPCJ820-XXX/WM



TL/DD/11208-4

Top View

Order Number COPCJ822-XXX/N or COPCJ822-XXX/WM



TL/DD/11208-5

Top View

Order Number COPCJ823-XXX/WM

FIGURE 3. COP820CJ Pinout

COP820CJ Pin Assignment

Port Pin	Typ	ALT Funct.	16 Pin	20 Pin	28 Pin
L0	I/O	MIWU/CMPOUT	5	7	11
L1	I/O	MIWU/CMPIN-	6	8	12
L2	I/O	MIWU/CMPIN+	7	9	13
L3	I/O	MIWU	8	10	14
L4	I/O	MIWU	9	11	15
L5	I/O	MIWU	10	12	16
L6	I/O	MIWU	11	13	17
L7	I/O	MIWU/MODOUT	12	14	18
G0	I/O	INTR		17	25
G1	I/O			18	26
G2	I/O			19	27
G3	I/O	TIO	15	20	28
G4	I/O	SO		1	1
G5	I/O	SK	16	2	2
G6	I	SI	1	3	3
G7	I	CKO	2	4	4
I0	I				7
I1	I				8
I2	I				9
I3	I				10
D0	O				19
D1	O				20
D2	O				21
D3	O				22
V _{CC}			4	6	6
GND			13	15	23
CKI			3	5	5
RESET			14	16	24

Pin Description

V_{CC} and **GND** are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT I is a 4-bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with the L port: a data register and a configuration register. Therefore, each L

I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input with Weak Pull-up
1	0	Push-pull Zero Output
1	1	Push-pull One Output

Three data memory address locations are allocated for this port, one each for data register [00D0], configuration register [00D1] and the input pins [00D2].

Port L has the following alternate features:

L0 MIWU or CMPOUT

L1 MIWU or CMPIN-

L2 MIWU or CMPIN+

L3 MIWU

L4 MIWU (high sink current capability)

L5 MIWU (high sink current capability)

L6 MIWU (high sink current capability)

L7 MIWU or MODOUT (high sink current capability)

The selection of alternate Port L functions is done through registers WKEN [00C9] to enable MIWU and CNTRL2 [00CC] to enable comparator and modulator.

All eight L-pins have Schmitt Triggers on their inputs.

PORT G is an 8-bit port with 6 I/O pins (G0-G5) and 2 input pins (G6, G7).

All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated with the G port: a data register and a configuration register. Therefore each G port bit can be individually configured under software control as shown below:

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input with Weak Pull-up
1	0	Push-pull Zero Output
1	1	Push-pull One Output

Three data memory address locations are allocated for this port, one for data register [00D3], one for configuration register [00D5] and one for the input pins [00D6]. Since G6 and G7 are Hi-Z input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the device will be placed in the Halt mode by writing a "1" to the G7 data bit.

Six pins of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input/general purpose input (if clock option is R/C or external clock)

Pin Description (Continued)

Pins G1 and G2 currently do not have any alternate functions.

The selection of alternate Port G functions are done through registers PSW [00EF] to enable external interrupt and CNTRL1 [00EE] to select TIO and MICROWIRE operations.

PORT D is a four bit output port that is preset when RESET goes low. One data memory address location is allocated for the data register [00DC].

Note: Care must be exercised with the D2 pin operation. At RESET, the external loads on this pin must ensure that the output voltages stay above $0.8 V_{CC}$ to prevent the chip from entering special modes. Also keep the external loading on D2 to less than 1000 pF.

Functional Description

The internal architecture is shown in the block diagram. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU and CPU Registers

The ALU can do an 8-bit addition, subtraction, logical or shift operations in one cycle time. There are five CPU registers:

- A is the 8-bit Accumulator register
- PC is the 15-bit Program Counter register
 - PU is the upper 7 bits of the program counter (PC)
 - PL is the lower 8 bits of the program counter (PC)
- B is the 8-bit address register and can be auto incremented or decremented.
- X is the 8-bit alternate address register and can be auto incremented or decremented.
- SP is the 8-bit stack pointer which points to the subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns. The SP must be preset by software upon initialization.

Memory

The COP820CJ memory is separated into two memory spaces: program and data.

PROGRAM MEMORY

Program memory consists of 1024 x 8 ROM. These bytes of ROM may be instructions or constant data. The memory is addressed by the 15-bit program counter (PC). ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

The data memory address space includes on chip RAM, I/O and registers. Data memory is addressed directly by the instruction or indirectly through B, X and SP registers. The device has 64 bytes of RAM. Sixteen bytes of RAM are mapped as "registers", these can be loaded immediately, decremented and tested. Three specific registers: X, B, and SP are mapped into this space, the other registers are available for general usage.

Any bit of data memory can be directly set, reset or tested. All I/O and registers (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested, except the write once only bit (WDREN, WATCHDOG Reset Enable), and the unused and read only bits in CNTRL2 and WDREG registers.

Reset

EXTERNAL RESET

The RESET input pin when pulled low initializes the microcontroller. The user must insure that the RESET pin is held low until V_{CC} is within the specified voltage range and the clock is stabilized. An R/C circuit with a delay 5x greater than the power supply rise time is recommended (Figure 4). The device immediately goes into reset state when the RESET input goes low. When the RESET pin goes high the device comes out of reset state synchronously. The device will be running within two instruction cycles of the RESET pin going high. The following actions occur upon reset:

Port L	TRI-STATE
Port G	TRI-STATE
Port D	HIGH
PC	CLEARED
RAM Contents	RANDOM with Power-On-Reset UNAFFECTED with external Reset (power already applied)
B, X, SP	Same as RAM
PSW, CNTRL1, CNTRL2 and WDREG Reg.	CLEARED
Multi-Input Wakeup Reg. WKEDG, WKEN WKPND	CLEARED UNKNOWN
Data and Configuration Registers for L & G	CLEARED
WATCHDOG Timer	Prescaler/Counter each loaded with FF

The device comes out of the HALT mode when the RESET pin is pulled low. In this case, the user has to ensure that the RESET signal is low long enough to allow the oscillator to restart. An internal 256 t_c delay is normally used in conjunction with the two pin crystal oscillator. When the device comes out of the HALT mode through Multi-Input Wakeup, this delay allows the oscillator to stabilize.

The following additional actions occur after the device comes out of the HALT mode through the RESET pin.

If a two pin crystal/resonator oscillator is being used:

RAM Contents	UNCHANGED
Timer T1 and A Contents	UNKNOWN
WATCHDOG Timer Prescaler/Counter	ALTERED

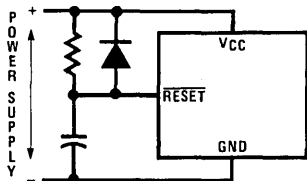
If the external or RC Clock option is being used:

RAM Contents	UNCHANGED
Timer T1 and A Contents	UNCHANGED
WATCHDOG Timer Prescaler/Counter	ALTERED

The external RESET takes priority over the Brown Out Reset.

Functional Description (Continued)

Note: If the RESET pin is pulled low while Brown Out occurs (Brown Out circuit has detected Brown Out condition), the external reset will not occur until the Brown Out condition is removed. External reset has priority only if V_{CC} is greater than the Brown Out voltage.



$RC > 5 \times \text{Power Supply Rise Time}$

TL/DD/11208-6

FIGURE 4. Recommended Reset Circuit

WATCHDOG RESET

With WATCHDOG enabled, the WATCHDOG logic resets the device if the user program does not service the WATCHDOG timer within the selected service window. The WATCHDOG reset does not disable the WATCHDOG. Upon WATCHDOG reset, the WATCHDOG Prescaler/Counter are each initialized with FF Hex.

The following actions occur upon WATCHDOG reset that are different from external reset.

WDREN WATCHDOG Reset Enable bit UNCHANGED
WDUDF WATCHDOG Underflow bit UNCHANGED

Additional initialization actions that occur as a result of WATCHDOG reset are as follows:

Port L	TRI-STATE
Port G	TRI-STATE
Port D	HIGH
PC	CLEARED
Ram Contents	UNCHANGED
B, X, SP	UNCHANGED
PSW, CNTRL1 and CNTRL2 (except WDUDF Bit) Registers	CLEARED
Multi-Input Wakeup Registers WKEDG, WKEN WKPND	CLEARED UNKNOWN
Data and Configuration Registers for L & G	CLEARED
WATCHDOG Timer	Prescaler/Counter each loaded with FF

BROWN OUT RESET

The on-board Brown Out protection circuit resets the device when the operating voltage (V_{CC}) is lower than the Brown Out voltage. The device is held in reset when V_{CC} stays below the Brown Out Voltage. The device will remain in RESET as long as V_{CC} is below the Brown Out Voltage. The Device will resume execution if V_{CC} rises above the Brown Out Voltage. If a two pin crystal/resonator clock option is selected, the Brown Out reset will trigger a 256tc delay. This delay allows the oscillator to stabilize before the device exits the reset state. The delay is not used if the clock option is either R/C or external clock. The contents of data registers and RAM are unknown following a Brown Out reset. The external reset takes priority over Brown Out Reset and will deactivate the 256 tc cycles delay if in progress. The Brown Out reset takes priority over the WATCHDOG reset.

The following actions occur as a result of Brown Out reset:

Port L	TRI-STATE
Port G	TRI-STATE
Port D	HIGH
PC	CLEARED
RAM Contents	RANDOM
B, X, SP	UNKNOWN
PSW, CNTRL1, CNTRL2 and WDREG Registers	CLEARED
Multi-Input Wakeup Registers WKEDG, WKEN WKPND	CLEARED UNKNOWN
Data and Configuration Registers for L & G	CLEARED
WATCHDOG Timer	Prescaler/Counter each loaded with FF
Timer T1 and Accumulator	Unknown data after coming out of the HALT (through Brown Out Reset) with any Clock option

Note: The development system will detect the BROWN OUT RESET externally and will force the RESET pin low. The Development System does not emulate the 256tc delay.

Brown Out Protection

An on-board protection circuit monitors the operating voltage (V_{CC}) and compares it with the minimum operating voltage specified. The Brown Out circuit is designed to reset the device if the operating voltage is below the Brown Out voltage (between 1.8V to 4.2V at -40°C to $+85^{\circ}\text{C}$). The Minimum operating voltage for the device is 2.5V with Brown Out disabled, but with BROWN OUT enabled the device is guaranteed to operate properly down to minimum Brown Out voltage (Max frequency 4 MHz). For temperature range of 0°C to 70°C the Brown Out voltage is expected to be between 1.9V to 3.9V. The circuit can be enabled or disabled by Brown Out mask option. If the device is intended to operate at lower V_{CC} (lower than Brown Out voltage VBO max), the Brown Out circuit should be disabled by the mask option.

The Brown Out circuit may be used as a power-up reset provided the power supply rise time is slower than 50 μs (0V to 6.0V).

Note: Brown Out Circuit is active in HALT mode (with the Brown Out mask option selected).

Mask Options

The COP820CJ has the following mask options associated with it:

G7 can be used either as a general purpose input or a control input to continue from the HALT mode.

The CKI and the CKO pins are automatically configured by selecting one of the three options.

The device can be driven by a clock input which can be between DC and 10 MHz.

Functional Description (Continued)

#	Option	Value	Description	Comments
1	CKI Input	1	Crystal Oscillator	(Divide by 10)
		2	External Oscillator	(Divide by 10) CKO available as G7 input
		3	R/C Oscillator	(Divide by 10) CKO available as G7 Input
2	"Brown Out"	1	Enable "Brown Out" Protection	(increased HALT current)
		2	Disable "Brown Out" Protection	
3	COP 820CJ Bonding	1	28-pin DIP	
		2	20-pin DIP/SO	
		3	16-pin SO Package	
		4	28-pin SO Package	

Oscillator Circuits

EXTERNAL OSCILLATOR

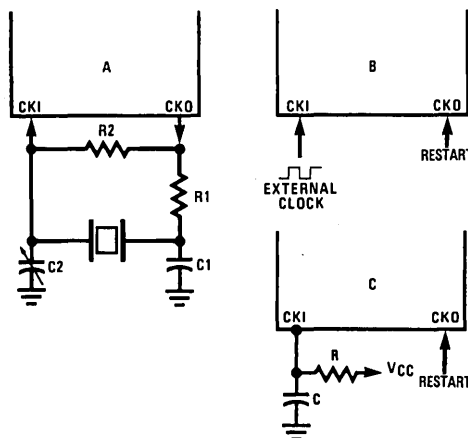
CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels. CKO is available as a general purpose input G7 and/or Halt control.

CRYSTAL OSCILLATOR

By selecting CKO as a clock output, CKI and CKO can be connected to create a crystal controlled oscillator. Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator, CKI can make a R/C oscillator. CKO is available as a general purpose input and/or HALT control. Table II shows variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/11208-7

FIGURE 5. Clock Oscillator Configurations

TABLE I. Crystal Oscillator Configuration

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq. (MHz)	Conditions
0	1	30	30-36	10	V _{CC} = 5V
0	1	30	30-36	4	V _{CC} = 5V
5.6	1	100	100-156	0.455	V _{CC} = 5V

TABLE II. RC Oscillator Configuration (Part-To-Part Variation)

R (kΩ)	C (pF)	CK1 Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	V _{CC} = 5V
5.6	100	1.1 to 1.3	7.4 to 9.0	V _{CC} = 5V
6.8	100	0.9 to 1.1	8.8 to 10.8	V _{CC} = 5V

2

Functional Description (Continued)

Current Drain

The total current drain of the chip depends on:

1. Oscillator operating mode - I1
2. Internal switching current - I2
3. Internal leakage current - I3
4. Output source current - I4
5. DC current caused by external input not at V_{CC} or GND - I5
6. DC current caused by the comparator (if comparator is enabled) - I6
7. DC current caused by the Brown Out - I7

Thus the total current drain is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$$

To reduce the total current drain, each of the above components must be minimum. Operating with a crystal network will draw more current than an external square-wave. The R/C-mode will draw the most. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

The following formula may be used to compute total current drain when operating the controller in different modes.

$$I_2 = C \times V \times f$$

where: C = equivalent capacitance of the chip

V = operating voltage

f = CKI frequency

Halt Mode

The COP820CJ is a fully static device. The device enters the HALT mode by writing a one to the G7 bit of the G data register. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. In this mode the chip will only draw leakage current (output current and DC current due to the Brown Out circuit if Brown Out is enabled).

The device supports four different methods of exiting the HALT mode. The first method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO is a dedicated output). It may be used either with an RC clock configuration or an external clock configuration. The second method of exiting the HALT mode is with the multi-Input Wakeup feature on the L port. The third method of exiting the HALT mode is by pulling the RESET input low. The fourth method is with the operating voltage going below Brown Out voltage (if Brown Out is enabled by mask option).

If the two pin crystal/resonator oscillator is being used and Multi-Input Wakeup or Brown Out causes the device to exit the HALT mode, the WAKEUP signal does not allow the chip to start running immediately since crystal oscillators have a delayed start up time to reach full amplitude and frequency stability. The WATCHDOG timer (consisting of an 8-bit prescaler followed by an 8-bit counter) is used to generate a fixed delay of 256tc to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid WAKEUP signal only the oscillator circuitry is enabled. The WATCHDOG Counter and Prescaler are each loaded with a value of FF Hex. The WATCHDOG prescaler is clocked with the tc instruction cycle. (The tc clock is derived by dividing the oscillator clock down by a factor of 10). The Schmitt trigger following the CKI inverter on the chip ensures that the WATCHDOG timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specs. This Schmitt trigger is not part of the oscillator closed loop. The start-up timeout from the WATCHDOG timer enables the clock signals to be routed to the rest of the chip. The delay is not activated when the device comes out of HALT mode through RESET pin. Also, if the clock option is either RC or External clock, the delay is not used, but the WATCHDOG Prescaler/-Counter contents are changed. The Development System will not emulate the 256tc delay.

The RESET pin or Brown Out will cause the device to reset and start executing from address X'0000. A low to high transition on the G7 pin (if single pin oscillator is used) or Multi-Input Wakeup will cause the device to start executing from the address following the HALT instruction.

When RESET pin is used to exit the device from the HALT mode and the two pin crystal/resonator (CKI/CKO) clock option is selected, the contents of the Accumulator and the Timer T1 are undetermined following the reset. All other information except the WATCHDOG Prescaler/Counter contents is retained until continuing. If the device comes out of the HALT mode through Brown Out reset, the contents of data registers and RAM are unknown following the reset. All information except the WATCHDOG Prescaler/Counter contents is retained if the device exits the HALT mode through G7 pin or Multi-Input Wakeup.

G7 is the HALT-restart pin, but it can still be used as an input. If the device is not halted, G7 can be used as a general purpose input.

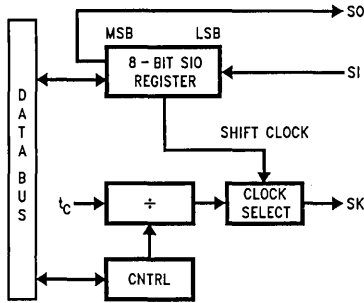
If the Brown Out Enable mask option is selected, the Brown Out circuit remains active during the HALT mode causing additional current to be drawn.

Note: To allow clock resynchronization, it is necessary to program two NOP's immediately after the device comes out of the HALT mode. The user must program two NOP's following the "enter HALT mode" (set G7 data bit) instruction.

Functional Description (Continued)

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the device to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMs, etc.) and with other microcontrollers which support the MICROWIRE/PLUS interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 6* shows the block diagram of the MICROWIRE/PLUS interface.



TL/DD/11208-8

FIGURE 6. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS interface with the internal clock source is called the Master mode of operation. Operating the MICROWIRE/PLUS interface with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	2t _c
0	1	4t _c
1	x	8t _c

where,
t_c is the instruction cycle time.

MICROWIRE/PLUS OPERATION

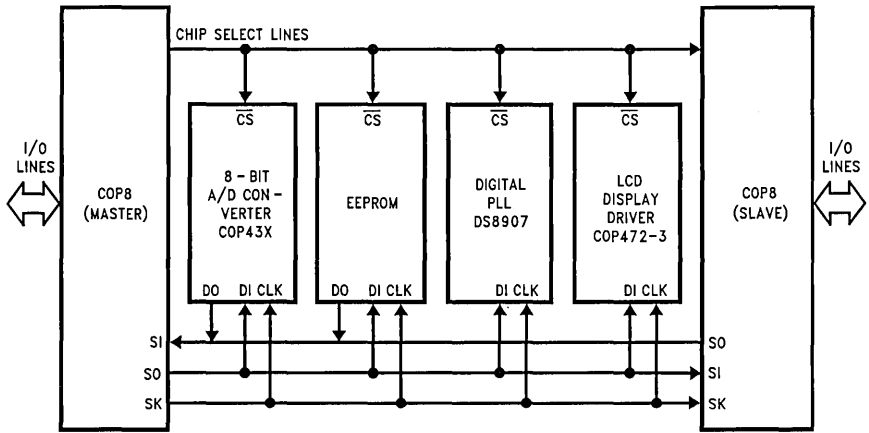
Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The device may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 7* shows how two device microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the device. The MICROWIRE/PLUS Master always initiates all data exchanges (*Figure 7*). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions on the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summarizes the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions on the G Port. The SK pin must be selected as an input and the SO pin selected as an output pin by appropriately setting up the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.



TL/DD/11208-23

FIGURE 7. MICROWIRE/PLUS Application

Functional Description (Continued)

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The device has a powerful 16-bit timer with an associated 16-bit register enabling it to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table V details various timer operating modes and their requisite control settings.

MODE 1. TIMER WITH AUTO-LOAD REGISTER

In this mode of operation, the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the regis-

ter R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allows the generation of square-wave outputs or pulse width modulated outputs under software control (Figure 8).

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt (Figure 9).

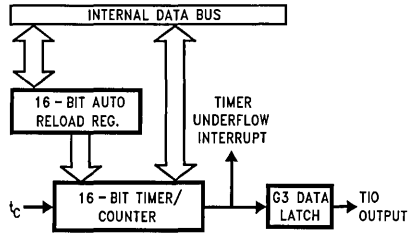


FIGURE 8. Timer/Counter Auto Reload Mode Block Diagram

TL/DD/11208-24

TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter w/Auto-Load Reg.	Timer Underflow	TIO Pos. Edge
0 0 1	External Counter w/Auto-Load Reg.	Timer Underflow	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer w/Auto-Load Reg.	Timer Underflow	t_c
1 0 1	Timer w/Auto-Load Reg./Toggle TIO Out	Timer Underflow	t_c
1 1 0	Timer w/Capture Register	TIO Pos. Edge	t_c
1 1 1	Timer w/Capture Register	TIO Neg. Edge	t_c

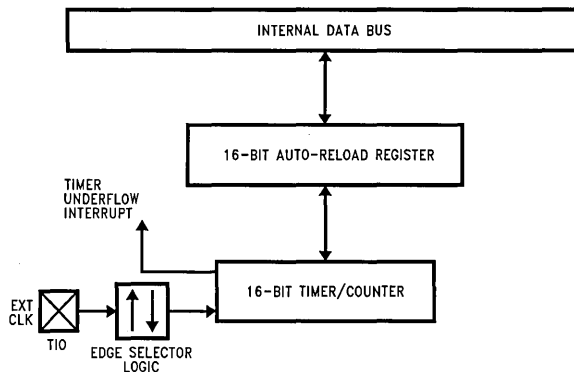


FIGURE 9. Timer in External Event Counter Mode

TL/DD/11208-29

Functional Description (Continued)

MODE 3: TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge (Figure 10).

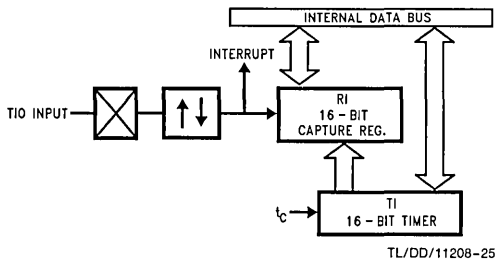


FIGURE 10. Timer Capture Mode Block Diagram

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.

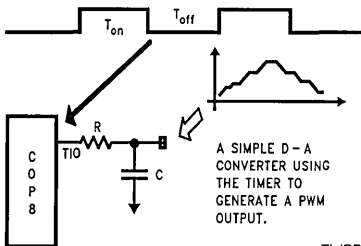


FIGURE 11. Timer Application

WATCHDOG

The COP820CJ has an on-board 8-bit WATCHDOG timer. The timer contains an 8-bit READ/WRITE down counter clocked by an 8-bit prescaler. Under software control the timer can be dedicated for the WATCHDOG or used as a general purpose counter. Figure 12 shows the WATCHDOG timer block diagram.

MODE 1: WATCHDOG TIMER

The WATCHDOG is designed to detect user programs getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The WATCHDOG can be enabled or disabled (only once) after the device is reset as a result of brown out reset or external reset. On power-up the WATCHDOG is disabled. The WATCHDOG is enabled by writing a "1" to WDREN bit (resides in WDREG register). Once enabled, the user program should write periodically into the 8-bit counter before the counter underflows. The 8-bit counter (WDCNT) is memory mapped at address 0CE Hex. The counter is loaded with n-1 to get n counts. The counter underflow resets the device, but does not disable the WATCHDOG. Loading the 8-bit counter initializes the prescaler with FF Hex and starts the prescaler/counter. Prescaler and counter are stopped upon counter underflow. Prescaler and counter are each loaded with FF Hex when the device goes into the HALT mode. The prescaler is used for crystal/resonator start-up when the device exits the HALT mode through Multi-Input Wakeup. In this case, the prescaler/counter contents are changed.

MODE 2: TIMER

In this mode, the prescaler/counter is used as a timer by keeping the WDREN (WATCHDOG reset enable) bit at 0. The counter underflow sets the WUDF (underflow) bit and the underflow does not reset the device. Loading the 8-bit counter (load n-1 for n counts) sets the WDTE bit (WATCHDOG Timer Enable) to "1", loads the prescaler with FF, and starts the timer. The counter underflow stops the timer. The WDTE bit serves as a start bit for the WATCHDOG timer. This bit is set when the 8-bit counter is loaded by the user program. The load could be as a result of WATCHDOG service (WATCHDOG timer dedicated for WATCHDOG function) or write to the counter (WATCHDOG timer used as a general purpose counter). The bit is cleared upon Brown Out reset, WATCHDOG reset or external reset. The bit is not memory mapped and is transparent to the user program.

TABLE VI. WATCHDOG Control/Status

Parameter	HALT Mode	WD Reset	EXT/BOR Reset (Note 1)	Counter Load
8-Bit Prescaler	FF	FF	FF	FF
8-Bit WD Counter	FF	FF	FF	User Value
WDREN Bit	Unchanged	Unchanged	0	No Effect
WUDF Bit	0	Unchanged	0	0
WDTE Signal	Unchanged	0	0	1

Note 1: BOR is Brown Out Reset.

Functional Description (Continued)

CONTROL/STATUS BITS

WDUDF: WATCHDOG Timer Underflow Bit

This bit resides in the CNTRL2 Register. The bit is set when the WATCHDOG timer underflows. The underflow resets the device if the WATCHDOG reset enable bit is set (WDREN = 1). Otherwise, WDUDF can be used as the timer underflow flag. The bit is cleared upon Brown-Out reset, external reset, load to the 8-bit counter, or going into the HALT mode. It is a read only bit.

WDREN: WD Reset Enable

WDREN bit resides in a separate register (bit 0 of WDREG). This bit enables the WATCHDOG timer to generate a reset. The bit is cleared upon Brown Out reset, or external reset. The bit under software control can be written to only once (once written to, the hardware does not allow the bit to be changed during program execution).

WDREN = 1 WATCHDOG reset is enabled.

WDREN = 0 WATCHDOG reset is disabled.

Table VI shows the impact of Brown Out Reset, WATCHDOG Reset, and External Reset on the Control/Status bits.

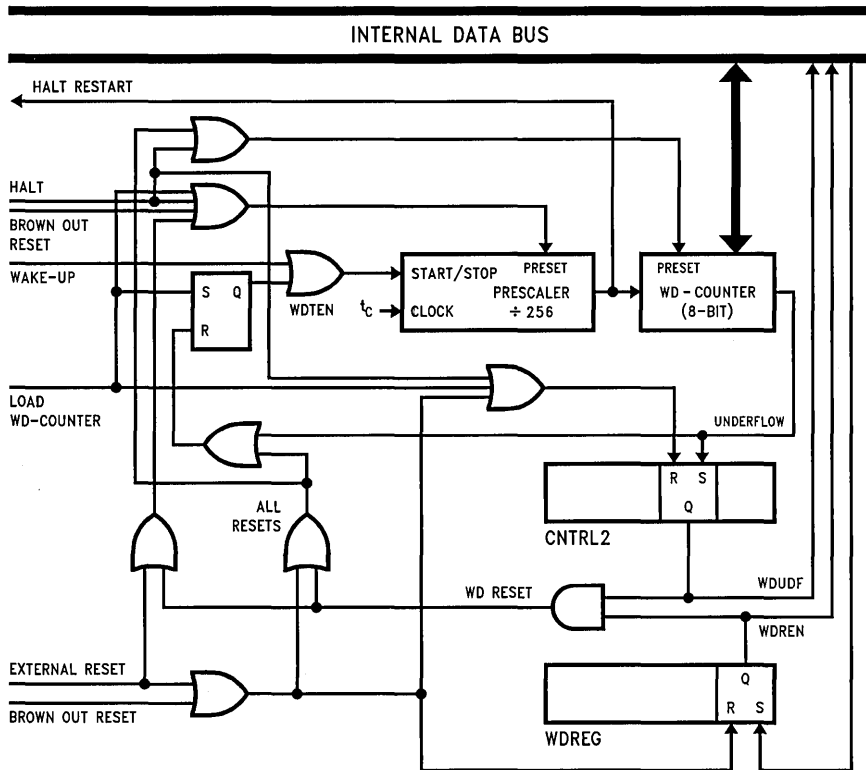


FIGURE 12. WATCHDOG Timer Block Diagram

TL/DD/11208-15

Modulator/Timer

The Modulator/Timer contains an 8-bit counter and an 8-bit autoreload register (MODRL address 0CF Hex). The Modulator/Timer has two modes of operation, selected by the control bit MC3. The Modulator/Timer Control bits MC1, MC2 and MC3 reside in CNTRL2 Register.

MODE 1: MODULATOR

The Modulator is used to generate high frequency pulses on the modulator output pin (L7). The L7 pin should be configured as an output. The number of pulses is determined by the 8-bit down counter. Under software control the modulator input clock can be either CKI or tC. The tC clock is derived by dividing down the oscillator clock by a factor of 10. Three control bits (MC1, MC2, and MC3) are used for the Modulator/Timer output control. When MC2 = 1 and MC3 = 1, CKI is used as the modulator input clock. When MC2 = 0, and MC3 = 1, tC is used as the modulator input clock. The user loads the counter with the desired number of counts (256 max) and sets MC1 to start the counter. The modulator autoreload register is loaded with n-1 to get n pulses. CKI or tC pulses are routed to the modulator output (L7) until the counter underflows (Figure 13). Upon underflow the hardware resets MC1 and stops the counter. The L7 pin goes low and stays low until the counter is restarted by the user program. The user program has the responsibility to timeout the low time. Unless the number of counts is changed, the user program does not have to load the counter each time the counter is started. The counter can simply be started by setting the MC1 bit. Setting MC1 by software will load the counter with the value of the autoreload register. The software can reset MC1 to stop the counter.

MODE 2: PWM TIMER

The counter can also be used as a PWM Timer. In this mode, an 8-bit register is used to serve as an autoreload register (MODRL).

a. 50% Duty Cycle:

When MC1 is 1 and MC2, MC3 are 0, a 50% duty cycle free running signal is generated on the L7 output pin (Figure 14). The L7 pin must be configured as an output pin. In this mode the 8-bit counter is clocked by tC. Setting the MC1

control bit by software loads the counter with the value of the autoreload register and starts the counter. The counter underflow toggles the (L7) output pin. The 50% duty cycle signal will be continuously generated until MC1 is reset by the user program.

b. Variable Duty Cycle:

When MC3 = 0 and MC2 = 1, a variable duty cycle PWM signal is generated on the L7 output pin. The counter is clocked by tC. In this mode the 16-bit timer T1 along with the 8-bit down counter are used to generate a variable duty cycle PWM signal. The timer T1 underflow sets MC1 which starts the down counter and it also sets L7 high (L7 should be configured as an output). When the counter underflows the MC1 control bit is reset and the L7 output will go low until the next timer T1 underflow. Therefore, the width of the output pulse is controlled by the 8-bit counter and the pulse duration is controlled by the 16-bit timer T1 (Figure 15). Timer T1 must be configured in "PWM Mode/Toggle TIO Out" (CNTRL1 Bits 7,6,5 = 101).

Table VII shows the different operation modes for the Modulator/Timer.

TABLE VII. Modulator/Timer Modes

Control Bits in CNTRL2(00CC)			Operation Mode L7 Function
MC3	MC2	MC1	
0	0	0	Normal I/O
0	0	1	50% Duty Cycle Mode (Clocked by tc)
0	1	X	Variable Duty Cycle Mode (Clocked by tc) Using Timer 1 Underflow
1	0	X	Modulator Mode (Clocked by tc)
1	1	X	Modulator Mode (Clocked by CKI)

Note: MC1, MC2 and MC3 control bits are cleared upon reset.

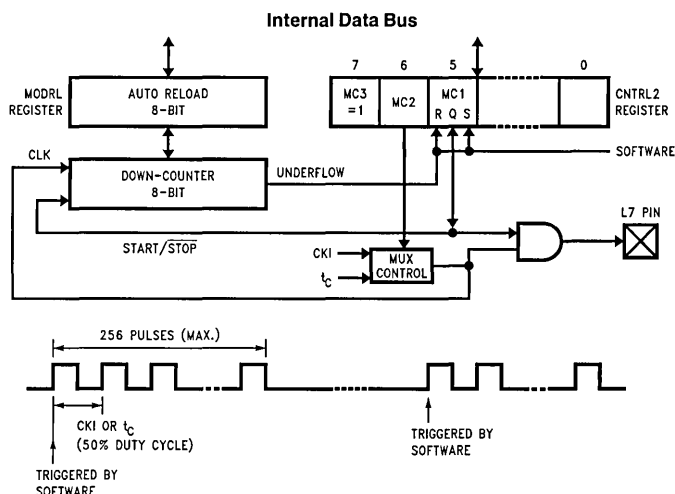
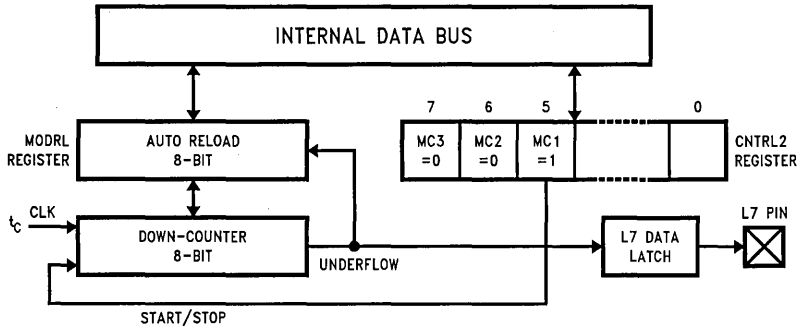


FIGURE 13. Mode 1: Modulator Block Diagram/Output Waveform

TL/DD/11208-16

Modulator/Timer (Continued)

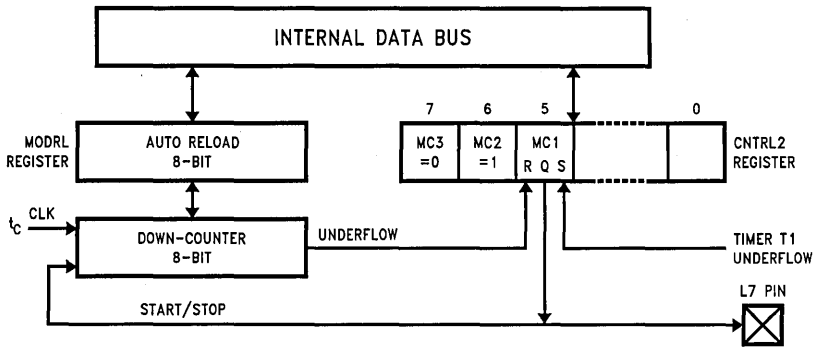


TL/DD/11208-17

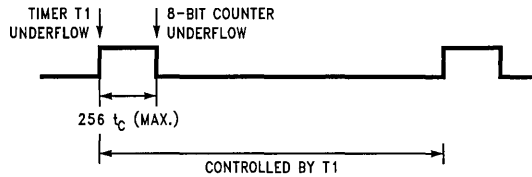


TL/DD/11208-18

FIGURE 14. Mode 2a: 50% Duty Cycle Output



TL/DD/11208-19



TL/DD/11208-20

FIGURE 15. Mode 2b: Variable Duty Cycle Output

Comparator

The COP820CJ has one differential comparator. Ports L0–L2 are used for the comparator. The output of the comparator is brought out to a pin. Port L has the following assignments:

- L0 Comparator output
- L1 Comparator negative input
- L2 Comparator positive input

THE COMPARATOR STATUS/CONTROL BITS

These bits reside in the CNTRL2 Register (Address 0CC)

- CMPEN Enables comparator ("1" = enable)
- CMPRD Reads comparator output internally (CMPEN = 1, CMPOE = X)
- CMPOE Enables comparator output to pin L0 ("1" = enable), CMPEN bit must be set to enable this function. If CMPEN = 0, L0 will be 0.

The Comparator Select/Control bits are cleared on RESET (the comparator is disabled). To save power the program should also disable the comparator before the device enters the HALT mode.

The user program must set up L0, L1 and L2 ports correctly for comparator Inputs/Output: L1 and L2 need to be configured as inputs and L0 as output. Table VIII shows the DC and AC characteristics for the comparator.

Multi-Input Wake Up

The Multi-Input Wakeup feature is used to return (wakeup) the device from the HALT mode. *Figure 16* shows the Multi-Input Wakeup logic.

This feature utilizes the L Port. The user selects which particular L port bit or combination of L Port bits will cause the device to exit the HALT mode. Three 8-bit memory mapped registers, Reg:WKEN, Reg:WKEDG, and Reg:WKPND are used in conjunction with the L port to implement the Multi-Input Wakeup feature.

All three registers Reg:WKEN, Reg:WKPND, and Reg:WKEDG are read/write registers, and are cleared at reset, except WKPND. WKPND is unknown on reset.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg:WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by

the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L port bit 5, where bit 5 has previously been enabled for an input. The program would be as follows:

```

RBIT 5,WKEN
SBIT 5,WKEDG
RBIT 5,WKPND
SBIT 5,WKEN
  
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared. This same procedure should be used following RESET, since the L port inputs are left floating as a result of RESET.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called Reg:WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since the Reg:WKPND is a pending register for the occurrence of selected wakeup conditions, the device will not enter the HALT mode if any Wakeup bit is both enabled and pending. Setting the G7 data bit under this condition will not allow the device to enter the HALT mode. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

If a crystal oscillator is being used, the Wakeup signal will not start the chip running immediately since crystal oscillators have a finite start up time. The WATCHDOG timer prescaler generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the device to execute instructions. In this case, upon detecting a valid Wakeup signal only the oscillator circuitry and the WATCHDOG timer are enabled. The WATCHDOG timer prescaler is loaded with a value of FF Hex (256 counts) and is clocked from the tc instruction cycle clock. The tc clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on chip inverter ensures that the WATCHDOG timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specs. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the WATCHDOG timer enables the clock signals to be routed to the rest of the chip.

TABLE VIII. DC and AC Characteristics $4V \leq V_{CC} \leq 6V$, $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ (Note 1)

Parameters	Conditions	Min	Type	Max	Units
Input Offset Voltage	$0.4V < V_{IN} < V_{CC} - 1.5V$		± 10	± 25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Voltage Gain			300k		V/V
DC Supply Current (when enabled)	$V_{CC} = 6.0V$			250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load			1	μs

Note 1: For comparator output current characteristics see L-Port specs.

Multi-Input Wakeup (Continued)

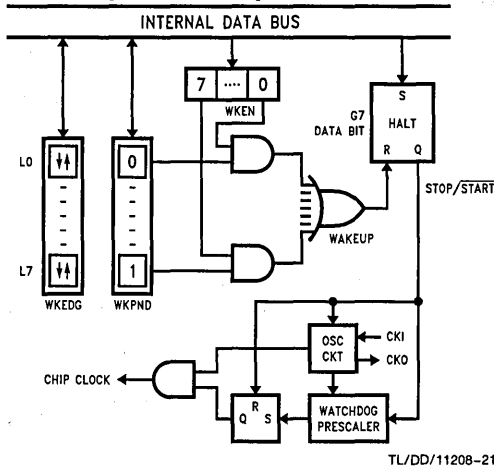


FIGURE 16. Multi-Input Wakeup Logic

INTERRUPTS

The device has a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer carry or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupts respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After an interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

DETECTION OF ILLEGAL CONDITIONS

The device incorporates a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise, and "brown out" voltage drop situations. Specifically, it detects cases of executing out of undefined ROM area and unbalanced tack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also "00". Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the device grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

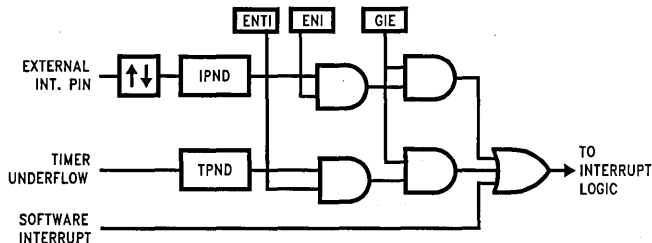


FIGURE 17. Interrupt Block Diagram

Control Registers

CNTRL1 REGISTER (ADDRESS 00EE)

The Timer and MICROWIRE control register contains the following bits:

- SL1 and SL0 Select the MICROWIRE clock divide-by (00 = 2, 01 = 4, 1x = 8)
- IEDG External interrupt edge polarity select
- MSEL Selects G5 and G4 as MICROWIRE signals SK and SO respectively
- TRUN Used to start and stop the timer/counter (1 = run, 0 = stop)
- TC1 Timer T1 Mode Control Bit
- TC2 Timer T1 Mode Control Bit
- TC3 Timer T1 Mode Control Bit

Bit 7							Bit 0	
TC1	TC2	TC3	TRUN	MSEL	IEDG	SL1	SL0	

PSW REGISTER (ADDRESS 00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable (enables interrupts)
- ENI External interrupt enable
- BUSY MICROWIRE busy shifting flag
- PND External interrupt pending
- ENTI Timer T1 interrupt enable
- TPND Timer T1 interrupt pending (timer Underflow or capture edge)
- C Carry Flip/Flop
- HC Half-Carry Flip/Flop

Bit 7							Bit 0	
HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE	

The Half-Carry bit is also effected by all the instructions that effect the Carry flag. The flag values depend upon the instruction. For example, after executing the ADC instruction the values of the Carry and the Half-Carry flag depend upon the operands involved. However, instructions like SET C and RESET C will set and clear both the carry flags. Table IX lists the instructions that effect the HC and the C flags.

TABLE IX. Instructions Effecting HC and C Flags

Instr.	HC Flag	C Flag
ADC	Depends on Operands	Depends on Operands
SUBC	Depends on Operands	Depends on Operands
SET C	Set	Set
RESET C	Set	Set
RRC	Depends on Operands	Depends on Operands

CNTRL2 REGISTER (ADDRESS 00CC)

Bit 7							Bit 0	
MC3	MC2	MC1	CMPEN	CMPRD	CMPOE	WDUDF	unused	
R/W	R/W	R/W	R/W	R/O	R/W	R/O		

- MC3 Modulator/Timer Control Bit
- MC2 Modulator/Timer Control Bit
- MC1 Modulator/Timer Control Bit
- CMPEN Comparator Enable Bit
- CMPRD Comparator Read Bit
- CMPOE Comparator Output Enable Bit
- WDUDF WATCHDOG Timer Underflow Bit (Read Only)

WDREG REGISTER (ADDRESS 00CD)

WDREN WATCHDOG Reset Enable Bit (Write Once Only)

Bit 7		Bit 0	
UNUSED		WDREN	

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

TABLE X. Memory Map

Address	Contents
00 to 2F	On-chip RAM bytes (48 bytes)
30 to 7F	Unused RAM Address Space (Reads as All Ones)
80 to BF	Expansion Space for On-Chip EERAM (Reads Undefined Data)
C0 to C7	Reserved
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	Reserved
CC	Control2 Register (CNTRL2)
CD	WATCHDOG Register (WDREG)
CE	WATCHDOG Counter (WDCNT)
CF	Modulator Reload (MODRL)
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8 to DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to EF	On-Chip Functions and Registers
E0 to E7	Reserved for Future Parts
E8	Reserved
E9	MICROWIRE Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer1 Autoreload Register Lower Byte
ED	Timer1 Autoreload Register Upper Byte
EE	CNTRL1 Control Register
EF	PSW Register
F0 to FF	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Reading other unused memory locations will return undefined data.

Addressing Modes

The COP820CJ has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

REGISTER INDIRECT

This is the "normal" addressing mode for the chip. The operand is the data memory addressed by the **B** or **X** pointer. REGISTER INDIRECT WITH AUTO POST INCREMENT OR DECREMENT

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the **B** or **X** pointer. This is a register indirect mode that automatically post increments or post decrements the **B** or **X** pointer after executing the instruction.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

SHORT IMMEDIATE

This addressing mode issued with the LD B,# instruction, where the immediate # is less than 16. The instruction contains a 4-bit immediate field as the operand.

INDIRECT

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

RELATIVE

This mode is used for the JP instruction with the instruction field being added to the program counter to produce the next instruction address. JP has a range from -31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "blocks" or "pages" when using JP since all 15 bits of the PC are used.

ABSOLUTE

This mode is used with the JMP and JSR instructions with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

ABSOLUTE LONG

This mode is used with the JMPL and JSRL instructions with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the entire 32k program memory space.

INDIRECT

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serves as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
Meml	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set

ADD ADC	add add with carry	A ← A + Meml A ← A + Meml + C, C ← Carry HC ← Half Carry
SUBC	subtract with carry	A ← A + Meml + C, C ← Carry HC ← Half Carry
AND OR XOR IFEQ IFGT IFBNE DRSZ SBIT	Logical AND Logical OR Logical Exclusive-OR IF equal IF greater than IF B not equal Decrement Reg., skip if zero Set bit	A ← A and Meml A ← A or Meml A ← A xor Meml Compare A and Meml, Do next if A = Meml Compare A and Meml, Do next if A > Meml Do next if lower 4 bits of B ≠ Imm Reg ← Reg - 1, skip if Reg goes to 0 1 to bit, Mem (bit = 0 to 7 immediate) 0 to bit, Mem If bit, Mem is true, do next instr.
RBIT IFBIT	Reset bit If bit	
X LD A LD mem LD Reg	Exchange A with memory Load A with memory Load Direct memory Immed. Load Register memory Immed.	A ↔ Mem A ← Meml Mem ← Imm Reg ← Imm
X X LD A LD A LD M	Exchange A with memory [B] Exchange A with memory [X] Load A with memory [B] Load A with memory [X] Load Memory Immediate	A ↔ [B] (B ← B ± 1) A ↔ [X] (X ← X ± 1) A ← [B] (B ← B ± 1) A ← [X] (X ← X ± 1) [B] ← Imm (B ← B ± 1)
CLRA INCA DECA LAID DCORA RRCA SWAPA SC RC IFC IFNC	Clear A Increment A Decrement A Load A indirect from ROM DECIMAL CORRECT A ROTATE A RIGHT THRU C Swap nibbles of A Set C Reset C If C If not C	A ← 0 A ← A + 1 A ← A - 1 A ← ROM(PU,A) A ← BCD correction (follows ADC, SUBC) C → A7 → ... → A0 → C A7...A4 ↔ A3...A0 C ← 1, HC ← 1 C ← 0, HC ← 0 If C is true, do next instruction If C is not true, do next instruction
JMPL JMP JP JSRL JSR JID RET RETSK RETI INTR NOP	Jump absolute long Jump absolute Jump relative short Jump subroutine long Jump subroutine Jump indirect Return from subroutine Return and Skip Return from Interrupt Generate an interrupt No operation	PC ← ii (ii = 15 bits, 0 to 32k) PC11..0 ← i (i = 12 bits) PC ← PC + r (r is -31 to +32, not 1) [SP] ← PL, [SP-1] ← PU, SP-2, PC ← ii [SP] ← PL, [SP-1] ← PU, SP-2, PC11..0 ← i PL ← ROM(PU,A) SP+2, PL ← [SP], PU ← [SP-1] SP+2, PL ← [SP], PU ← [SP-1], Skip next instruction SP+2, PL ← [SP], PU ← [SP-1], GIE ← 1 [SP] ← PL, [SP-1] ← PU, SP-2, PC ← 0FF PC ← PC + 1

Bits 7-4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	OPCODE LIST
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0, [B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A, [B]	IFBIT 1, [B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A, [B]	IFBIT 2, [B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X-]	X A, [B-]	IFGT A, #i	IFGT A, [B]	IFBIT 3, [B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A, [B]	IFBIT 4, [B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	AND A, #i	AND A, [B]	IFBIT 5, [B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A, [B]	IFBIT 6, [B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	OR A, #i	OR A, [B]	IFBIT 7, [B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0, [B]	RBIT 0, [B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1, [B]	RBIT 1, [B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2, [B]	RBIT 2, [B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X-]	LD A, [B-]	LD [B-], #i	DECA	SBIT 3, [B]	RBIT 3, [B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	*	SBIT 4, [B]	RBIT 4, [B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP + 13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5, [B]	RBIT 5, [B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP + 14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00-0EFF	JMP 0E00-0EFF	JP + 31	JP + 15	E
JP -0	JP -16	LD 0FF, #i	DRSZ 0FF	*	*	*	RETI	SBIT 7, [B]	RBIT 7, [B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

OPCODE LIST

BITS 3-0

2-4/8

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

Arithmetic Instructions (Bytes/Cycles)

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions (Bytes/Cycles)

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr	
	[B]	[X]			[B+, B-]	[X+, X-]
X A,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B,Imm				1/1		
LD B,Imm				2/3		
LD Mem,Imm			3/3		2/2	
LD Reg,Imm				2/3		

(If B < 16)
(If B > 15)

* => Memory location addressed by B or X or directly.

Instructions Using A & C

Instructions	Bytes/Cycles
CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

Instructions	Bytes/Cycles
JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

BYTES and CYCLES per INSTRUCTION (Continued)

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MH-820CJ20D5PC	20 DIP	4.5V-5.5V	COP822CJ
MHW-820CJ20DWPC	20 DIP	2.3V-6.0V	COP822CJ
MHW-820CJ28D5PC	28 DIP	4.5V-5.5V	COP820CJ
MHW-820CJ28DWPC	28 DIP	2.3V-6.0V	COP820CJ

Development Support (Continued)

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM® PC-XT®, PC-AT® or compatible	424410527-001

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

SINGLE CHIP EMULATOR

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP820CJMHD-X	X = 1: crystal X = 2: external X = 3: R/C	28 DIP	Multi-Chip Module (MCM), UV erasable	COP820CJ
COP820CJMHEA-X	X = 1: crystal X = 2: external X = 3: R/C	28 LCC	MCM (same footprint as 28 SO), UV erasable	COP820CJ
COP822CJMHD-X	X = 1: crystal X = 2: external X = 3: R/C	20 DIP	MCM, UV erasable	COP822CJ

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator board for 28 DIP and for use with Scrambler Boards	COP820CJMHD
COP8-SCRM-DIP	Scrambler board for 20 DIP socket	COP822CJMHD
COP8-SCRM-SBX	Scrambler board for 28 LCC sockets	COP820CJMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler board	COP822CJMHD COP820CJMHD
COP8-PRGM-SBX	Duplicator Board with COP8-SCRM-SBX Scrambler Board	COP820CJMHEA COP820CJHHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user has questions, he can leave messages on our electronic bulletin board.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 baud

Setup: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs. 7 Days

COP8620C/COP8622C/COP8640C/COP8642C/ COP86L20C/COP86L22C/COP86L40C/COP86L42C Single-Chip microCMOS Microcontrollers

General Description

The COP8620C/COP8640C are members of the COP86™ microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. These low cost microcontrollers are complete microcomputers containing all system timing, interrupt logic, ROM, RAM, EEPROM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSTM™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP8620C/COP8640C to the specific application. The part operates over a voltage range of 4.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate.

Features

- Low Cost 8-bit microcontroller
- Fully static CMOS
- 1 μ s instruction time
- Low current drain (2.2 mA at 3 μ s instruction rate)
 Low current static HALT mode (Typically < 1 μ A)
- Single supply operation: 4.5 to 6.0V
- 2048 Bytes ROM/64 Bytes RAM/64 Bytes EEPROM on COP8640C

- 1024 bytes ROM/64 bytes RAM/64 bytes EEPROM on COP8620C
- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - Reset master clear
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE PLUSTM™ serial I/O
- 28 pin package (optional 20 pin package)
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature range: -40°C to +85°C, -55°C to +125°C
- Compatible with COP8640CMH Series of Emulators
- Fully supported by National's Development Systems

Block Diagram

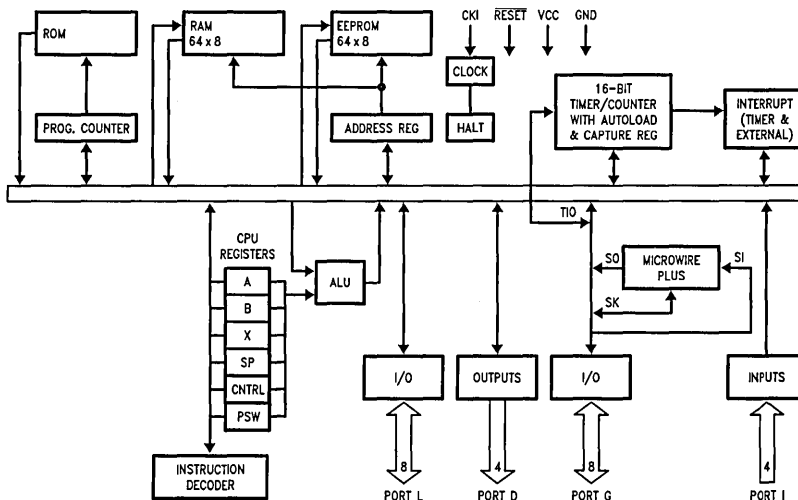


FIGURE 1

TL/DD/10366-1

COP86L20C/COP86L22C/COP86L40C/COP86L42C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 7V
 Voltage at any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 50 mA

Total Current out of GND Pin (Sink) 60 mA
 Storage Temperature Range $-65^{\circ}C$ to $+140^{\circ}C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		2.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			$0.1 V_{CC}$	V
Operating Voltage during EEPROM Write		4.5		6.0	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			9	mA
Supply Current during Write Operation (Note 2) CKI = 10 MHz	$V_{CC} = 6.0V, t_c = 1 \mu s$			15	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$		<1	10	μA
Input Levels					
RESET, CKI					
Logic High		$0.9 V_{CC}$		$0.1 V_{CC}$	V
Logic Low					V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V, V_{IN} = 0V$	40		250	μA
G Port Input Hysteresis (Note 5)				$0.35 V_{CC}$	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	2			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 4) Without Latchup (Room Temp) (Note 5)	Room Temp			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance (Note 5)				7	pF
EEPROM Characteristics					
EEPROM Write Cycle Time				10	ms
EEPROM Number of Write Cycles				10,000	Cycle
EEPROM Data Retention		10			Years

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports are at TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at G6 and RESET pins must be limited to less than 14V.

COP86L20C/COP86L22C/COP86L40C/COP86L42C (Continued)**AC Electrical Characteristics** $-40^{\circ}\text{C} < T_A < +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext, Crystal/Resonator (Div-by 10) R/C Oscillator Mode (Div-by 10)	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
	$2.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	2.5		DC	μs
	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	7.5		DC	μs
CKI Clock Duty Cycle (Note 5)		40		60	%
Rise Time (Note 5)	$f_r = 10\text{ MHz Ext Clock}$			12	ns
Fall Time (Note 5)	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}		200			ns
t_{HOLD}		60			ns
Output Propagation Delay	$C_L = 100\text{ pF}, R_L = 2.2\text{ k}\Omega$				
$t_{\text{PD1}}, t_{\text{PD0}}$				0.7	μs
SO, SK				1	μs
All Others					
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).

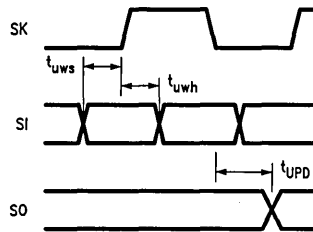
Timing Diagram

FIGURE 2. MICROWIRE/PLUS Timing

TL/DD/10366-19

COP8620C/COP8622C/COP8640C/COP8642C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 7V
 Voltage at any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 50 mA

Total Current out of GND Pin (Sink) 60 mA
 Storage Temperature Range $-65^{\circ}C$ to $+140^{\circ}C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			$0.1 V_{CC}$	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			9	mA
Supply Current during Write Operation (Note 2) CKI = 10 MHz	$V_{CC} = 6.0V, t_c = 1 \mu s$			15	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0$ MHz		<1	10	μA
Input Levels RESET, CKI					
Logic High		$0.9 V_{CC}$		$0.1 V_{CC}$	V
Logic Low				$0.1 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V, V_{IN} = 0V$	40		250	μA
G Port Input Hysteresis (Note 5)				$0.35 V_{CC}$	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 4) Without Latchup (Room Temp) (Note 5)	Room Temp			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance (Note 5)				7	pF
EEPROM Characteristics					
EEPROM Write Cycle Time				10	ms
EEPROM Number of Write Cycles				10,000	Cycle
EEPROM Data Retention		10			Years

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports are at TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at G6 and RESET pins must be limited to less than 14V.

COP8620C/COP8622C/COP8640C/COP8642C (Continued)**AC Electrical Characteristics** $-40^{\circ}\text{C} < T_A < +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext, Crystal/Resonator (Div-by 10)		1		DC	μs
R/C Oscillator Mode (Div-by 10)		3		DC	μs
CKI Clock Duty Cycle (Note 5)		40		60	%
Rise Time (Note 5)	$f_r = 10\text{ MHz Ext Clock}$			12	ns
Fall Time (Note 5)	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}		200			ns
t_{HOLD}		60			ns
Output Propagation Delay	$C_L = 100\text{ pF}, R_L = 2.2\text{ k}\Omega$				
$t_{\text{PD1}}, t_{\text{PD0}}$				0.7	μs
SO, SK				1	μs
All Others					
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).

COP6620C/COP6622C/COP6640C/COP6642C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 6V
 Voltage at any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 40 mA

Total Current out of GND Pin (Sink) 48 mA
 Storage Temperature Range $-65^{\circ}C$ to $+140^{\circ}C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-55^{\circ}C \leq T_A \leq +125^{\circ}C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			$0.1 V_{CC}$	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1 \mu s$			15	mA
Supply Current during Write Operation (Note 2) CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1 \mu s$			21	mA
HALT Current (Note 3)	$V_{CC} = 5.5V$, CKI = 0 MHz		<10	40	μA
Input Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		$0.9 V_{CC}$ $0.7 V_{CC}$		$0.1 V_{CC}$ $0.2 V_{CC}$	V V V V
Hi-Z Input Leakage Input Pullup Current	$V_{CC} = 5.5V$ $V_{CC} = 4.5V$	-5 35		+5 300	μA μA
G Port Input Hysteresis (Note 5)				$0.35 V_{CC}$	V
Output Current Levels D Outputs Source Sink All Others Source (Weak Pull-Up) Source (Push-Pull Mode) Sink (Push-Pull Mode) TRI-STATE Leakage	$V_{CC} = 4.5V$, $V_{OH} = 3.8V$ $V_{CC} = 4.5V$, $V_{OL} = 1.0V$ $V_{CC} = 4.5V$, $V_{OH} = 3.2V$ $V_{CC} = 4.5V$, $V_{OH} = 3.8V$ $V_{CC} = 4.5V$, $V_{OL} = 0.4V$	0.35 9 9 0.35 1.4 -5.0		 120 +5.0	mA mA μA mA mA mA
Allowable Sink/Source Current Per Pin D Outputs (Sink) All Others				12 2.5	mA mA
Maximum Input Current (Note 4) Without Latchup (Room Temp) (Note 5)	Room Temp			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.5			V
Input Capacitance (Note 5)				7	pF
EEPROM Characteristics EEPROM Write Cycle Time EEPROM Number of Write Cycles EEPROM Data Retention				10 10,000	ms Cycle Years

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports are at TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at G6 and RESET pins must be limited to less than 14V.

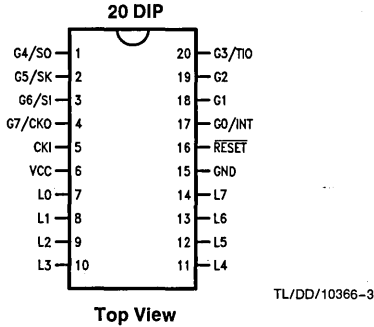
COP6620C/COP6622C/COP6640C/COP6642C (Continued)**AC Electrical Characteristics** $-55^{\circ}\text{C} < T_A < +125^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext. Crystal/Resonator (Div-by 10)		1		DC	μs
CKI Clock Duty Cycle (Note 5)		40		60	%
Rise Time (Note 5)	$f_r = 9 \text{ MHz Ext Clock}$			12	ns
Fall Time (Note 5)	$f_r = 9 \text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}		220			ns
t_{HOLD}		66			ns
Output Propagation Delay	$C_L = 100 \text{ pF}, R_L = 2.2 \text{ k}\Omega$				
$t_{\text{PD1}}, t_{\text{PDO}}$				0.8	μs
SO, SK				1.1	μs
All Others					
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).

Connection Diagrams

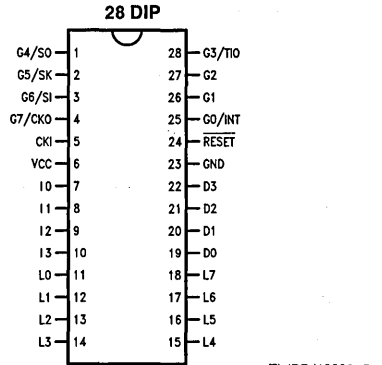
DUAL-IN-LINE PACKAGE



Top View

Order Number

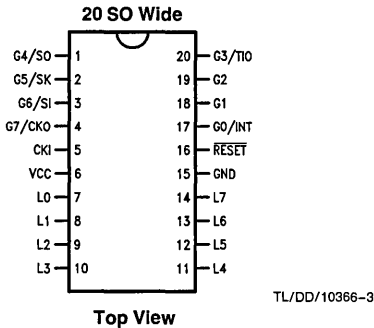
COP6622C-XXX/N, COP66L22C-XXX/N,
 COP6642C-XXX/N, COP66L42C-XXX/N,
 COP8622C-XXX/N, COP86L22C-XXX/N,
 COP8642C-XXX/N, COP86L42C-XXX/N
 See NS Package Number D20A or N20A
 (D Package for Prototypes Only)



Order Number

COP6620C-XXX/N, COP66L20C-XXX/N,
 COP6640C-XXX/N, COP66L40C-XXX/N,
 COP8620C-XXX/N, COP86L20C-XXX/N,
 COP8640C-XXX/N, COP86L40C-XXX/N
 See NS Package Number D28C or N28B
 (D Package for Prototypes Only)

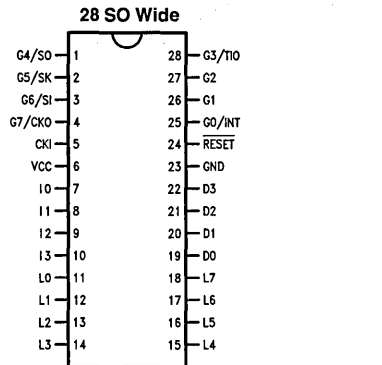
SURFACE MOUNT



Top View

Order Number

COP6622C-XXX/WM, COP66L22C-XXX/WM,
 COP6642C-XXX/WM, COP66L42C-XXX/WM,
 COP8622C-XXX/WM, COP86L22C-XXX/WM,
 COP8642C-XXX/WM, COP86L42C-XXX/WM
 See NS Package Number M20B



Order Number

COP6620C-XXX/WM, COP66L20C-XXX/WM,
 COP6640C-XXX/WM, COP66L40C-XXX/WM,
 COP8620C-XXX/WM, COP86L20C-XXX/WM,
 COP8640C-XXX/WM, COP86L40C-XXX/WM
 See NS Package Number M28B

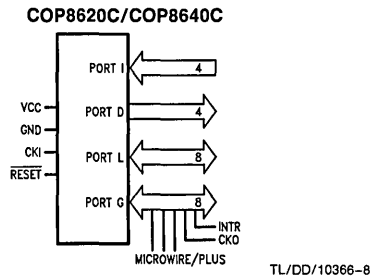
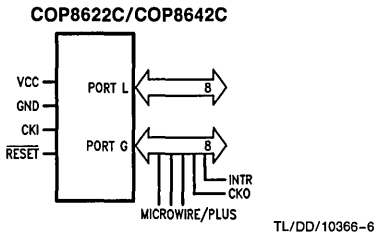


FIGURE 3

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

$\overline{\text{RESET}}$ is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0-G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below.

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when $\overline{\text{RESET}}$ goes low.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 8-bit Accumulator register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns.

PROGRAM MEMORY

Program memory for the COP8620C consists of 1024 bytes of ROM and the COP8640C consists of 2048 bytes of ROM. These bytes may hold program instructions or constant data. The program memory is addressed by the 15-bit program counter (PC). ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

The data memory address space includes on chip RAM, EEPROM, I/O and registers. Data memory is addressed directly by the instruction or indirectly through B, X and SP registers.

The COP8620C/COP8640C has 64 bytes of RAM. Sixteen bytes of RAM are mapped as "registers", these can be loaded immediately and decremented and tested. Three specific registers: X, B, and SP are mapped into this space, the other registers are available for general usage.

Any bit of data memory can be directly set, reset or tested. I/O and registers (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested.

The COP8620C/COP8640C provides 64 bytes of EEPROM for nonvolatile data memory. The data EEPROM can be read and written in exactly the same way as the RAM. All instructions that perform read and write operations on the RAM work similarly upon the data EEPROM. The data EEPROM contains all 00s when shipped by the factory.

A data EEPROM programming cycle is initiated by an instruction such as X, LD, SBIT and RBIT. The EE memory support circuitry sets the BsyERAM flag in the EEER register immediately upon beginning a data EEPROM write cycle. It will be automatically reset by the hardware at the end of the data EEPROM write cycle. The application program should test the BsyERAM flag before attempting a write operation to the data EEPROM. A second EEPROM write operation while a write operation is in progress will be ignored and the Werr flag in the EEER register will be set to indicate the error status. Once the write operation starts, nothing will stop the write operation, not by resetting the device, and not even turning off the V_{CC} will guarantee the write operation to stop.

Functional Description (Continued)

EECR AND EE SUPPORT CIRCUITRY

The EEPROM module contains EE support circuits to generate all necessary high voltage programming pulses. An EEPROM cell in the erase state is read out as a 0 and the written state as a 1. The EECR register provides control, status and test mode functions for the EE module. The EECR register bit assignments are shown below.

Werr Write Error. Writing to EEPROM while a previous write cycle is still busy, that is BsyERAM is 1, causes Werr to be set to 1 indicating error status. Werr is a Read/Write bit and is cleared by writing a 0 into it.

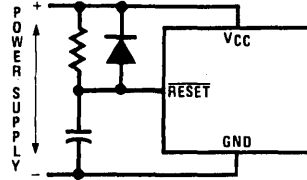
BsyERAM This bit is a read only bit and is set to 1 when EEPROM is being written. It is automatically reset by the hardware upon completion of the write operation. This bit is not cleared by reset. If the bit is set upon power up or reset, the application program should test the BsyERAM flag and wait for the flag to go low before attempting a write operation to the data EEPROM.

Bits 4 to 7 of the EECR register are used for encoding various EEPROM module test modes, most of which are for factory manufacturing tests. Except BsyERAM (bit 3) the EECR is cleared by reset. EECR is mapped into address location E0. Bit 2 can be used as flag. Bits 1 and 4 are always read as "0" and cannot be used as flags.

RESET

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the ports L and G are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L & G are cleared. Except bit 3, the EECR register is cleared.

The external RC network shown in Figure 4 should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.



TL/DD/10366-9

RC \geq 5X Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

Wr	Test Mode Codes				Unused	Unused		
Rd	Test Mode Codes				BsyERAM		Werr	
Bit	7**	6**	5**	4**	3	2*	1**	0
	R/W	R/W	R/W	R/O	R/O	R/W	R/O	R/W

*Can be used as flag bit

**Cannot be used as flag bit

Functional Description (Continued)

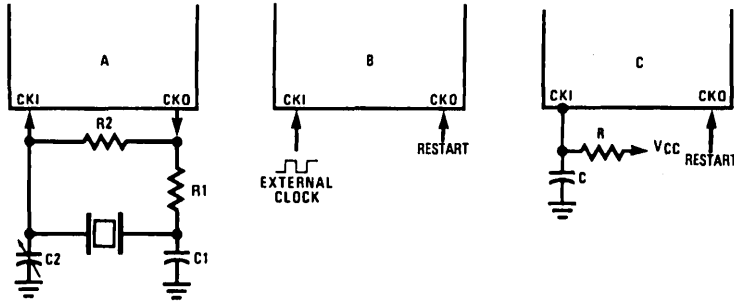


FIGURE 5. Crystal and R-C Connection Diagrams

TL/DD/10366-10

OSCILLATOR CIRCUITS

Figure 5 shows the three clock oscillator configurations available for the COP8640C.

A. CRYSTAL OSCILLATOR

The COP8620C/COP8640C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies (due to the part) as functions of the R/C component values (R/C tolerances not included).

TABLE I. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$, $V_{CC} = 5.0\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30-36	10
0	1	30	30-36	4
5.5	1	100	100	0.455

TABLE II. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$, $V_{CC} = 5.0\text{V}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)
3.3	82	2.2 to 2.7	3.7 to 4.6
5.6	100	1.1 to 1.3	7.4 to 9.0
6.8	100	0.9 to 1.1	8.8 to 10.8

Note: $3\text{k} \leq R \leq 200\text{k}$
 $50\text{ pF} \leq C \leq 200\text{ pF}$

Functional Description (Continued)

The COP8620C/COP8640C microcontroller has three mask options for configuring the clock input. The CKI and CKO pins are automatically configured upon selecting a particular option.

- Crystal/Resonator (CKI/10) CKO for crystal configuration
- External (CKI/10) CKO available as G7 input
- R/C (CKI/10) CKO available as G7 input

G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

- 1) Oscillator operating mode—I1
- 2) Internal switching current—I2
- 3) Internal leakage current—I3
- 4) Output source current—I4
- 5) DC current caused by external input not at V_{CC} or GND—I5
- 6) EEPROM current during EE read operation. This current is active during 20% of the instruction cycle time—I6
- 7) EEPROM current during write operation—I7

Thus the total current drain, I_t is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

Where

C = equivalent capacitance of the chip.

V = operating voltage

f = CKI frequency

HALT MODE

The COP8620C/COP8640C supports a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to V_r (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the RESET or by the CKO pin. A low on the RESET line reinitializes the microcontroller and starts executing from the address 0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction. This also resets the G7 data bit.

INTERRUPTS

The COP8620C/COP8640C has a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer underflow or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

Functional Description (Continued)

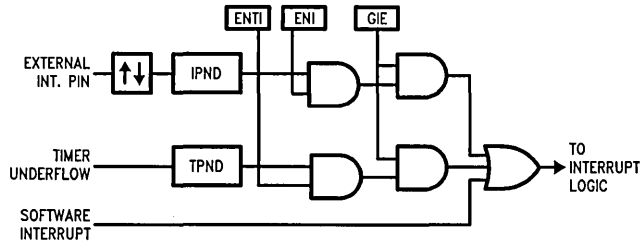


FIGURE 6. Interrupt Block Diagram

TL/DD/10366-11

DETECTION OF ILLEGAL CONDITIONS

The COP8620C/COP8640C incorporates a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined RAM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP8620C/COP8640C grows down for each subroutine call. By initializing the stack pointer to the top of RAM (02F), the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the COP8620C/COP8640C to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMS, etc.) and with other microcontrollers which support the MICROWIRE/PLUS interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 7 shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS interface with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS interface with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	2t _C
0	1	4t _C
1	x	8t _C

where,

t_C is the instruction cycle clock.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP8620C/COP8640C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. Figure 8 shows how two COP8620C/COP8640C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP8620C/COP8640C. The MICROWIRE/PLUS Master always initiates all data exchanges. (See Figure 8). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summarizes the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by appropriately setting up the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See Figure 8.)

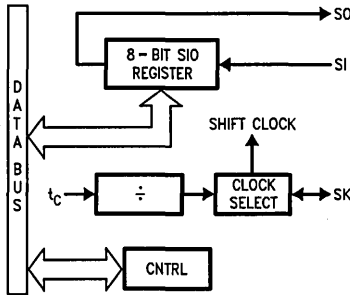
Functional Description (Continued)

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The COP8620C/COP8640C has a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table V details various timer operating modes and their requisite control settings.



TL/DD/10366-12

FIGURE 7. MICROWIRE/PLUS Block Diagram

MODE 1. TIMER WITH AUTO-LOAD REGISTER

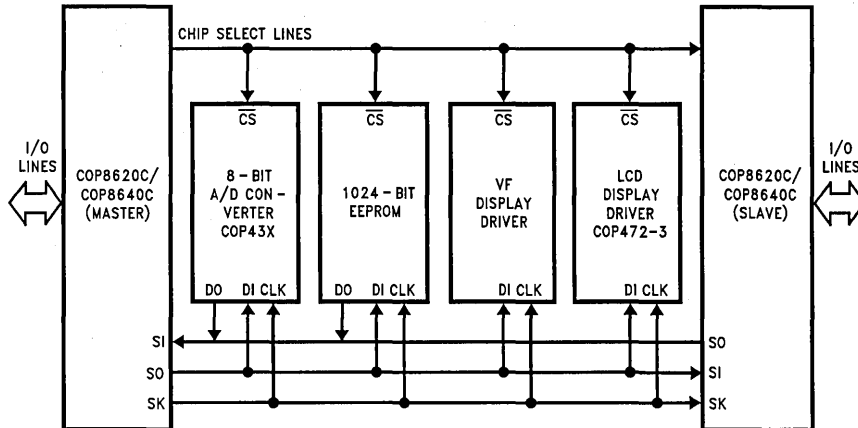
In this mode of operation, the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allow the generation of square-wave outputs or pulse width modulated outputs under software control. (See Figure 9)

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See Figure 9)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See Figure 10.)



TL/DD/10366-13

FIGURE 8. MICROWIRE/PLUS Application

Functional Description (Continued)

TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t _c
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t _c
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t _c
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t _c

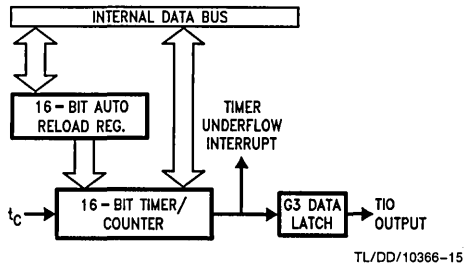


FIGURE 9. Timer/Counter Auto Reload Mode Block Diagram

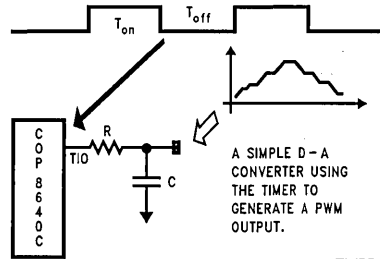


FIGURE 11. Timer Application

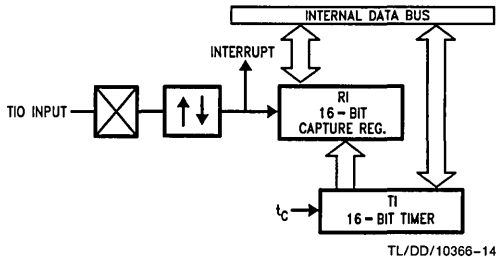


FIGURE 10. Timer Capture Mode Block Diagram

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE/PLUS control register contains the following bits:

- S1 & S0 Select the MICROWIRE/PLUS clock divide-by
- IEDG External interrupt edge polarity select (0 = rising edge, 1 = falling edge)
- MSEL Enable MICROWIRE/PLUS functions SO and SK
- TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)
- TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)
- TC2 Selects the capture mode
- TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0
BIT 7				BIT 0			

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable
- ENI External interrupt enable
- BUSY MICROWIRE/PLUS busy shifting
- IPND External interrupt pending
- ENTI Timer interrupt enable
- TPND Timer interrupt pending
- C Carry Flag
- HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE
BIT 7				BIT 0			

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
COP8620C/COP8640C	
00 to 2F	On Chip RAM Bytes
30 to 7F	Unused RAM Address Space (Reads as all Ones)
80 to BF	On Chip EEPROM (64 bytes)
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8-DB	Reserved for Port C
DC	Port D Data Register
DD-DF	Reserved for Port D
E0 to EF	On Chip Functions and Registers
E0	EECR
E1-E8	Reserved
E9	MICROWIRE/PLUS Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for COP8620C/COP8640C. The operand is the memory addressed by the B register or X register.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from -31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
MemI	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

Instruction Set

ADD ADC SUBC AND OR XOR IFEQ IFGT IFBNE DRSZ SBIT RBIT IFBIT	add add with carry subtract with carry Logical AND Logical OR Logical Exclusive-OR IF equal IF greater than IF B not equal Decrement Reg., skip if zero Set bit Reset bit If bit	$A \leftarrow A + MemI$ $A \leftarrow A + MemI + C, C \leftarrow Carry$ $HC \leftarrow Half\ Carry$ $A \leftarrow A + MemI + C, C \leftarrow Carry$ $HC \leftarrow Half\ Carry$ $A \leftarrow A\ and\ MemI$ $A \leftarrow A\ or\ MemI$ $A \leftarrow A\ xor\ MemI$ Compare A and MemI, Do next if $A = MemI$ Compare A and MemI, Do next if $A > MemI$ Do next if lower 4 bits of $B \neq Imm$ $Reg \leftarrow Reg - 1$, skip if Reg goes to 0 1 to bit, Mem (bit = 0 to 7 immediate) 0 to bit, Mem If bit, Mem is true, do next instr.
X LD A LD mem LD Reg	Exchange A with memory Load A with memory Load Direct memory Immed. Load Register memory Immed.	$A \leftrightarrow Mem$ $A \leftarrow MemI$ $Mem \leftarrow Imm$ $Reg \leftarrow Imm$
X X LD A LD A LD M	Exchange A with memory [B] Exchange A with memory [X] Load A with memory [B] Load A with memory [X] Load Memory Immediate	$A \leftrightarrow [B]$ ($B \leftarrow B \pm 1$) $A \leftrightarrow [X]$ ($X \leftarrow X \pm 1$) $A \leftarrow [B]$ ($B \leftarrow B \pm 1$) $A \leftarrow [X]$ ($X \leftarrow X \pm 1$) $[B] \leftarrow Imm$ ($B \leftarrow B \pm 1$)
CLRA INCA DECA LAID DCORA RRCA SWAPA SC RC IFC IFNC	Clear A Increment A Decrement A Load A indirect from ROM DECIMAL CORRECT A ROTATE A RIGHT THRU C Swap nibbles of A Set C Reset C If C If not C	$A \leftarrow 0$ $A \leftarrow A + 1$ $A \leftarrow A - 1$ $A \leftarrow ROM(PU,A)$ $A \leftarrow BCD\ correction\ (follows\ ADC,\ SUBC)$ $C \rightarrow A7 \rightarrow \dots \rightarrow A0 \rightarrow C$ $A7 \dots A4 \leftrightarrow A3 \dots A0$ $C \leftarrow 1, HC \leftarrow 1$ $C \leftarrow 0, HC \leftarrow 0$ If C is true, do next instruction If C is not true, do next instruction
JMPL JMP JP JSRL JSR JID RET RETSK RETI INTR NOP	Jump absolute long Jump absolute Jump relative short Jump subroutine long Jump subroutine Jump indirect Return from subroutine Return and Skip Return from Interrupt Generate an interrupt No operation	$PC \leftarrow ii$ ($ii = 15\ bits, 0\ to\ 32k$) $PC_{11..0} \leftarrow i$ ($i = 12\ bits$) $PC \leftarrow PC + r$ ($r\ is\ -31\ to\ +32, not\ 1$) $[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC \leftarrow ii$ $[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC_{11..0} \leftarrow i$ $PL \leftarrow ROM(PU,A)$ $SP+2, PL \leftarrow [SP], PU \leftarrow [SP-1]$ $SP+2, PL \leftarrow [SP], PU \leftarrow [SP-1]$, Skip next instruction $SP+2, PL \leftarrow [SP], PU \leftarrow [SP-1], GIE \leftarrow 1$ $[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC \leftarrow OFF$ $PC \leftarrow PC + 1$

COP8620C/COP8622C/COP8640C/COP8642C/COP86L20C/COP86L22C/COP86L40C/COP86L42C

Bits 7-4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	OPCODE LIST
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADCA, #i	ADCA, [B]	IFBIT 0, [B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBCA, #i	SUBCA, [B]	IFBIT 1, [B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQA, #i	IFEQA, [B]	IFBIT 2, [B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X-]	X A, [B-]	IFGTA, #i	IFGTA, [B]	IFBIT 3, [B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADDA, #i	ADDA, [B]	IFBIT 4, [B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	ANDA, #i	ANDA, [B]	IFBIT 5, [B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XORA, #i	XORA, [B]	IFBIT 6, [B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	ORA, #i	ORA, [B]	IFBIT 7, [B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0, [B]	RBIT 0, [B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1, [B]	RBIT 1, [B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2, [B]	RBIT 2, [B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X-]	LD A, [B-]	LD [B-], #i	DECA	SBIT 3, [B]	RBIT 3, [B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	*	SBIT 4, [B]	RBIT 4, [B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP + 13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5, [B]	RBIT 5, [B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP + 14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00-0EFF	JMP 0E00-0EFF	JP + 31	JP + 15	E
JP -0	JP -16	LD 0FF, #1	DRSZ 0FF	*	*	*	RETI	SBIT 7, [B]	RBIT 7, [B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

Bits 3-0

2-70

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr	
	[B]	[X]			[B+, B-]	[X+, X-]
X A,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B,Imm				1/1		
LD B,Imm				2/3		
LD Mem,Imm	2/2		3/3		2/2	
LD Reg,Imm				2/3		

(If B < 16)
(If B > 15)

* => Memory location addressed by B or X or directly.

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

BYTES and CYCLES per INSTRUCTION (Continued)

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Option List

The COP8620C/COP8640C mask programmable options are listed out below. The options are programmed at the same time as the ROM pattern to provide the user with hardware flexibility to use a variety of oscillator configuration.

OPTION 1: CKI INPUT

- = 1 Crystal/Resonator (CKI/10) CKO for crystal configuration
- = 2 External (CKI/10) CKO available as G7 input
- = 3 R/C (CKI/10) CKO available as G7 input

OPTION 2: COP8620C/COP8640C BONDING

- = 1 28 pin DIP
- = 2 N/A
- = 3 20 pin DIP
- = 4 20 SO
- = 5 28 SO

The following option information is to be sent to National along with the EPROM.

Option Data

Option 1 Value is: __ CKI Input

Option 2 Value is: __ COP Bonding

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32k bytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, dis-

abled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed or ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2k baud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS-232 serial interface cable.
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Pkg.	Voltage Range	Emulates
MHW-8640C20D5PC	20 DIP	4.5-5.5V	COP8642C, 8622C
MHW-8640C20DWPC	20 DIP	2.5-6.0V	COP8642C, 8622C
MHW-8640C28D5PC	28 DIP	4.5-5.5V	COP8640C, 8620C
MHW-8640C28DWPC	28 DIP	2.5-6.0V	COP8640C, 8620C

Development Support

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM® PC-XT®, PC-AT® or compatible	424410527-001

SIMULATOR

The COP8 Designers' Toolkit is available for evaluating National Semiconductor's COP8 microcontroller family. The kit contains programmer's manuals, device datasheets, pocket reference guides, assembler and simulator which allow the user to write, test, debug and run code on an industry standard compatible PC. The simulator has a windowed user interface and can handle script files that simulate hardware inputs, interrupts and automatic command processing. The capture file feature enables the user to record to a file current cycle count and output port changes which are caused by the program under test.

Simulator Ordering Information

Part Number	Description	Manual
COP8-TOOL-KIT	COP8 Designer's Tool Kit Assembler and Simulator	420420270-001
		424420269-001

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP8640CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	28 DIP	Multi Chip Module (MCM), UV Erasable	COP8640C, 8620C
COP8640CMHEA-X	X = 1: Crystal X = 2: External X = 3: R/C	28 LCC	MCM (Same Footprint as 28 SO), UV Erasable	COP8640C, 8620C
COP8642CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	20 DIP	MCM, UV Erasable	COP8642C, 8622C

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP and for use with Scrambler Boards	COP8640CMHD
COP8-SCRM-DIP	Scrambler Board for 20 DIP Socket	COP8642CMHD
COP8-SCRM-SBX	Scrambler Board for 28 LCC Socket	COP8640CMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP8642CMHD COP8640CMHD

Development Support (Continued)

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP
Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Setup: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs. 7 Days

COP680C/COP681C/COP880C/COP881C/ COP980C/COP981C Microcontrollers

General Description

The COP880C and COP881C are members of the COPSM microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSSM serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP880C and COP881C to the specific application. The part operates over a voltage range of 2.5 to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS
- 1 μ s instruction time
- Low current drain
 - Low current static HALT mode (Typically < 1 μ A)
- Single supply operation: 2.5 to 6.0V
- 4096 bytes ROM/128 Bytes RAM

- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - Reset master clear
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE PLUSSM serial I/O
- 44 PLCC, 36 I/O pins
- 40 DIP, 36 I/O pins
- 28 DIP and SO, 24 I/O pins
- Software selectable I/O options (TRI-STATE[®], push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature ranges: COP98XC/COP98XCH 0°C to 70°C, COP88XC -40°C to +85°C, COP68XC -55°C to +125°C.
- Form factor emulation devices
- Fully supported by National's development system

Block Diagram

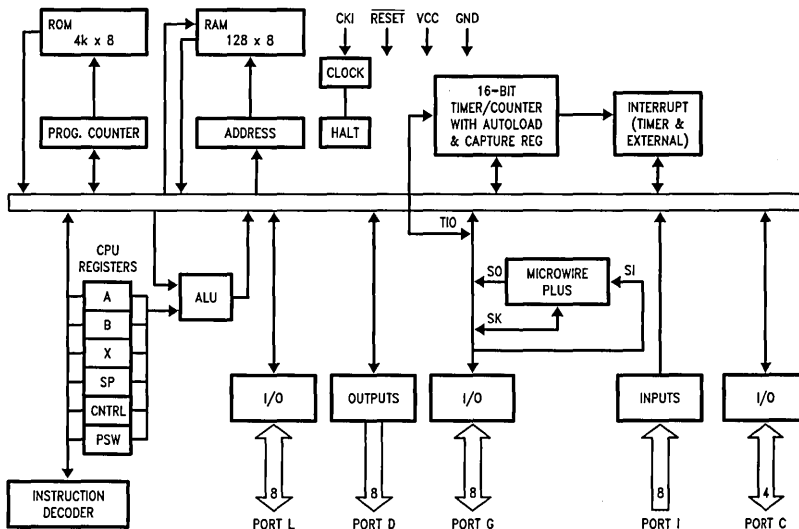


FIGURE 1

TL/DD/10802-1

COP980C/COP981C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	50 mA

Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP980XC; $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage					
98XC		2.3		4.0	V
98XCH		4.0		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			8.0	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			4.4	mA
CKI = 4 MHz	$V_{CC} = 4.0V, t_c = 2.5 \mu s$			2.2	mA
CKI = 1 MHz	$V_{CC} = 4.0V, t_c = 10 \mu s$			1.4	mA
(Note 2)					
HALT Current	$V_{CC} = 6V, CKI = 0 MHz$		<0.7	8	μA
(Note 3)	$V_{CC} = 4.0V, CKI = 0 MHz$		<0.4	5	μA
Input Levels					
RESET, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-1.0		+1.0	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
G Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.3V, V_{OH} = 1.6V$	0.2			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
	$V_{CC} = 2.3V, V_{OL} = 0.4V$	2			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
	$V_{CC} = 2.3V, V_{OH} = 1.6V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
	$V_{CC} = 2.3V, V_{OH} = 1.6V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.3V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage	$V_{CC} = 6.0V$	-1.0		+1.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 4) Without Latchup (Room Temp)	Room Temp			± 100	mA
RAM Retention Voltage, V_r (Note 5)	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

COP980C/COP981C**DC Electrical Characteristics** (Continued)

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L, C and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typ). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Note 5: To maintain RAM integrity, the voltage must not be dropped or raised instantaneously.

AC Electrical Characteristics $0^{\circ}\text{C} < T_A < +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal/Resonator or External (Div-by 10)	$V_{CC} \geq 4.0\text{V}$ $2.3\text{V} \leq V_{CC} \leq 4.0\text{V}$	1 2.5		DC DC	μs μs
R/C Oscillator Mode (Div-by 10)	$V_{CC} \geq 4.0\text{V}$ $2.3\text{V} \leq V_{CC} \leq 4.0\text{V}$	3 7.5		DC DC	μs μs
CKI Clock Duty Cycle (Note 6)	fr = Max	40		60	%
Rise Time (Note 6)	fr = 10 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 10 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.0\text{V}$ $2.3\text{V} \leq V_{CC} \leq 4.0\text{V}$	200 500			ns ns
t_{HOLD}	$V_{CC} \geq 4.0\text{V}$ $2.3\text{V} \leq V_{CC} \leq 4.0\text{V}$	60 150			ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK	$C_L = 100\text{ pF}$, $R_L = 2.2\text{ k}\Omega$ $V_{CC} \geq 4.0\text{V}$ $2.3\text{V} \leq V_{CC} \leq 4.0\text{V}$			0.7 1.75	μs μs
All Others	$V_{CC} \geq 4.0\text{V}$ $2.3\text{V} \leq V_{CC} \leq 4.0\text{V}$			1 2.5	μs μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1.0			μs

Note 6: Parameter sampled (not 100% tested).

COP880C/COP881C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V _{CC})	7V
Voltage at any Pin	-0.3V to V _{CC} + 0.3V
Total Current into V _{CC} Pin (Source)	50 mA

Total Current out of GND Pin (Sink) 60 mA

Storage Temperature Range -65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP88XC; -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		2.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V _{CC}	V
Supply Current					
CKI = 10 MHz	V _{CC} = 6V, t _c = 1 μs			8.0	mA
CKI = 4 MHz	V _{CC} = 6V, t _c = 2.5 μs			4.4	mA
CKI = 4 MHz	V _{CC} = 4.0V, t _c = 2.5 μs			2.2	mA
CKI = 1 MHz	V _{CC} = 4.0V, t _c = 10 μs			1.4	mA
(Note 2)					
HALT Current (Note 3)	V _{CC} = 6V, CKI = 0 MHz		<1	10	μA
	V _{CC} = 3.5V, CKI = 0 MHz		<0.5	6	μA
Input Levels					
RESET, CKI					
Logic High		0.9 V _{CC}			V
Logic Low				0.1 V _{CC}	V
All Other Inputs					
Logic High		0.7 V _{CC}			V
Logic Low				0.2 V _{CC}	V
Hi-Z Input Leakage	V _{CC} = 6.0V	-2		+2	μA
Input Pullup Current	V _{CC} = 6.0V	40		250	μA
G Port Input Hysteresis				0.35 V _{CC}	V
Output Current Levels					
D Outputs					
Source	V _{CC} = 4.5V, V _{OH} = 3.8V	0.4			mA
	V _{CC} = 2.5V, V _{OH} = 1.8V	0.2			mA
Sink	V _{CC} = 4.5V, V _{OL} = 1.0V	10			mA
	V _{CC} = 2.5V, V _{OL} = 0.4V	2			mA
All Others					
Source (Weak Pull-Up)	V _{CC} = 4.5V, V _{OH} = 3.2V	10		110	μA
	V _{CC} = 2.5V, V _{OH} = 1.8V	2.5		33	μA
Source (Push-Pull Mode)	V _{CC} = 4.5V, V _{OH} = 3.8V	0.4			mA
	V _{CC} = 2.5V, V _{OH} = 1.8V	0.2			mA
Sink (Push-Pull Mode)	V _{CC} = 4.5V, V _{OL} = 0.4V	1.6			mA
	V _{CC} = 2.5V, V _{OL} = 0.4V	0.7			mA
TRI-STATE Leakage	V _{CC} = 6.0V	-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 4) Without Latchup (Room Temp)	Room Temp			±100	mA
RAM Retention Voltage, V _r (Note 5)	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

COP880C/COP881C

DC Electrical Characteristics (Continued)

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L, C and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typ). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Note 5: To maintain RAM integrity, the voltage must not be dropped or raised instantaneously.

AC Electrical Characteristics $-40^{\circ}\text{C} < T_A < +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal/Resonator or External (Div-by 10) R/C Oscillator Mode (Div-by 10)	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	2.5		DC	μs
	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	7.5		DC	μs
CKI Clock Duty Cycle (Note 6) Rise Time (Note 6) Fall Time (Note 6)	fr = Max	40		60	%
	fr = 10 MHz Ext Clock			12	ns
	fr = 10 MHz Ext Clock			8	ns
Inputs t_{SETUP} t_{HOLD}	$V_{CC} \geq 4.5\text{V}$	200			ns
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	500			ns
	$V_{CC} \geq 4.5\text{V}$	60			ns
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$	150			ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$C_L = 100\text{ pF}$, $R_L = 2.2\text{ k}\Omega$				
	$V_{CC} \geq 4.5\text{V}$			0.7	μs
	$2.5\text{V} \leq V_{CC} < 4.5\text{V}$			1.75	μs
	$V_{CC} \geq 4.5\text{V}$ $2.5\text{V} \leq V_{CC} < 4.5\text{V}$			1 2.5	μs μs
MICROWIRE™ Setup Time (t_{uws}) MICROWIRE Hold Time (t_{uwh}) MICROWIRE Output Propagation Delay (t_{UPD})		20			ns
		56			ns
				220	ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		t_c			
		t_c			
		t_c			
		t_c			
		t_c			
Reset Pulse Width		1.0			μs

Note 6: Parameter sampled (not 100% tested).

Timing Diagram

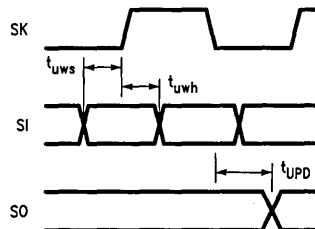


FIGURE 2. MICROWIRE/PLUS Timing

TL/DD/10802-2

COP680C/COP681C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	40 mA

Total Current Out of GND Pin (Sink)	48 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP68XC: -55°C ≤ T_A ≤ +125°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current (Note 2)				8.0	mA
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			4.4	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu s$			30	μA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0 MHz$		<10		μA
Input Levels					
\overline{RESET}, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-5		+5	μA
Input Pullup Current	$V_{CC} = 4.5V$	35		300	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.35			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	9			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	9		120	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	0.35			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 3.8V$	1.4			mA
TRI-STATE Leakage	$V_{CC} = 5.5V$	-5.0		+5.0	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				12	mA
All Others				2.5	mA
Maximum Input Current (Room Temp) without Latchup (Note 4)	Room Temp			±100	mA
RAM Retention Voltage, V_r (Note 5)	500 ns Rise and Fall Time (Min)	2.5			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and \overline{RESET} are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

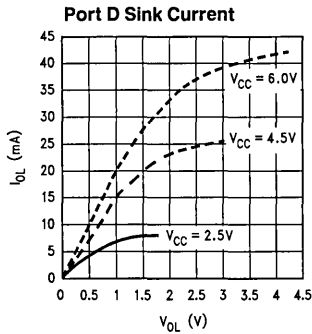
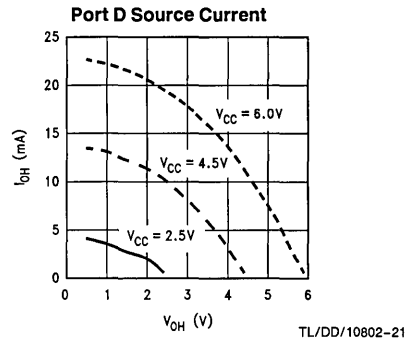
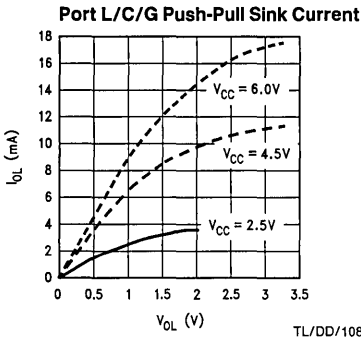
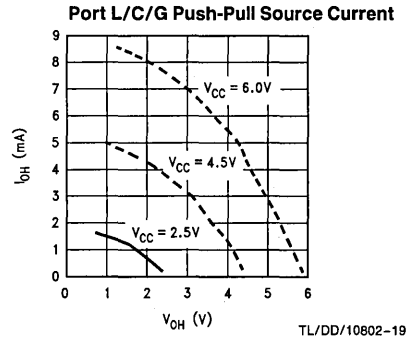
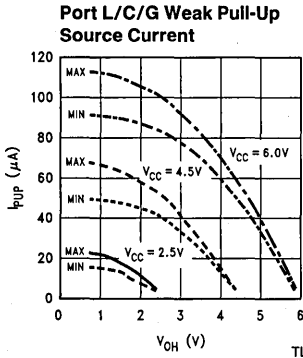
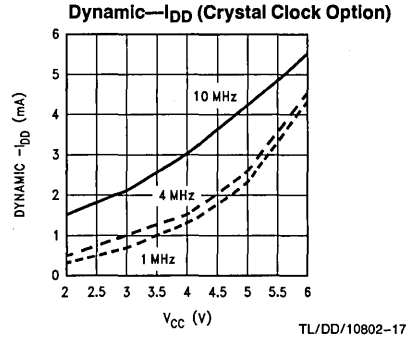
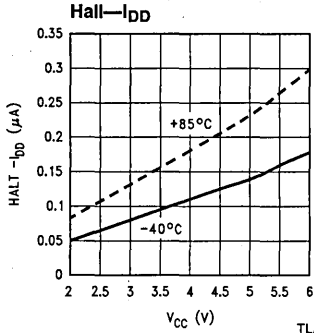
Note 5: To maintain RAM integrity, the voltage must not be dropped or raised instantaneously.

COP680C/COP681C**AC Electrical Characteristics** $-55^{\circ}\text{C} < T_A < +125^{\circ}\text{C}$ unless otherwise specified

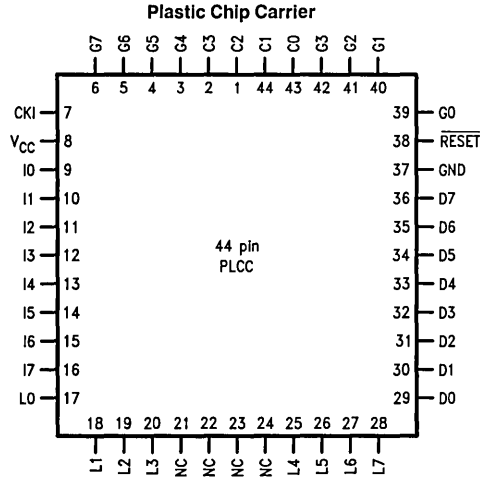
Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Ext. or Crystal/Resonant (Div-by 10)	$V_{CC} \geq 4.5V$	1		DC	μs
CKI Clock Duty Cycle (Note 6)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 6)	$f_r = 10 \text{ MHz Ext Clock}$			12	ns
Fall Time (Note 6)	$f_r = 10 \text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	220			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	66			ns
Output Propagation Delay	$R_L = 2.2k, C_L = 100 \text{ pF}$				
t_{PD1}, t_{PD0}	$V_{CC} \geq 4.5V$			0.8	μs
SO, SK	$V_{CC} \geq 4.5V$			1.1	μs
All Others	$V_{CC} \geq 4.5V$				
MICROWIRE Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1			μs

Note 6: Parameter sampled (not 100% tested).

Typical Performance Characteristics ($-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$)



Connection Diagrams

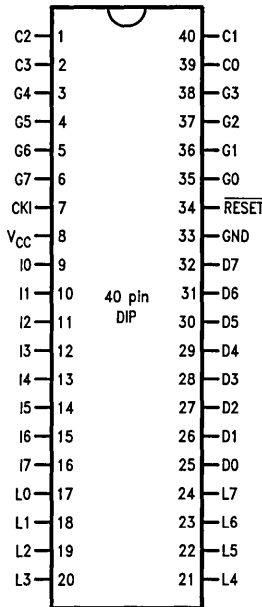


TL/DD/10802-3

Top View

Order Number COP680C-XXX/V, COP880C-XXX/V, COP980C-XXX/V or COP980CH-XXX/V

Dual-In-Line Package

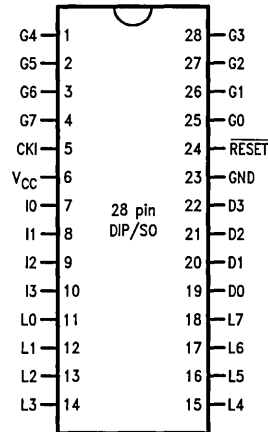


TL/DD/10802-4

Top View

Order Number COP680C-XXX/N, COP880C-XXX/N, COP980C-XXX/N or COP980CH-XXX/N

**Dual-In-Line Package (N)
and 28 Wide SO (WM)**



TL/DD/10802-5

Top View

Order Number COP881C-XXX/N, COP981C-XXX/N, COP881C-XXX/WM, COP981C-XXX/WM, COP981CH-XXX/N or COP981CH-XXX/WM

FIGURE 3

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

\overline{RESET} is the master reset input. See Reset description.

PORT I is an 8-bit Hi-Z input port.

PORT L is an 8-bit I/O port.

PORT C is a 4-bit I/O port.

Three memory locations are allocated for the L, G and C ports, one each for data register, configuration register and the input pins. Reading bits 4–7 of the C-Configuration register, data register, and input pins returns undefined data.

There are two registers associated with the L and C ports: a data register and a configuration register. Therefore, each L and C I/O bit can be individually configured under software control as shown below:

Config.	Data	Ports L and C Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

On the 28-pin part, it is recommended that all bits of Port C be configured as outputs.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated with the G port: a data register and a configuration register. Therefore, each G port bit can be individually configured under software control as shown below:

Config.	Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Since G6 and G7 are input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. The COP880C will be placed in the HALT mode by writing to the G7 bit in the G-port data register.

Six pins of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is an 8-bit output port that is preset high when \overline{RESET} goes low. Care must be exercised with the D2 pin operation. At \overline{RESET} , the external loads on this pin must ensure that the output voltages stay above 0.9 V_{CC} to prevent the chip from entering special modes. Also, keep the external loading on D2 to less than 1000 pF.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 8-bit Accumulator register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns.

PROGRAM MEMORY

Program memory for the COP880C/COP881C consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data. The program memory is addressed by the 15-bit program counter (PC). ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

The data memory address space includes on chip RAM, I/O and registers. Data memory is addressed directly by the instruction or indirectly by the B, X and SP registers.

The COP880C/COP881C has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" that can be loaded immediately, decremented or tested. Three specific registers: B, X and SP are mapped into this space, the other bytes are available for general usage.

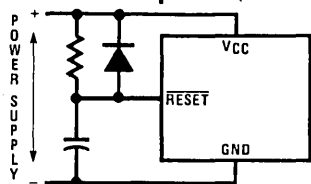
The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except the A & PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. A is not memory mapped, but bit operations can be still performed on it.

RESET

The \overline{RESET} input when pulled low initializes the microcontroller. Initialization will occur whenever the \overline{RESET} input is pulled low. Upon initialization, the ports L, G and C are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L, G and C are cleared.

The external RC network shown in *Figure 4* should be used to ensure that the \overline{RESET} pin is held low until the power supply to the chip stabilizes.

Functional Description (Continued)



$RC \geq 5X$ Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

TL/DD/10802-6

OSCILLATOR CIRCUITS

Figure 5 shows the three clock oscillator configurations available for the COP880C and COP881C.

A. CRYSTAL OSCILLATOR

The COP880C/COP881C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

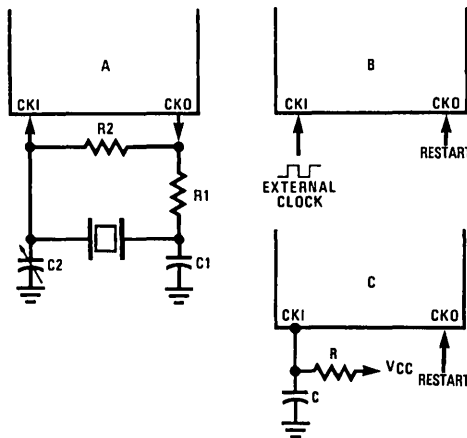
B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10802-7

FIGURE 5. Crystal and R-C Connection Diagrams

OSCILLATOR MASK OPTIONS

The COP880C and COP881C can be driven by clock inputs between DC and 10 MHz.

TABLE I. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4	$V_{CC} = 2.5V$
5.6	1	200	100-150	0.455	$V_{CC} = 5V$

TABLE II. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Note: (R/C Oscillator Configuration): $3k \leq R \leq 200k$, $50 \text{ pF} \leq C \leq 200 \text{ pF}$.

Functional Description (Continued)

The COP880C and COP881C microcontrollers have five mask options for configuring the clock input. The CKI and CKO pins are automatically configured upon selecting a particular option.

- Crystal (CKI/10); CKO for crystal configuration
- External (CKI/10); CKO available as G7 input
- R/C (CKI/10); CKO available as G7 input

G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

- 1) Oscillator operating mode— I_1
- 2) Internal switching current— I_2
- 3) Internal leakage current— I_3
- 4) Output source current— I_4
- 5) DC current caused by external input not at V_{CC} or GND — I_5

Thus the total current drain, I_t is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

Where

C = equivalent capacitance of the chip.

V = operating voltage

f = CKI frequency

HALT MODE

The COP880C and COP881C support a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to V_r (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the \overline{RESET} or by the CKO pin. A low on the \overline{RESET} line reinitializes the microcontroller and starts executing from the address

0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction. This also resets the G7 data bit.

INTERRUPTS

The COP880C and COP881C have a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer underflow or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

Functional Description (Continued)

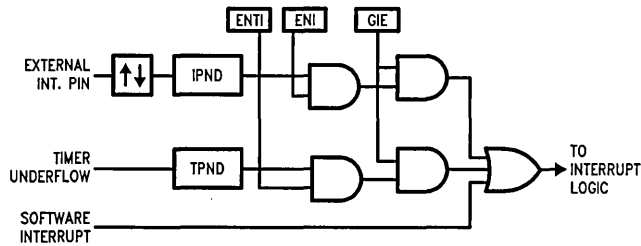


FIGURE 6. Interrupt Block Diagram

TL/DD/10802-8

DETECTION OF ILLEGAL CONDITIONS

The COP880C and COP881C incorporate a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP880C and COP881C grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the COP880C and COP881C to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMS, etc.) and with other microcontrollers which support the MICROWIRE/PLUS interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 7 shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS interface with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS interface with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	2t _C
0	1	4t _C
1	x	8t _C

where,

t_C is the instruction cycle clock.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP880C and COP881C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. Figure 8 shows how two COP880C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP880C. The MICROWIRE/PLUS Master always initiates all data exchanges. (See Figure 8). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summarizes the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by appropriately setting up the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See Figure 8.)

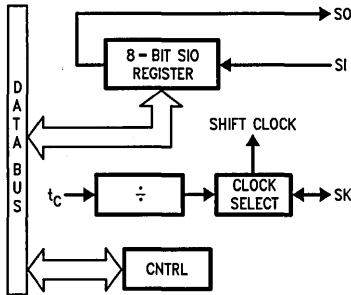
Functional Description (Continued)

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The COP880C and COP881C have a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table V details various timer operating modes and their requisite control settings.



TL/DD/10802-9

FIGURE 7. MICROWIRE/PLUS Block Diagram

MODE 1. TIMER WITH AUTO-LOAD REGISTER

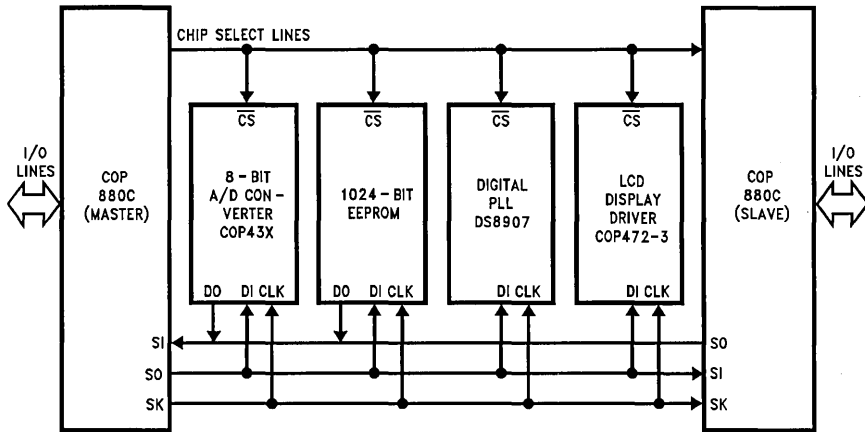
In this mode of operation, the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allows the generation of square-wave outputs or pulse width modulated outputs under software control. (See Figure 9.)

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See Figure 9)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See Figure 10.)



TL/DD/10802-10

FIGURE 8. MICROWIRE/PLUS Application

Functional Description (Continued)

TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t_c
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t_c
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t_c
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t_c

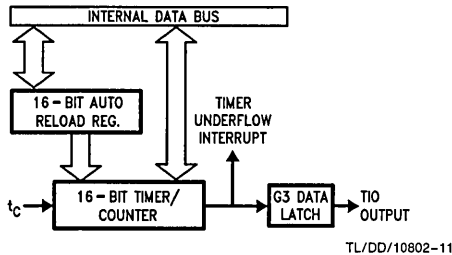


FIGURE 9. Timer/Counter Auto Reload Mode Block Diagram

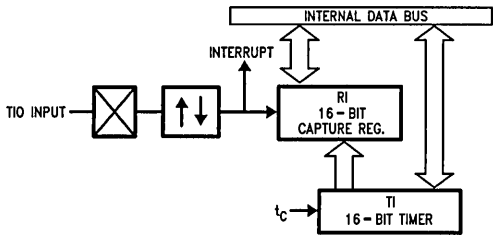


FIGURE 10. Timer Capture Mode Block Diagram

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.

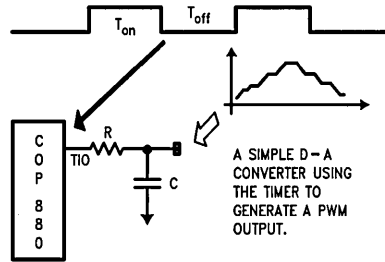


FIGURE 11. Timer Application

TL/DD/10802-13

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE/PLUS control register contains the following bits:

- S1 & S0 Select the MICROWIRE/PLUS clock divide-by
- IEDG External interrupt edge polarity select
(0 = rising edge, 1 = falling edge)
- MSEL Enable MICROWIRE/PLUS functions SO and SK
- TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)
- TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)
- TC2 Selects the capture mode
- TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0
BIT 7							BIT 0

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable
- ENI External interrupt enable
- BUSY MICROWIRE/PLUS busy shifting
- IPND External interrupt pending
- ENTI Timer interrupt enable
- TPND Timer interrupt pending
- C Carry Flag
- HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE
Bit 7							Bit 0

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for COP880C and COP881C. The operand is the memory addressed by the B register or X register.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from -31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
00 to 6F	On Chip RAM Bytes
70 to 7F	Unused RAM Address Space (Reads as all Ones)
80 to BF	Expansion Space for on Chip EERAM
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD-DF	Reserved for Port D
E0 to EF	On Chip Functions and Registers
E0-E7	Reserved for Future Parts
E8	Reserved
E9	MICROWIRE/PLUS Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
Meml	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set

ADD ADC SUBC AND OR XOR IFEQ IFGT IFBNE DRSZ SBIT RBIT IFBIT	add add with carry subtract with carry Logical AND Logical OR Logical Exclusive-OR IF equal IF greater than IF B not equal Decrement Reg. ,skip if zero Set bit Reset bit If bit	A ← A + Meml A ← A + Meml + C, C ← Carry HC ← Half Carry A ← A + Meml + C, C ← Carry HC ← Half Carry A ← A and Meml A ← A or Meml A ← A xor Meml Compare A and Meml, Do next if A = Meml Compare A and Meml, Do next if A > Meml Do next if lower 4 bits of B ≠ Imm Reg ← Reg - 1, skip if Reg goes to 0 1 to bit, Mem (bit = 0 to 7 immediate) 0 to bit, Mem If bit, Mem is true, do next instr.
X LD A LD mem LD Reg	Exchange A with memory Load A with memory Load Direct memory Immed. Load Register memory Immed.	A ↔ Mem A ← Meml Mem ← Imm Reg ← Imm
X X LD A LD A LD M	Exchange A with memory [B] Exchange A with memory [X] Load A with memory [B] Load A with memory [X] Load Memory Immediate	A ↔ [B] (B ← B ± 1) A ↔ [X] (X ← X ± 1) A ← [B] (B ← B ± 1) A ← [X] (X ← X ± 1) [B] ← Imm (B ← B ± 1)
CLRA INCA DECA LAID DCORA RRCA SWAPA SC RC IFC IFNC	Clear A Increment A Decrement A Load A indirect from ROM DECIMAL CORRECT A ROTATE A RIGHT THRU C Swap nibbles of A Set C Reset C If C If not C	A ← 0 A ← A + 1 A ← A - 1 A ← ROM(PU,A) A ← BCD correction (follows ADC, SUBC) C → A7 → ... → A0 → C A7 ... A4 ↔ A3 ... A0 C ← 1, HC ← 1 C ← 0, HC ← 0 If C is true, do next instruction If C is not true, do next instruction
JMPL JMP JP JSRL JSR JID RET RETSK RETI INTR NOP	Jump absolute long Jump absolute Jump relative short Jump subroutine long Jump subroutine Jump indirect Return from subroutine Return and Skip Return from Interrupt Generate an interrupt No operation	PC ← ii (ii = 15 bits, 0 to 32k) PC11..0 ← i (i = 12 bits) PC ← PC + r (r is -31 to +32, not 1) [SP] ← PL,[SP-1] ← PU,SP-2,PC ← ii [SP] ← PL,[SP-1] ← PU,SP-2,PC11..0 ← i PL ← ROM(PU,A) SP + 2,PL ← [SP],PU ← [SP-1] SP + 2,PL ← [SP],PU ← [SP-1],Skip next instruction SP + 2,PL ← [SP],PU ← [SP-1],GIE ← 1 [SP] ← PL,[SP-1] ← PU,SP-2,PC ← 0FF PC ← PC + 1

Bits 7-4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	OPCODE LIST
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0, [B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A, [B]	IFBIT 1, [B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A, [B]	IFBIT 2, [B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X-]	X A, [B-]	IFGT A, #i	IFGT A, [B]	IFBIT 3, [B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A, [B]	IFBIT 4, [B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	AND A, #i	AND A, [B]	IFBIT 5, [B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A, [B]	IFBIT 6, [B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	OR A, #i	OR A, [B]	IFBIT 7, [B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0, [B]	RBIT 0, [B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1, [B]	RBIT 1, [B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2, [B]	RBIT 2, [B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X-]	LD A, [B-]	LD [B-], #i	DECA	SBIT 3, [B]	RBIT 3, [B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	*	SBIT 4, [B]	RBIT 4, [B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP + 13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5, [B]	RBIT 5, [B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP + 14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00-0EFF	JMP 0E00-0EFF	JP + 31	JP + 15	E
JP -0	JP -16	LD 0FF, #1	DRSZ 0FF	*	*	*	RETI	SBIT 7, [B]	RBIT 7, [B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

Bits 3-0

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr		
	[B]	[X]			[B+, B-]	[X+, X-]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B,Imm				1/1			(If B < 16)
LD B,Imm				2/3			(If B > 15)
LD Mem,Imm	2/2		3/3		2/2		
LD Reg,Imm				2/3			

* => Memory location addressed by B or X or directly.

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

BYTES and CYCLES per INSTRUCTION (Continued)

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Option List

The COP880C/COP881C mask programmable options are listed out below. The options are programmed at the same time as the ROM pattern to provide the user with hardware flexibility to use a variety of oscillator configuration.

OPTION 1: CKI INPUT

- = 1 Crystal (CKI/10) CKO for crystal configuration
- = 2 External (CKI/10) CKO available as G7 input
- = 3 R/C (CKI/10) CKO available as G7 input

OPTION 2: COP880C/COP881C BONDING

- = 1 44-Pin PLCC
- = 2 40-Pin DIP
- = 3 28-Pin SO
- = 4 28-Pin DIP

The following option information is to be sent to National along with the EPROM.

Option Data

- Option 1 Value__is: CKI Input
- Option 2 Value__is: COP Bonding

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real-time, full-speed emulation up to 10 MHz, 32 kbytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy-to-use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS-232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-880C28D5PC	28 DIP	4.5V–5.5V	COP820C, 840C, 881C, 8781C
MHW-880C28DWPC	28 DIP	2.5V–6.0V	COP820C, 840C, 881C, 8781C
MHW-880C40D5PC	40 DIP	4.5V–5.5V	COP880C, 8780C
MHW-880C40DWPC	40 DIP	2.5V–6.0V	COP880C, 8780C
MHW-880C44D5PC	44 PLCC	4.5V–5.5V	COP880C, 8780C
MHW-880C44DWPC	44 PLCC	2.5V–6.0V	COP880C, 8780C

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM PC-XT®, PC-AT®, or compatible	424410527-001

COP880C/COP881C/COP880C/COP881C/COP980C/COP981C

Development Support (Continued)

SINGLE-CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. Two types of single-chip emulators are available: Multi-Chip Module emulators, which combine the microcontroller-die and an EPROM-die in one package,

and emulators where the microcontroller's standard ROM has been replaced with an on-chip EPROM. For more detailed information, refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

Single-Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP880CMHEL-X	X = 1: Crystal X = 2: External X = 3: R/C	44 LDCC	Multi-Chip Module (MCM), UV Erasable	COP880C
COP880CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	40 DIP	MCM, UV Erasable	COP880C
COP881CMHD-X	X = 1: Crystal X = 2: External X = 3: R/C	28 DIP	MCM, UV Erasable	COP881C
COP881CMHEA-X	X = 1: Crystal X = 2: External X = 3: R/C	28 LCC	MCM (Same Footprint as 28 SO), UV Erasable	COP881C
COP8780CV	Programmable	44 PLCC	One-Time Programmable (OTP)	COP880C
COP8780CEL	Programmable	44 LDCC	UV Erasable	COP880C
COP8780CN	Programmable	40 DIP	OTP	COP880C
COP8780CJ	Programmable	40 DIP	UV Erasable	COP880C
COP8781CN	Programmable	28 DIP	OTP	COP881C
COP8781CJ	Programmable	28 DIP	UV Erasable	COP881C
COP8781CWM	Programmable	28 SO	OTP	COP881C
COP8781CMC	Programmable	28 SO	UV Erasable	COP881C

PROGRAMMING SUPPORT

Programming of the single-chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single-chip emulator and vice versa. Data I/O supports COP8 emulator

device programming with its UniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 FAX: (206) 882-1043

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for Use with Scrambler Boards	COP881CMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP Socket	COP880CMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP880CMHEL
COP8-SCRM-SBX	MCM Scrambler Board for 28 LCC Socket	COP881CMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP881CMHD, COP880CMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC Scrambler Board	COP880CMEL, COP881CMHD
COP8-PRGM-87A	Duplicator Board with COP87XX Scrambler for 28 DIP, 28 SO, and 40 DIP	COP8781CN, COP8781CJ, COP8781CWM, COP8781CMC, COP8780CN, COP8780CJ
COP8-PRGM-87B	Duplicator Board with COP87XX Scrambler for 44 PLCC/LDCC	COP8780CV, COP8780CEL

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 baud

Setup: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs. 7 Days



COP688CL/COP684CL, COP888CL/COP884CL, COP988CL/COP984CL Single-Chip microCMOS Microcontroller

General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. The COP888CL is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μ s instruction cycle time
- 4096 bytes on-board ROM
- 128 bytes on-board RAM
- Single supply operation: 2.5V–6V
- MICROWIRE/PLUS™ serial I/O
- WATCHDOG™ and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Timers TA, TB (Each with 2 Interrupts)
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - Default VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PLCC or 40 N or 28 N or 28 SO
 - 44 PLCC with 39 I/O pins
 - 40 N with 33 I/O pins
 - 28 SO or 28 N, each with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: 0°C to +70°C,
–40°C to +85°C,
–55°C to +125°C
- Single chip hybrid emulation device COP888CLMH
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram

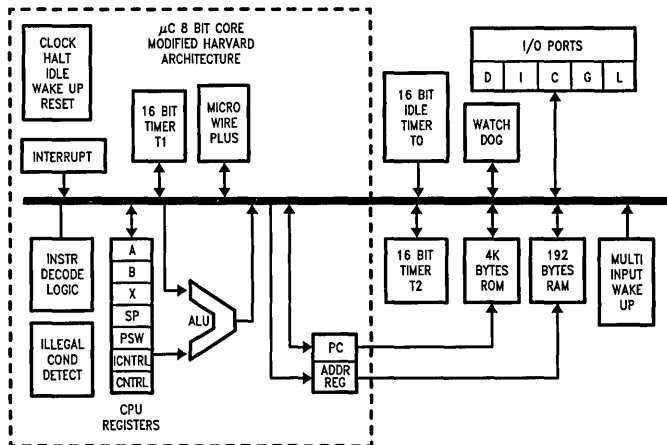


FIGURE 1. COP888CL Block Diagram

TL/DD/9766–1

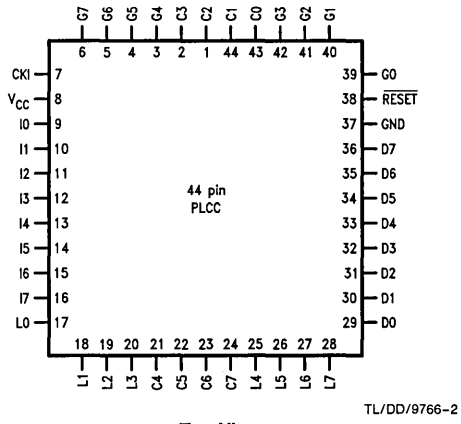
General Description (Continued)

It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), and two power savings modes (HALT and IDLE), both with a multi-

sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CL operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

Connection Diagrams

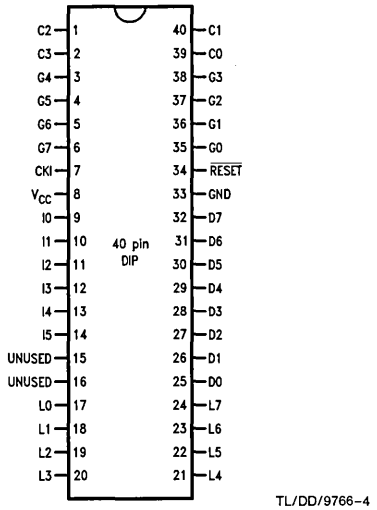
Plastic Chip Carrier



Top View

Order Number COP688CL-XXX/V, COP888CL-XXX/V or COP988CL-XXX/V
See NS Plastic Chip Package Number V44A

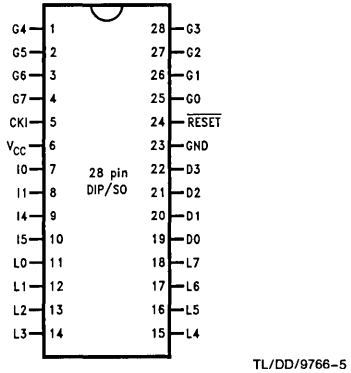
Dual-In-Line Package



Top View

Order Number COP688CL-XXX/N, COP888CL-XXX/N or COP988CL-XXX/N
See NS Molded Package Number N40A

Dual-In-Line Package



Top View

Order Number COP688CL-XXX/N, COP884CL-XXX/N or COP984CL-XXX/N
See NS Molded Package Number N28B

Order Number COP684CL-XXX/WM,
COP884CL-XXX/WM or COP984CL-XXX/WM
See NS Surface Mount Package Number M28B

FIGURE 2. COP888CL Connection Diagrams

Connection Diagrams (Continued)

COP888CL Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU		12	18	18
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT RESTART		4	6	6
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
I0	I			7	9	9
I1	I			8	10	10
I2	I				11	11
I3	I				12	12
I4	I			9	13	13
I5	I			10	14	14
I6	I					15
I7	I					16
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
Unused*					16	
Unused*					15	
VCC				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

* = On the 40-pin package Pins 15 and 16 must be connected to GND.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

COP98XCL: 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage COP98XCL COP98XCLH		2.5		4.0	V
		4.0		6.0	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2) CKI = 10 MHz CKI = 4 MHz CKI = 4 MHz CKI = 1 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			12.5	mA
	$V_{CC} = 6V, t_c = 2.5 \mu s$			5.5	mA
	$V_{CC} = 4V, t_c = 2.5 \mu s$			2.5	mA
	$V_{CC} = 4V, t_c = 10 \mu s$			1.4	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 \text{ MHz}$		<0.7	8	μA
	$V_{CC} = 4V, CKI = 0 \text{ MHz}$		<0.4	5	μA
IDLE Current CKI = 10 MHz CKI = 4 MHz CKI = 1 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			3.5	mA
	$V_{CC} = 6V, t_c = 2.5 \mu s$			2.5	mA
	$V_{CC} = 4V, t_c = 10 \mu s$			0.7	mA
Input Levels RESET Logic High Logic Low CKI (External and Crystal Osc. Modes) Logic High Logic Low All Other Inputs Logic High Logic Low		0.8 V_{CC}			V
				0.2 V_{CC}	V
		0.7 V_{CC}			V
				0.2 V_{CC}	V
		0.7 V_{CC}			V
				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	-1		+1	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels D Outputs Source Sink All Others Source (Weak Pull-Up Mode) Source (Push-Pull Mode) Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
	$V_{CC} = 4V, V_{OL} = 1V$	10			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	2.0			mA
	$V_{CC} = 4V, V_{OH} = 2.7V$	10		100	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
	$V_{CC} = 4V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
TRI-STATE Leakage	$V_{CC} = 6.0\text{V}$	-1		+1	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Note 5)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal or Resonator	$4\text{V} \leq V_{CC} \leq 6\text{V}$	1		DC	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	2.5		DC	μs
R/C Oscillator	$4\text{V} \leq V_{CC} \leq 6\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	7.5		DC	μs
CKI Clock Duty Cycle (Note 4)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Fall Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Inputs					
t_{SETUP}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	200			ns
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	500			ns
t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	60			ns
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	150			ns
Output Propagation Delay	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$				
$t_{\text{PD1}}, t_{\text{PD0}}$	$4\text{V} \leq V_{CC} \leq 6\text{V}$			0.7	μs
SO, SK	$2.5\text{V} \leq V_{CC} < 4\text{V}$			1.75	μs
All Others	$4\text{V} \leq V_{CC} \leq 6\text{V}$			1	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

Note 4: Parameter sampled (not 100% tested).

Note 5: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP88XCL: -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			5.5	mA
CKI = 4 MHz	$V_{CC} = 4V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 4V, t_c = 10 \mu s$			1.4	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$ $V_{CC} = 4V, CKI = 0 MHz$		<1 <0.5	10 6	μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 4V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 4V, t_c = 10 \mu s$			0.7	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	-1		+1	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	0.4 0.2			mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	10 2.0			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	10 2.5		100 33	μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	0.4 0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	1.6 0.7			mA
TRI-STATE Leakage	$V_{CC} = 6.0V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				15 3	mA mA
Maximum Input Current without Latchup (Note 5)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal or Resonator	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	1 2.5		DC DC	μs μs
R/C Oscillator	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	3 7.5		DC DC	μs μs
CKI Clock Duty Cycle (Note 4)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Fall Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Inputs t_{SETUP} t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	200 500 60 150			ns ns ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100\text{ pF}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$			0.7 1.75 1 2.5	μs μs μs μs
MICROWIRE Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Parameter sampled (not 100% tested).

Note 5: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Electrical Specifications

DC ELECTRICAL SPECIFICATIONS

COP688CL Absolute Specifications

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	90 mA
Total Current out of GND Pin (Sink)	100 mA
Storage Temperature Range	-65°C to $+150^{\circ}\text{C}$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP68XCL: $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1 V_{CC}$	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu\text{s}$			12.5	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu\text{s}$			5.5	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, \text{CKI} = 0 \text{ MHz}$		<10	30	μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu\text{s}$			3.5	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu\text{s}$			2.5	mA
Input Levels					
RESET					
Logic High		$0.8 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
CKI (External and Crystal Osc. Modes)					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 5.5V, V_{IN} = 0V$	-5		+5	μA
Input Pullup Current	$V_{CC} = 5.5V, V_{IN} = 0V$	35		400	μA
G and L Port Input Hysteresis				$0.35 V_{CC}$	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	9			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	9.0		140	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.4			mA
TRI-STATE Leakage	$V_{CC} = 5.5V$	-5.0		+5.0	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

DC Electrical Characteristics $-55^{\circ}\text{C} \leq T_A \leq +25^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				12	mA
All others				2.5	mA
Maximum Input Current without Latchup (Note 5)				150	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The Clock Monitor and the comparators are disabled.

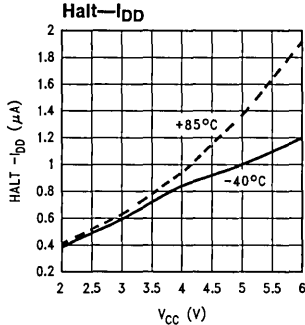
AC Specifications for COP688CL**AC Electrical Characteristics** $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator, or External Oscillator	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
R/C Oscillator (div-by 10)	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
CKI Clock Duty Cycle (Note 4)	$f_r = \text{Max}$	45		55	%
(Crystal Resonator or External Clock)					
Rise Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			12	ns
Fall Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5\text{V}$	200			ns
t_{HOLD}	$V_{CC} \geq 4.5\text{V}$	60			ns
Output Propagation Delay	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$				
$t_{\text{PD1}}, t_{\text{PD0}}$	$V_{CC} \geq 4.5\text{V}$			0.7	μs
SO, SK	$V_{CC} \geq 4.5\text{V}$			1	μs
All Others	$V_{CC} \geq 4.5\text{V}$				
MICROWIRE Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

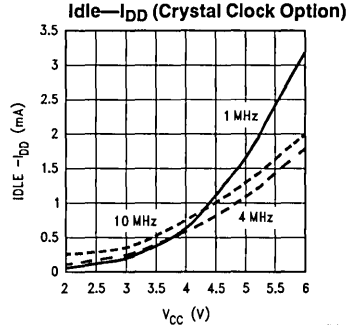
Note 4: Parameter sampled (not 100% tested).

Note 5: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

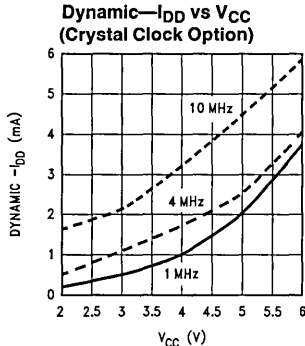
Typical Performance Characteristics ($-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$)



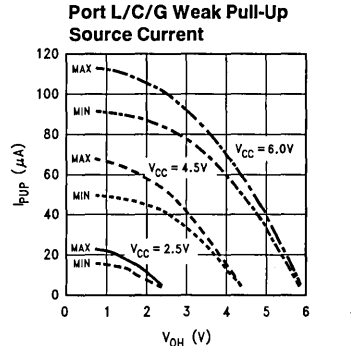
TL/DD/9766-27



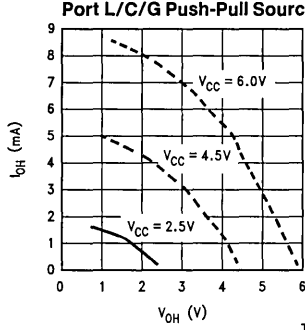
TL/DD/9766-28



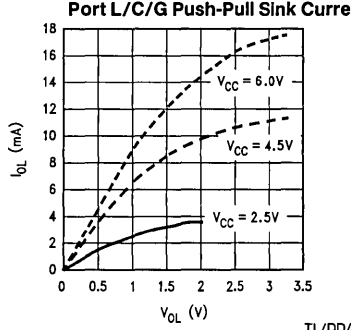
TL/DD/9766-29



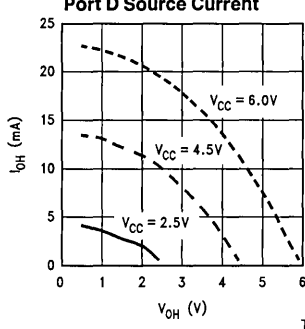
TL/DD/9766-30



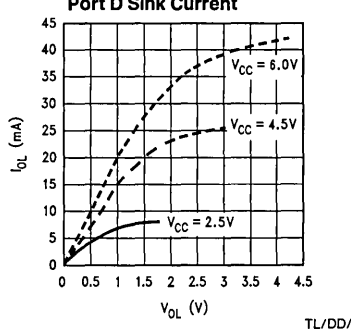
TL/DD/9766-31



TL/DD/9766-32



TL/DD/9766-33



TL/DD/9766-34

AC Electrical Characteristics (Continued)

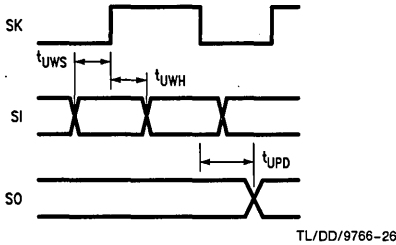


FIGURE 2. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CL contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports G and L), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CL memory map for the various addresses associated with the I/O ports.) Figure 3 shows the I/O port configurations for the COP888CL. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

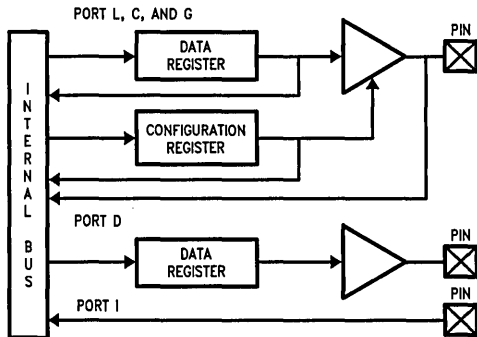


FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wakeup (MIWU) on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B.

Port L has the following alternate features:

- L0 MIWU
- L1 MIWU
- L2 MIWU
- L3 MIWU
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin, but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin or general purpose input (R/C clock configuration), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE™ Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Pin Descriptions (Continued)

Port G has the following dedicated functions:

- G1 WDOUT WATCHDOG and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

Port C is an 8-bit I/O port. The 40-pin device does not have a full complement of Port C pins. The unavailable pins are not terminated. A read operation for these unterminated pins will return unpredictable values.

Port I is an 8-bit Hi-Z input port. The 40-pin device does not have a full complement of Port I pins. Pins 15 and 16 on this package must be connected to GND.

The 28-pin device has four I pins (I0, I1, I4, I5). The user should pay attention when reading port I to the fact that I4 and I5 are in bit positions 4 and 5 rather than 2 and 3.

The unavailable pins (I4–I7) are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes into account by either masking or restricting the accesses to bit operations. The unterminated port I pins will draw power only when addressed.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CL is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CL architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_c) cycle time.

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CL consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors

for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CL vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The COP888CL has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

The instruction set of the COP888CL permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CL (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

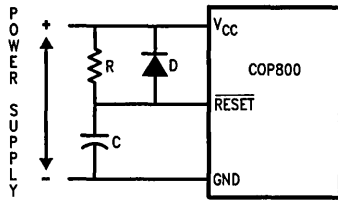
Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for Ports L, G, and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WATCHDOG and/or Clock Monitor error output pin. Port D is initialized high with RESET. The PC, PSW, CNTRL, ICNTRL, and T2CNTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The Stack Pointer, SP, is initialized to 06F Hex.

The COP888CL comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, and with both the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor detector circuits are inhibited during reset. The WATCHDOG service window bits are initialized to the maximum WATCHDOG service window of 64k t_c clock cycles. The Clock Monitor bit is initialized high, and will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16–32 t_c clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 4* should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes.

Reset (Continued)



TL/DD/9766-7

$RC > 5 \times$ Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

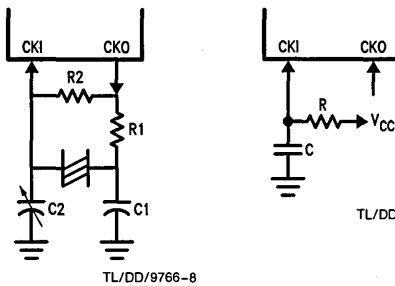
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9766-8

FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4	$V_{CC} = 5.0V$
0	1	200	100-150	0.455	$V_{CC} = 5V$

TABLE B. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Note: $3k \leq R \leq 200k$, $50 \text{ pF} \leq C \leq 200 \text{ pF}$

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. Clock Monitor current when enabled—I6

Thus the total current drain, It, is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

where C = equivalent capacitance of the chip
 V = operating voltage
 f = CKI frequency

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively

Control Registers (Continued)

T1C0	Timer T1 Start/Stop control in timer modes 1 and 2
	Timer T1 Underflow Interrupt Pending Flag in timer mode 3
T1C1	Timer T1 mode control bit
T1C2	Timer T1 mode control bit
T1C3	Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7							Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt
BUSY	MICROWIRE/PLUS busy shifting flag
EXPND	External interrupt pending
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
T1PNDA	Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
C	Carry Flag
HC	Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE
Bit 7							Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB	Timer T1 Interrupt Enable for T1B Input capture edge
T1PNDB	Timer T1 Interrupt Pending Flag for T1B capture edge

μ WEN	Enable MICROWIRE/PLUS interrupt
μ WPND	MICROWIRE/PLUS interrupt pending
T0EN	Timer T0 Interrupt Enable (Bit 12 toggle)
T0PND	Timer T0 Interrupt pending
LPEN	L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μ WPND	μ WEN	T1PNDB	T1ENB
Bit 7							Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB	Timer T2 Interrupt Enable for T2B Input capture edge
T2PNDB	Timer T2 Interrupt Pending Flag for T2B capture edge
T2ENA	Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
T2PNDA	Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
T2C0	Timer T2 Start/Stop control in timer modes 1 and 2
	Timer T2 Underflow Interrupt Pending Flag in timer mode 3
T2C1	Timer T2 mode control bit
T2C2	Timer T2 mode control bit
T2C3	Timer T2 mode control bit

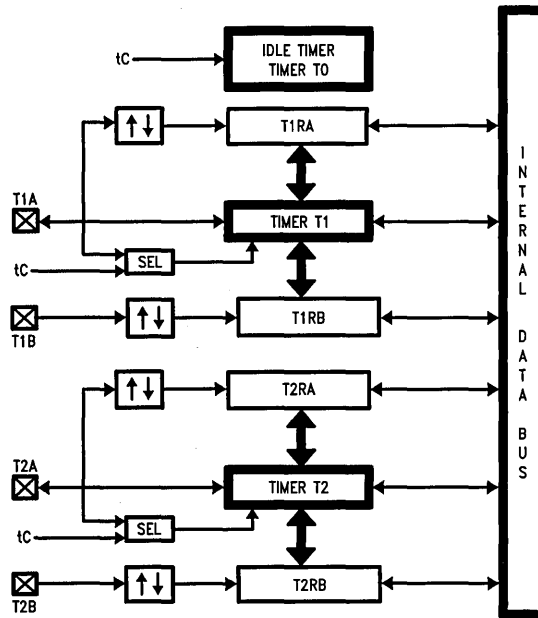
T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

Timers

The COP888CL contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

Figure 6 shows a block diagram for the timers on the COP888CL.

Timers (Continued)



TL/DD/9766-11

FIGURE 6. Timers for the COP888CL

TIMER T0 (IDLE TIMER)

The COP888CL supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c . The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

- Exit out of the Idle Mode (See Idle Mode description)
- WATCHDOG logic (See WATCHDOG description)
- Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_c = 1 \mu\text{s}$). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1 AND TIMER T2

The COP888CL has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2, are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer

block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CL to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits Tx3, Tx2, and Tx1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CL to generate a PWM signal with very minimal user intervention.

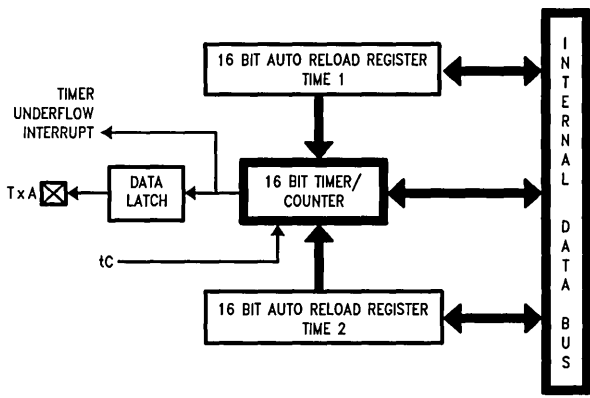
The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_c . Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, Tx3, Tx2 and Tx1 set up the timer for PWM mode operation.

Figure 7 shows a block diagram of the timer in PWM mode.

Timers (Continued)



TL/DD/9766-13

FIGURE 7. Timer in PWM Mode

The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPND A and TxPND B. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

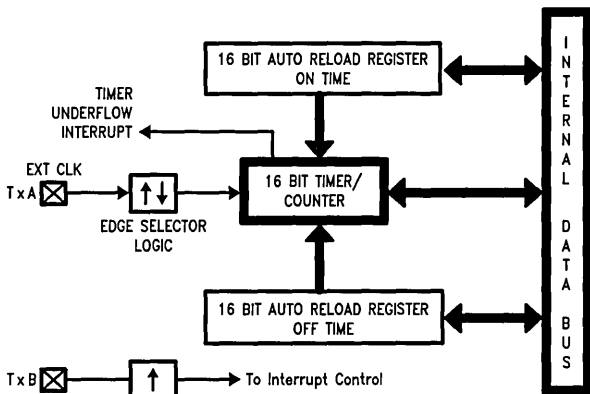
Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, Tx C3, Tx C2 and Tx C1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPND A pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin Tx B can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the Tx B input pin is latched into the TxPND B flag.

Figure 8 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.



TL/DD/9766-14

FIGURE 8. Timer in External Event Counter Mode

Timers (Continued)

Mode 3. Input Capture Mode

The COP888CL can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_c rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxCO pending flag (the TxCO control bit serves as the timer under-

flow interrupt pending flag in the Input Capture mode). Consequently, the TxCO control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxCO pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 9 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

The timers T1 and T2 have identical control structures. The control bits and their functions are summarized below.

TxCO	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop
	Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag
	1 = Timer Interrupt Enabled
	0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control

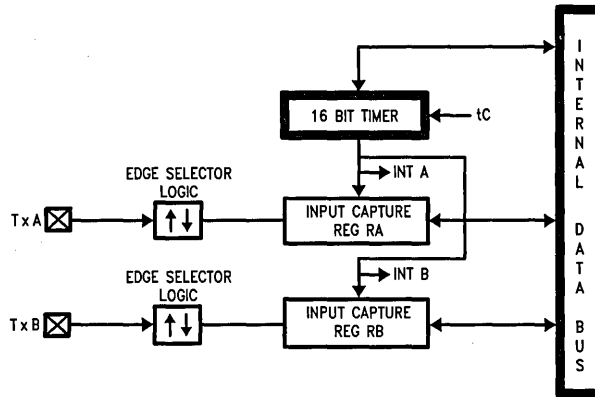


FIGURE 9. Timer in Input Capture Mode

TL/DD/9766-15

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c

Power Save Modes

The COP888CL offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CL is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock, timers, are stopped. The WATCHDOG logic on the COP888CL is disabled during the HALT mode. However, the clock monitor circuitry, if enabled, remains active and will cause the WATCHDOG output pin (WDOUT) to go low. If the HALT mode is used and the user does not want to activate the WDOUT pin, the Clock Monitor should be disabled after the device comes out of reset (resetting the Clock Monitor control bit with the first write to the WDSVR register). In the HALT mode, the power requirements of the COP888CL are minimal and the applied voltage (V_{CC}) may be decreased to V_r ($V_r = 2.0V$) without altering the state of the machine.

The COP888CL supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is

with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the \overline{RESET} pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

Power Save Modes (Continued)

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

The COP888CL has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CL will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CL cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WATCHDOG detector circuit is inhibited during the HALT mode. However, the clock monitor circuit, if enabled, remains active during HALT mode in order to ensure a clock monitor error if the COP888CL inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CL is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the WATCHDOG logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CL can be returned to normal operation with a reset, or with a Multi-Input Wake-up from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_c = 1 \mu\text{s}$) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the TOPND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the TOPND bit gets set, the COP888CL will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CL will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CL from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 10 shows the Multi-Input Wakeup logic for the COP888CL microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CL to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```

RBIT 5, WKEN
SBIT 5, WKEDG
RBIT 5, WKPND
SBIT 5, WKEN
  
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CL will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

The WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to

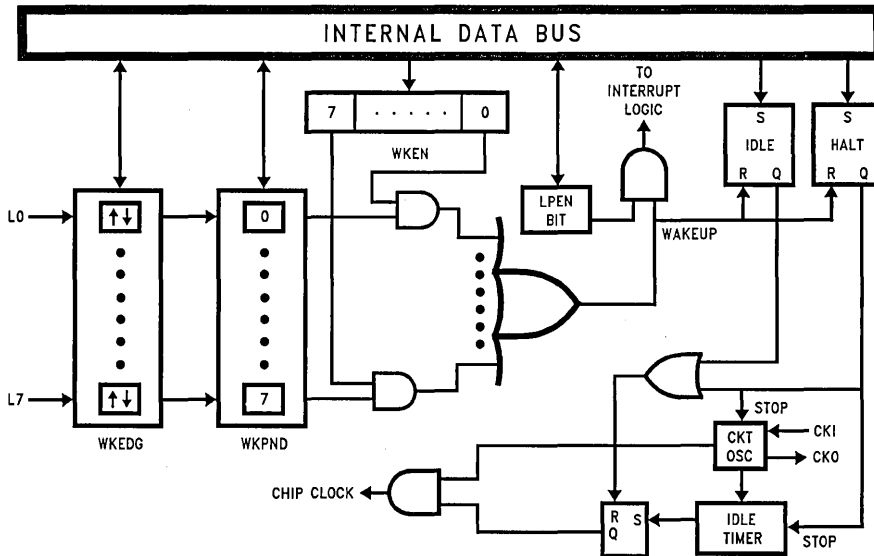


FIGURE 10. Multi-Input Wake Up Logic

TL/DD/9766-16

Multi-Input Wakeup (Continued)

be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CL out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CL will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CL will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CL to execute instructions. In this

case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

Interrupts

The COP888CL supports a vectored interrupt scheme. It supports a total of ten interrupt sources. The following table lists all the possible COP888CL interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE-0yFF
	Reserved	for Future Use	0yFC-0yFD
(2)	External	Pin G0 Edge	0yFA-0yFB
(3)	Timer T0	Underflow	0yF8-0yF9
(4)	Timer T1	T1A/Underflow	0yF6-0yF7
(5)	Timer T1	T1B	0yF4-0yF5
	MICROWIRE/PLUS	BUSY Goes Low	0yF2-0yF3
	Reserved	for Future Use	0yF0-0yF1
	Reserved	for UART	0yEE-0yEF
(6)	Reserved	for UART	0yEC-0yED
	Timer T2	T2A/Underflow	0yEA-0yEB
	Timer T2	T2B	0yE8-0yE9
	Reserved	for Future Use	0yE6-0yE7
(7)	Reserved	for Future Use	0yE4-0yE5
	Port L/Wakeup	Port L Edge	0yE2-0yE3
	(9)	Default	VIS Instr. Execution without Any Interrupts
(10) Lowest			

y is VIS page, y ≠ 0.

Interrupts (Continued)

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If $GIE = 1$ and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes $7 t_c$ cycles to execute.

At this time, since $GIE = 0$, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service

routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0-0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 11 shows the COP888CL Interrupt block diagram.

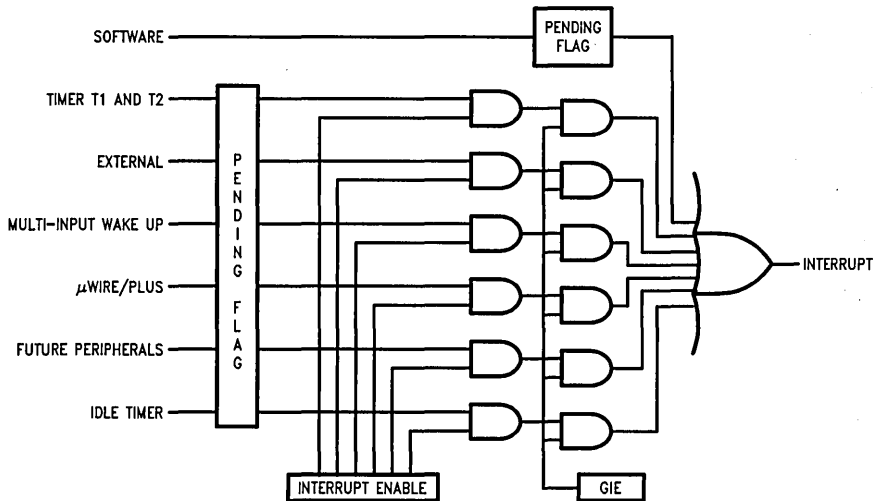


FIGURE 11. COP888CL Interrupt Block Diagram

TL/DD/9766-18

Interrupts (Continued)

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit (**not accessible by the user**) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WATCHDOG

The COP888CL contains a WATCHDOG and clock monitor. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WATCHDOG consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WATCHDOG consists of writing a specific value to a WATCHDOG Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table I shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table II shows the four possible combinations of lower and upper limits for the WATCHDOG service window. This flexibility in choosing the WATCHDOG service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE I. WATCHDOG Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE II. WATCHDOG Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k-8k t_c Cycles
0	1	2k-16k t_c Cycles
1	0	2k-32k t_c Cycles
1	1	2k-64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the COP888CL can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WATCHDOG Operation

The WATCHDOG and Clock Monitor are disabled during reset. The COP888CL comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCHDOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. Table III shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WATCHDOG, the logic will pull the WDOUT (G1) pin low for an additional $16 t_c$ – $32 t_c$ cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CL will stop forcing the WDOUT output low.

The WATCHDOG service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WATCHDOG service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WATCHDOG will time out and WDOUT will enter high impedance state.

WATCHDOG Operation (Continued)**TABLE III. WATCHDOG Service Actions**

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

**TABLE IV. MICROWIRE/PLUS
Master Mode Clock Select**

SL1	SL0	SK
0	0	$2 \times t_c$
0	1	$4 \times t_c$
1	x	$8 \times t_c$

Where t_c is the instruction cycle clock

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following $16 t_c$ – $32 t_c$ clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10$ kHz—No clock rejection.

$1/t_c < 10$ Hz—Guaranteed clock rejection.

WATCHDOG AND CLOCK MONITOR SUMMARY

The following salient points regarding the COP888 WATCHDOG and Clock Monitor should be noted:

- Both WATCHDOG and Clock Monitor detector circuits are inhibited during reset.
- Following reset, the WATCHDOG and Clock Monitor are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and Clock Monitor enable/disable option can only be changed once, during the initial WATCHDOG service following reset.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.
- The Clock Monitor detector circuit is active during both the HALT and IDLE modes. Consequently, the COP888 inadvertently entering the HALT mode will be detected as a Clock Monitor error (provided that the Clock Monitor enable option has been selected by the program).

- With the single-pin R/C oscillator mask option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.
- With the crystal oscillator mask option selected, or with the single-pin R/C oscillator mask option selected and the CLKDLY bit set, the WATCHDOG service window will be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.
- The IDLE timer T0 is not initialized with reset.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the TOPND flag. The TOPND flag is set whenever the thirteenth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the TOPND flag.
- A hardware WATCHDOG service occurs just as the COP888 exits the IDLE mode. Consequently, the Watchdog should not be serviced for at least 2048 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following reset, the initial WATCHDOG service (where the service window and the Clock Monitor enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 2048 instruction cycles without causing a WATCHDOG error.

Detection of Illegal Conditions

The COP888CL can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

Detection of Illegal Conditions (Continued)

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F Hex is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

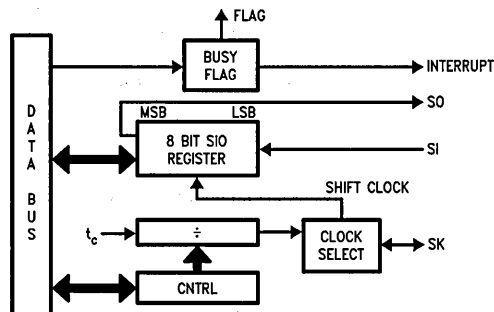
- Executing from undefined ROM
- Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before re-starting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CL to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E²PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 12* shows a block diagram of the MICROWIRE logic.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.



TL/DD/9766-20

FIGURE 12. MICROWIRE/PLUS Block Diagram

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the

master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table IV details the different clock rates that may be selected.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The COP888CL may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 13* shows how two COP888CL microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. The SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock for the SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CL. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table V summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

The COP888CL allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

MICROWIRE/PLUS (Continued)

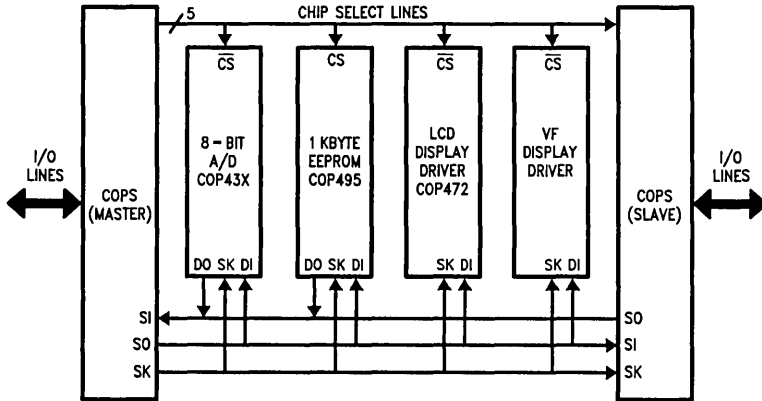


FIGURE 13. MICROWIRE/PLUS Application

TL/DD/9766-21

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE V

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space

Address	Contents
00 to 6F	On-Chip RAM bytes
70 to BF	Unused RAM Address Space
C0	Timer T2 Lower Byte
C1	Timer T2 Upper Byte
C2	Timer T2 Autoload Register T2RA Lower Byte
C3	Timer T2 Autoload Register T2RA Upper Byte
C4	Timer T2 Autoload Register T2RB Lower Byte
C5	Timer T2 Autoload Register T2RB Upper Byte
C6	Timer T2 Control Register
C7	WATCHDOG Service Register (Reg:WDSVR)
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	Reserved
CC	Reserved
CD to CF	Reserved
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved

Reading memory locations 70-7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

The COP888CL has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CL. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the Load B Immediate instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from -31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A ← A + Meml
ADC	A,Meml	ADD with Carry	A ← A + Meml + C, C ← Carry
SUBC	A,Meml	Subtract with Carry	HC ← Half Carry A ← A - Meml + C, C ← Carry
AND	A,Meml	Logical AND	HC ← Half Carry A ← A and Meml
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A ← A or Meml
XOR	A,Meml	Logical EXclusive OR	A ← A xor Meml
IFEQ	MD,Imm	IF Equal	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF Equal	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A ≠ Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg ← Reg - 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A ↔ Mem
X	A,[X]	EXchange A with Memory [X]	A ↔ [X]
LD	A,Meml	LoaD A with Memory	A ← Meml
LD	A,[X]	LoaD A with Memory [X]	A ← [X]
LD	B,Imm	LoaD B with Immed.	B ← Imm
LD	Mem,Imm	LoaD Memory Immed	Mem ← Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg ← Imm
X	A, [B ±]	EXchange A with Memory [B]	A ↔ [B], (B ← B ± 1)
X	A, [X ±]	EXchange A with Memory [X]	A ↔ [X], (X ← X ± 1)
LD	A, [B ±]	LoaD A with Memory [B]	A ← [B], (B ← B ± 1)
LD	A, [X ±]	LoaD A with Memory [X]	A ← [X], (X ← X ± 1)
LD	[B ±],Imm	LoaD Memory [B] Immed.	[B] ← Imm, (B ← ± 1)
CLR	A	CLeaR A	A ← 0
INC	A	INCrement A	A ← A + 1
DEC	A	DECrement A	A ← A - 1
LAI	A	Load A InDirect from ROM	A ← ROM (PU,A)
DCOR	A	Decimal CORect A	A ← BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C ↔ A7 ↔ ... ↔ A0 ↔ C
RLC	A	Rotate A Left thru C	C ← A7 ← ... ← A0 ← C
SWAP	A	SWAP nibbles of A	A7...A4 ↔ A3...A0
SC	A	Set C	C ← 1, HC ← 1
RC	A	Reset C	C ← 0, HC ← 0
IFC	A	IF C	IF C is true, do next instruction
IFNC	A	IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP ← SP + 1, A ← [SP]
PUSH	A	PUSH A onto the stack	[SP] ← A, SP ← SP - 1
VIS		Vector to Interrupt Service Routine	PU ← [VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9...0 ← i (i = 12 bits)
JP	Disp.	Jump relative short	PC ← PC + r (r is -31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] ← PL, [SP-1] ← PU, SP-2, PC ← ii
JSR	Addr	Jump SubRoutine	[SP] ← PL, [SP-1] ← PU, SP-2, PC9...0 ← i
JID		Jump InDirect	PL ← ROM (PU,A)
RET		RETurn from subroutine	SP+2, PL ← [SP], PU ← [SP-1]
RETSK		RETurn and SKip	SP+2, PL ← [SP], PU ← [SP-1]
RETI		RETurn from Interrupt	SP+2, PL ← [SP], PU ← [SP-1], GIE ← 1
INTR		Generate an Interrupt	[SP] ← PL, [SP-1] ← PU, SP-2, PC ← 0FF
NOP		No Operation	PC ← PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1
------	-----

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.	
	[B]	[X]			[B+, B-]	[X+, X-]
X A,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B, Imm				1/1		
LD B, Imm				2/2		
LD Mem, Imm	2/2		3/3		2/2	
LD Reg, Imm			2/3			
IFEQ MD, Imm			3/3			

(IF B < 16)

(IF B > 15)

* = > Memory location addressed by B or X or directly.

COP888CL Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP -15	JP -31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	0
JP -14	JP -30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP -13	JP -29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	2
JP -12	JP -28	LD 0F3, # i	DRSZ 0F3	X A, [X-]	X A,[B-]	IFGT A,#i	IFGT A,[B]	3
JP -11	JP -27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	4
JP -10	JP -26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	5
JP -9	JP -25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	6
JP -8	JP -24	LD 0F7, # i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	7
JP -7	JP -23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	8
JP -6	JP -22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	9
JP -5	JP -21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+], #i	INCA	A
JP -4	JP -20	LD 0FB, # i	DRSZ 0FB	LD A,[X-]	LD A,[B-]	LD [B-], #i	DECA	B
JP -3	JP -19	LD 0FC, # i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	C
JP -2	JP -18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP -1	JP -17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B], #i	RET	E
JP -0	JP -16	LD 0FF, # i	DRSZ 0FF	*	*	LD B,#i	RETI	F

COP888CL Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B, #0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP + 17	INTR	0
IFBIT 1,[B]	*	LD B, #0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP + 18	JP + 2	1
IFBIT 2,[B]	*	LD B, #0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP + 19	JP + 3	2
IFBIT 3,[B]	*	LD B, #0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP + 20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B, #0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP + 21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B, #0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP + 22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B, #09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP + 23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B, #08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP + 24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B, #07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP + 25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B, #06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP + 26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B, #05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP + 27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B, #04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP + 28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B, #03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP + 29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B, #02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP + 30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B, #01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP + 31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B, #00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP + 32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CL mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

- = 1 Crystal Oscillator (CKI/10)
G7 (CK0) is clock generator output to crystal/resonator
CKI is the clock input
- = 2 Single-pin RC controlled oscillator (CKI/10)
G7 is available as a HALT restart and/or general purpose input

OPTION 2: HALT

- = 1 Enable HALT mode
- = 2 Disable HALT mode

OPTION 3: COP888CL BONDING

- = 1 44-Pin PCC
- = 2 40-Pin DIP
- = 3 N.A.
- = 4 28-Pin DIP
- = 5 28-Pin SO

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real-time, full-speed emulation, up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user-selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time

spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC® via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884CL28D5PC	28 DIP	4.5V-5.5V	COP884CL
MHW-884CL28DWPC	28 DIP	2.5V-6.0V	COP884CL
MHW-888CL40D5PC	40 DIP	4.5V-5.5V	COP888CL
MHW-888CL40DWPC	40 DIP	2.5V-6.0V	COP888CL
MHW-888CL44D5PC	44 PLCC	4.5V-5.5V	COP888CL
MHW-888CL44DWPC	44 PLCC	2.5V-6.0V	COP888CL

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM® PC-XT®, PC-AT® or compatible	424410527-001

Development Support (Continued)

SIMULATOR

The COP8 Designer's Tool Kit is available for evaluating National Semiconductor's COP8 microcontroller family. The kit contains programmer's manuals, device datasheets, pocket reference guide, assembler and simulator, which allows the user to write, test, debug and run code on an industry standard compatible PC. The simulator has a windowed user interface and can handle script files that simulate hardware inputs, interrupts and automatic command processing. The capture file feature enables the user to record to a file current cycle and output port changes which are caused by the program under test.

Simulator Ordering Information

Part Number	Description	Manual
COP8-TOOL-KIT	COP8 Designer's Tool Kit Assembler and Simulator	420420270-001 424420269-001

SINGLE-CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAM SUPPORT

Programming of the single-chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single-chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 FAX: (206) 882-104

Duplicate Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for use with Scrambler Boards	COP884CLMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP Socket	COP888CLMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888CLMHEL
COP8-SCRM-SBX	MCM Scrambler Board for 28 LCC Socket	COP884CLMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP884CLMHD, COP888CLMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC Scrambler Board	COP888CLMEL, COP884CLMHD

Single-Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888CLMHEL-X	X = 1: crystal X = 3: R/C	44 LDCC	Multi-Chip Module (MCM), UV Erasable	COP888CL
COP888CLMHD-X	X = 1: crystal X = 3: R/C	40 DIP	MCM, UV Erasable	COP888CL
COP884CLMHD-X	X = 1: crystal X = 3: R/C	28 DIP	MCM, UV Erasable	COP884CL
COP884CLMHEA-X	X = 1: crystal X = 3: R/C	28 LCC	MCM (same footprint as 28 SO), UV Erasable	COP884CL

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System.

Information System

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Set-up: Length: 8-Bit
Parity: None
Stop Bit 1
Operation: 24 Hours, 7 Days

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contents
Dial-A-Helper User Manual P/N
Public Domain Communications Software

Factory Applications Support

Dial-A-Helper also provides immediate factor applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

COP988CF/COP984CF/COP888CF/COP884CF

Single-Chip microCMOS Microcontroller

General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. The COP888CF is a member of this expandable 8-bit core processor family of microcontrollers.

(Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μ s instruction cycle time
- 4096 bytes on-board ROM
- 128 bytes on-board RAM
- Single supply operation: 2.5V–6V
- 8-channel A/D converter with prescaler and both differential and single ended modes
- MICROWIRE/PLUSTM serial I/O
- WATCHDOG™ and Clock Monitor logic
- Ten multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Two Timers (Each with 2 Interrupts)
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - Default VIS
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Two 16-bit timers, each with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PLCC or 40 N
 - 44 PLCC with 37 I/O pins
 - 40 N with 33 I/O pins
 - 28 SO or 28 N, each with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: –40°C to + 85°C
–55°C to + 125°C
- Emulation device—COP888CFMH
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram

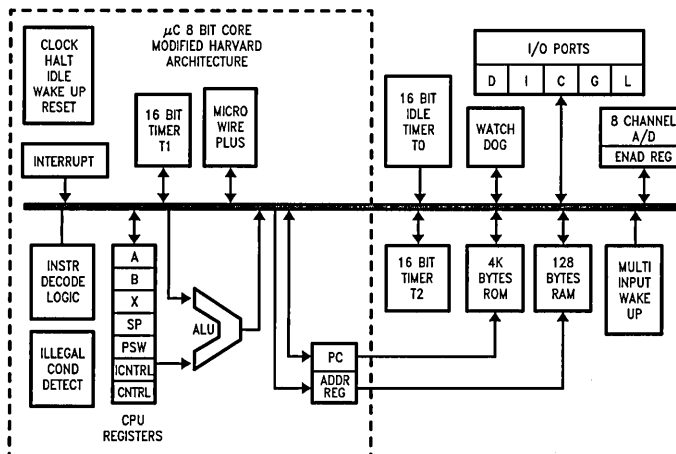


FIGURE 1. COP888CF Block Diagram

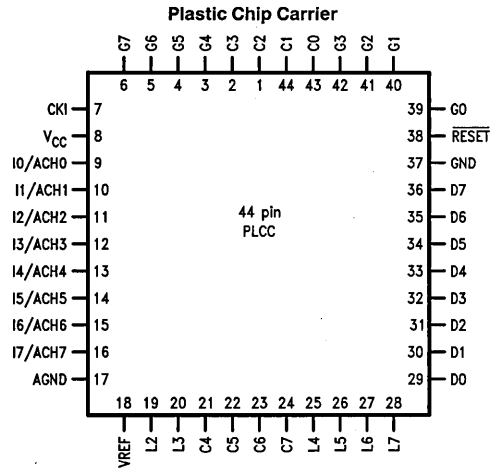
TL/DD/9425-1

General Description (Continued)

It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), an 8-channel, 8-bit A/D converter with both differential and single ended modes, and two power savings modes (HALT and

IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CF operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

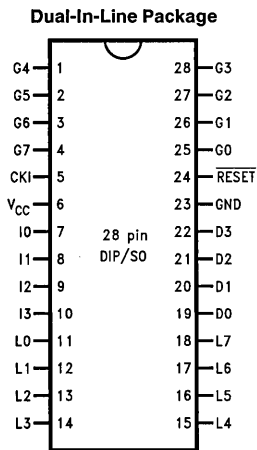
Connection Diagrams



TL/DD/9425-2

Top View

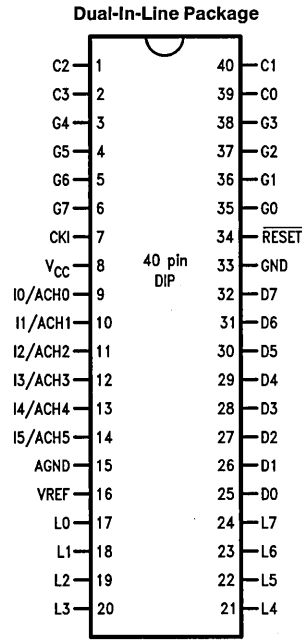
Order Number COP888CF-XXX/V
See NS Plastic Chip Package Number V44A



TL/DD/9425-37

Top View

Order Number COP884CF-XXX/N
or COP884CF-XXX/WM
See NS Package Number D28G or M28B



TL/DD/9425-4

Top View

Order Number COP888F-XXX/N
See NS Molded Package Number N40A

FIGURE 2. COP888CF/COP884CF Connection Diagrams

Connection Diagrams (Continued)

COP888CF Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	—
L1	I/O	MIWU		12	18	—
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
I0	I	ACH0		7	9	9
I1	I	ACH1		8	10	10
I2	I	ACH2			11	11
I3	I	ACH3			12	12
I4	I	ACH4			13	13
I5	I	ACH5			14	14
I6	I	ACH6				15
I7	I	ACH7				16
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
VREF	+VREF			10	16	18
AGND	AGND			9	15	17
Vcc				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 7V
 Voltage at Any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 100 mA

Total Current out of GND Pin (Sink) 110 mA
 Storage Temperature Range -65°C to $+140^{\circ}\text{C}$

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 988CF: $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage 988CF 998CFH		2.5 4.0		4.0 6.0	V V
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1 V_{CC}$	V
Supply Current (Note 2) CKI = 10 MHz CKI = 4 MHz CKI = 4 MHz CKI = 1 MHz	$V_{CC} = 6V, t_c = 1 \mu\text{s}$ $V_{CC} = 6V, t_c = 2.5 \mu\text{s}$ $V_{CC} = 4V, t_c = 2.5 \mu\text{s}$ $V_{CC} = 4V, t_c = 10 \mu\text{s}$			12.5 5.5 2.5 1.4	mA mA mA mA
HALT Current (Note 3)	$V_{CC} = 6V, \text{CKI} = 0 \text{ MHz}$ $V_{CC} = 4.0V, \text{CKI} = 0 \text{ MHz}$		<0.7 <0.3	8 4	μA μA
IDLE Current CKI = 10 MHz CKI = 4 MHz CKI = 1 MHz	$V_{CC} = 6V, t_c = 1 \mu\text{s}$ $V_{CC} = 6V, t_c = 2.5 \mu\text{s}$ $V_{CC} = 4.0V, t_c = 10 \mu\text{s}$			3.5 2.5 0.7	mA mA mA
Input Levels RESET Logic High Logic Low CKI (External and Crystal Osc. Modes) Logic High Logic Low All Other Inputs Logic High Logic Low		$0.8 V_{CC}$ $0.7 V_{CC}$ $0.7 V_{CC}$		 $0.2 V_{CC}$ $0.2 V_{CC}$ $0.2 V_{CC}$	V V V V V V
Hi-Z Input Leakage	$V_{CC} = 6V$	-1		+1	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis				$0.35 V_{CC}$	V
Output Current Levels D Outputs Source Sink All Others Source (Weak Pull-Up Mode) Source (Push-Pull Mode) Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$ $V_{CC} = 4V, V_{OL} = 1V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$ $V_{CC} = 4V, V_{OH} = 2.7V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$ $V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$ $V_{CC} = 4V, V_{OL} = 0.4V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	0.4 0.2 10 2.0 10 2.5 0.4 0.2 1.6 0.7		100 33	mA mA mA mA μA μA mA mA mA mA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The A/D is disabled. V_{REF} is tied to AGND (effectively shorting the Reference resistor). The clock monitor is disabled.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
TRI-STATE Leakage	$V_{CC} = 6.0\text{V}$	-1		+1	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications $V_{CC} = 5\text{V} \pm 10\% (V_{SS} - 0.050\text{V}) \leq \text{Any Input} \leq (V_{CC} + 0.050\text{V})$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			± 1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the Best Straight Line			$\pm 1/2$	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			$\pm 1/2$	LSB
Input Reference Resistance		1.6		4.8	$k\Omega$
Common Mode Input Range (Note 7)		AGND		V_{REF}	V
DC Common Mode Error				$\pm 1/4$	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
A/D Clock Frequency (Note 5)		0.1		1.67	MHz
Conversion Time (Note 4)			12		A/D Clock Cycles

Note 4: Conversion Time includes sample and hold time.

Note 5: See Prescaler description.

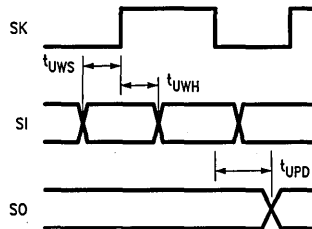
Note 6: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Note 7: For $V_{IN(-)} \geq V_{IN(+)}$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)	Crystal, Resonator	$4\text{V} \leq V_{CC} \leq 6\text{V}$	1		μs
		$2.5\text{V} \leq V_{CC} < 4\text{V}$	2.5	DC	μs
	R/C Oscillator	$4\text{V} \leq V_{CC} \leq 6\text{V}$	3	DC	μs
		$2.5\text{V} \leq V_{CC} < 4\text{V}$	7.5	DC	μs
CKI Clock Duty Cycle (Note 8)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 8)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Fall Time (Note 8)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Inputs	t_{SETUP}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	200		ns
		$2.5\text{V} \leq V_{CC} < 4\text{V}$	500		ns
	t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	60		ns
		$2.5\text{V} \leq V_{CC} < 4\text{V}$	150		ns
Output Propagation Delay	$t_{\text{PD1}}, t_{\text{PD0}}$ SO, SK	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$			
		$4\text{V} \leq V_{CC} \leq 6\text{V}$		0.7	μs
		$2.5\text{V} \leq V_{CC} < 4\text{V}$		1.75	μs
	All Others	$4\text{V} \leq V_{CC} \leq 6\text{V}$		1	μs
		$2.5\text{V} \leq V_{CC} < 4\text{V}$		2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width	Interrupt Input High Time		1		t_c
	Interrupt Input Low Time		1		t_c
	Timer Input High Time		1		t_c
	Timer Input Low Time		1		t_c
	Reset Pulse Width		1		

Note 8: Parameter sample (not 100% tested).



TL/DD/9425-26

FIGURE 3. MICROWIRE/PLUS Timing

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 888CF: -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			5.5	mA
CKI = 4 MHz	$V_{CC} = 4V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 4V, t_c = 10 \mu s$			1.4	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 \text{ MHz}$ $V_{CC} = 4V, CKI = 0 \text{ MHz}$		<1 <0.5	10 6	μA μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 4V, t_c = 10 \mu s$			0.7	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$	10			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	2.0			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$	10		100	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage	$V_{CC} = 6.0V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The A/D is disabled. V_{REF} is tied to AGND (effectively shorting the Reference resistor). The clock monitor is disabled.

DC Electrical Characteristics 888CF: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				15 3	mA mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications 888CF: $V_{CC} = 5V \pm 10\%$ ($V_{SS} - 0.050V$) \leq Any Input \leq ($V_{CC} + 0.050V$)

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			± 1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the Best Straight Line			$\pm 1/2$	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			$\pm 1/2$	LSB
Input Reference Resistance		1.6		4.8	k Ω
Common Mode Input Range (Note 7)		AGND		V_{REF}	V
DC Common Mode Error				$\pm 1/4$	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
A/D Clock Frequency (Note 5)		0.1		1.67	MHz
Conversion Time (Note 4)			12		A/D Clock Cycles

Note 4: Conversion Time includes sample and hold time.**Note 5:** See Prescaler description.**Note 6:** Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.**Note 7:** For $V_{IN(-)} \geq V_{IN(+)}$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

AC Electrical Characteristics 888CF: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator	$4\text{V} \leq V_{CC} \leq 6\text{V}$	1		DC	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	2.5		DC	μs
	$4\text{V} \leq V_{CC} \leq 6\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	7.5		DC	μs
R/C Oscillator	$f_r = \text{Max}$	40		60	%
	$f_r = 10\text{ MHz Ext Clock}$			5	ns
	$f_r = 10\text{ MHz Ext Clock}$			5	ns
CKI Clock Duty Cycle (Note 8) Rise Time (Note 8) Fall Time (Note 8)	$f_r = \text{Max}$	40		60	%
	$f_r = 10\text{ MHz Ext Clock}$			5	ns
	$f_r = 10\text{ MHz Ext Clock}$			5	ns
	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Inputs t_{SETUP} t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	200			ns
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	500			ns
	$4\text{V} \leq V_{CC} \leq 6\text{V}$	60			ns
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	150			ns
Output Propagation Delay $t_{\text{PD1}}, t_{\text{PD0}}$ SO, SK All Others	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$				
	$4\text{V} \leq V_{CC} \leq 6\text{V}$			0.7	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$			1.75	μs
	$4\text{V} \leq V_{CC} \leq 6\text{V}$			1	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
	Interrupt Input High Time	1			t_c
	Interrupt Input Low Time	1			t_c
	Timer Input High Time	1			t_c
	Timer Input Low Time	1			t_c
Reset Pulse Width		1			μs

Note 8: Parameter sample (not 100% tested).

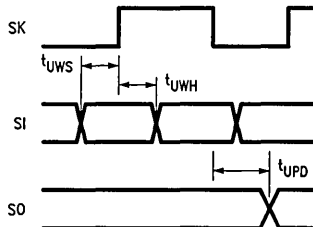
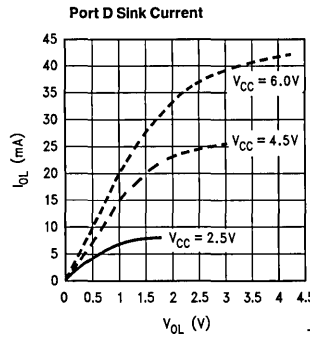
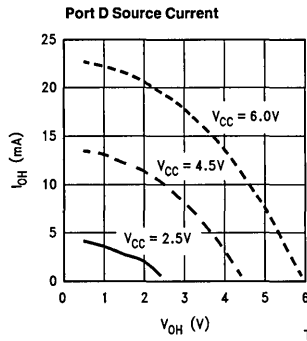
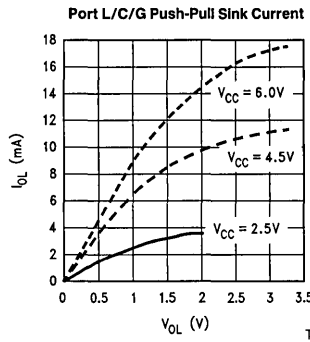
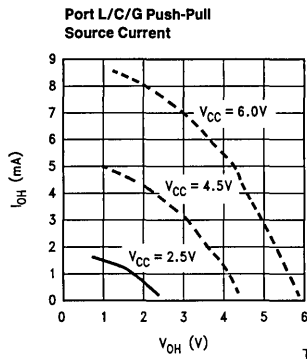
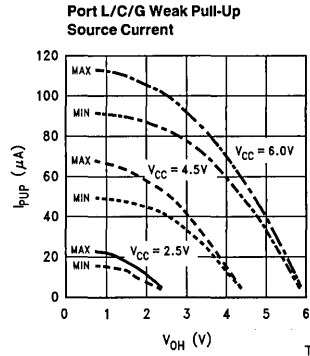
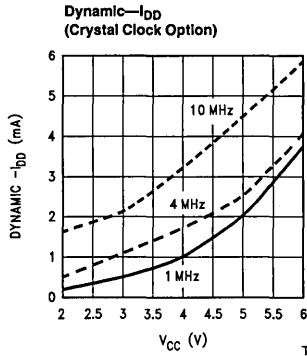
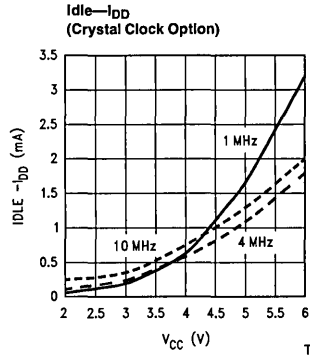
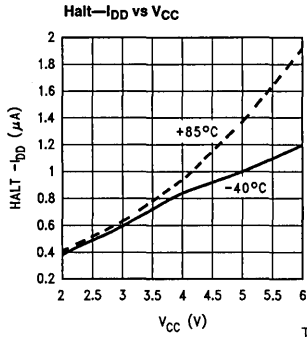


FIGURE 3. MICROWIRE/PLUS Timing

TL/DD/9425-26

Typical Performance Characteristics



Pin Descriptions

V_{CC} and GND are the power supply pins.

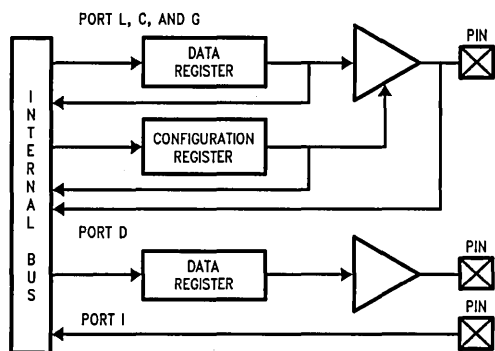
V_{REF} and AGND are the reference voltage pins for the on-board A/D converter.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

\overline{RESET} is the master reset input. See Reset Description section.

The COP888CF contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports G and L), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CF memory map for the various addresses associated with the I/O ports.) Figure 4 shows the I/O port configurations for the COP888CF. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/9425-6

FIGURE 4. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wakeup (MIWU) on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B. L0 and L1 are not available on the 44-pin version of the COP888CF, since they are replaced by V_{REF} and AGND. L0 and L1 are not terminated on the 44-pin version. Consequently, reading L0 or L1 as inputs will return unreliable data with the 44-pin package, so this data should be masked out with user software when the L port is read for input data. It is recommended that the pins be configured as outputs.

Port L has the following alternate features:

L0	MIWU
L1	MIWU
L2	MIWU
L3	MIWU
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU
L7	MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin, but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin or general purpose input (R/C clock configuration), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0	INTR (External Interrupt Input)
G2	T1B (Timer T1 Capture Input)
G3	T1A (Timer T1 I/O)
G4	SO (MICROWIRE Serial Data Output)
G5	SK (MICROWIRE Serial Clock)
G6	SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1	WDOOUT WatchDog and/or Clock Monitor dedicated output
G7	CKO Oscillator dedicated output or general purpose input

Port C is an 8-bit I/O port. The 40-pin device does not have a full complement of Port C pins. The unavailable pins are not terminated. A read operation for these unterminated pins will return unpredictable values.

Pin Descriptions (Continued)

Port I is an 8-bit Hi-Z input port, and also provides the analog inputs to the A/D converter. The 28-pin device does not have a full complement of Port I pins. The unavailable pins are not terminated (i.e. they are floating). A read operation from these unterminated pins will return unpredictable values. The user should ensure that the software takes this into account by either masking out these inputs, or else restricting the accesses to bit operations only. If unterminated, Port I pins will draw power only when addressed.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CF is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CF architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_c) cycle time.

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CF consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CF vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The COP888CF has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

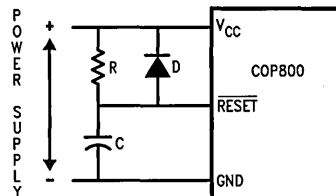
The instruction set of the COP888CF permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CF (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for Ports L, G, and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is initialized high with RESET. The PC, PSW, CNTRL, ICNTRL, and T2CNTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The A/D control register ENAD is cleared, resulting in the ADC being powered down initially. The Stack Pointer, SP, is initialized to 06F Hex.

The COP888CF comes out of reset with both the WatchDog logic and the Clock Monitor detector armed, and with both the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor detector circuits are inhibited during reset. The WatchDog service window bits are initialized to the maximum WatchDog service window of 64k t_c clock cycles. The Clock Monitor bit is initialized high, and will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16–32 t_c clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in Figure 5 should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes.



TL/DD/9425-7

$RC > 5 \times$ Power Supply Rise Time

FIGURE 5. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock (1/t_c).

Figure 6 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.

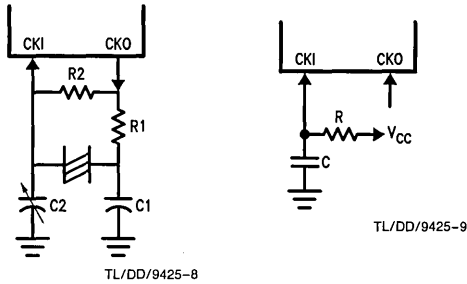


FIGURE 6. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	10	V _{CC} = 5V
0	1	30	30-36	4	V _{CC} = 5V
0	1	200	100-150	0.455	V _{CC} = 5V

TABLE B. R/C Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	V _{CC} = 5V
5.6	100	1.1 to 1.3	7.4 to 9.0	V _{CC} = 5V
6.8	100	0.9 to 1.1	8.8 to 10.8	V _{CC} = 5V

Note: 3k ≤ R ≤ 200k
50 pF ≤ C ≤ 200 pF

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5

6. DC reference current contribution from the A/D converter—I6
7. Clock Monitor current when enabled—I7

Thus the total current drain, I_t, is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
- T1C0 Timer T1 Start/Stop control in timer modes 1 and 2
Timer T1 Underflow Interrupt Pending Flag in timer mode 3
- T1C1 Timer T1 mode control bit
- T1C2 Timer T1 mode control bit
- T1C3 Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7				Bit 0			

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable (enables interrupts)
- EXEN Enable external interrupt
- BUSY MICROWIRE/PLUS busy shifting flag
- EXPND External interrupt pending
- T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
- T1PND A Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)

- C Carry Flag
- HC Half Carry Flag

HC	C	T1PND A	T1ENA	EXPND	BUSY	EXEN	GIE
Bit 7				Bit 0			

Control Registers (Continued)

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB	Timer T1 Interrupt Enable for T1B Input capture edge
T1PNDB	Timer T1 Interrupt Pending Flag for T1B capture edge
μ WEN	Enable MICROWIRE/PLUS interrupt
μ WPND	MICROWIRE/PLUS interrupt pending
T0EN	Timer T0 Interrupt Enable (Bit 12 toggle)
T0PND	Timer T0 Interrupt pending
LPEN	L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μ WPND	μ WEN	T1PNDB	T1ENB
Bit 7							Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB	Timer T2 Interrupt Enable for T2B Input capture edge
T2PNDB	Timer T2 Interrupt Pending Flag for T2B capture edge
T2ENA	Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
T2PND	Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
T2C0	Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3
T2C1	Timer T2 mode control bit
T2C2	Timer T2 mode control bit
T2C3	Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PND	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

Timers

The COP888CF contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

Figure 7 shows a block diagram for the timers on the COP888CF.

TIMER T0 (IDLE TIMER)

The COP888CF supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c . The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

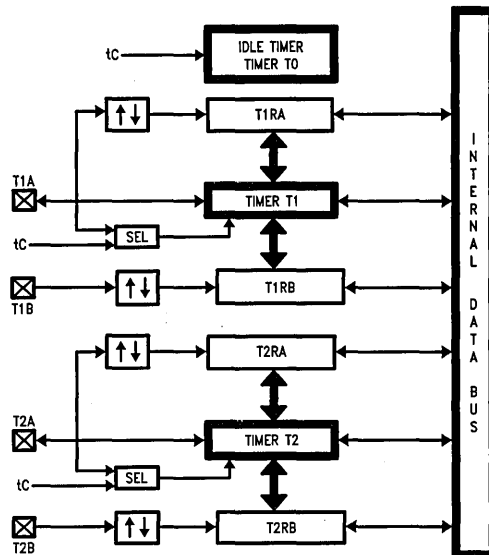
- Exit out of the Idle Mode (See Idle Mode description)
- WatchDog logic (See WatchDog description)
- Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the TOPND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_c = 1 \mu\text{s}$). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1 AND TIMER T2

The COP888CF has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2, are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CF to



TL/DD/9425-11

FIGURE 7. Timers for the COP888CF

easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Timers (Continued)

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CF to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_c . Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode. The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPND A and TxPND B. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

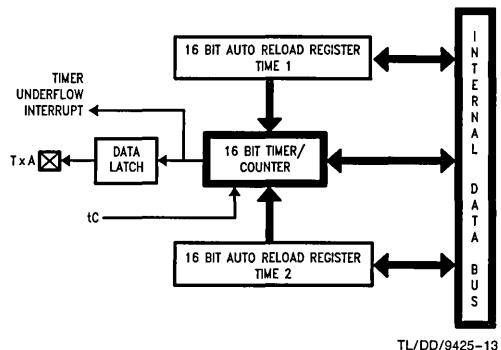


FIGURE 8. Timer in PWM Mode

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the

timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPND A pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPND B flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

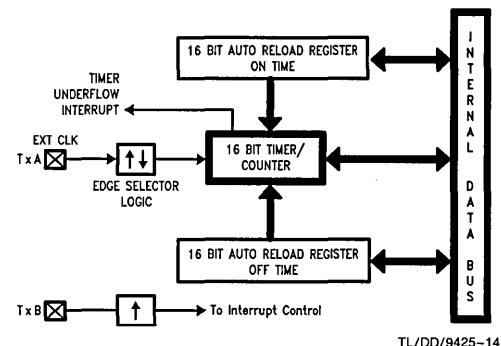


FIGURE 9. Timer in External Event Counter Mode

Mode 3. Input Capture Mode

The COP888CF can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_c rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

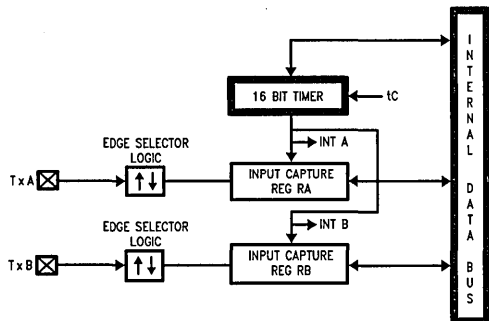
The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPND A and TxPND B. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxCO pending flag (the TxCO control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxCO control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both

Timers (Continued)

whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.



TL/DD/9425-15

FIGURE 10. Timer in Input Capture Mode

TIMER CONTROL FLAGS

The timers T1 and T2 have identical control structures. The control bits and their functions are summarized below.

- TxC0 Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop
Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
- TxPND A Timer Interrupt Pending Flag
- TxPND B Timer Interrupt Pending Flag
- TxENA Timer Interrupt Enable Flag
- TxENB Timer Interrupt Enable Flag
1 = Timer Interrupt Enabled
0 = Timer Interrupt Disabled
- TxC3 Timer mode control
- TxC2 Timer mode control
- TxC1 Timer mode control

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c

Power Save Modes

The COP888CF offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CF is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock, timers, and A/D converter, are stopped. The WatchDog logic on the COP888CF is disabled during the HALT mode. However, the clock monitor circuitry if enabled remains active and will cause the WatchDog output pin (WDOUT) to go low. If the HALT mode is used and the user does not want to activate the WDOUT pin, the Clock Monitor should be disabled after the device comes out of reset (resetting the Clock Monitor control bit with the first write to the WDSVR register). In the HALT mode, the power requirements of the COP888CF are minimal and the applied voltage (V_{CC}) may be decreased to V_r ($V_r = 2.0V$) without altering the state of the machine.

The COP888CF supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

The COP888CF has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CF will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CF cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the COP888CF inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CF is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the WatchDog logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CF can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_c = 1 \mu s$) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the TOPND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the TOPND bit gets set, the COP888CF will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CF will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CF from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 11 shows the Multi-Input Wakeup logic for the COP888CF microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CF to exit the HALT or IDLE modes. The selection is made via the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```

RBIT 5, WKEN
SBIT 5, WKEDG
RBIT 5, WKPND
SBIT 5, WKEN
  
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CF will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

The WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

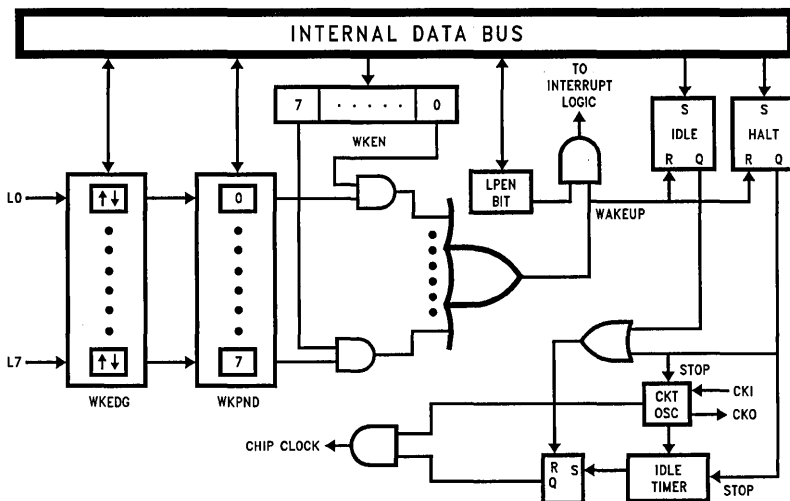


FIGURE 11. Multi-Input Wake Up Logic

TL/DD/9425-16

Multi-Input Wakeup (Continued)

The GIE (global interrupt enable) bit enables the interrupt function. A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CF out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CF will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CF will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CF to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

A/D Converter

The COP888CF contains an 8-channel, multiplexed input, successive approximation, A/D converter. Two dedicated pins, V_{REF} and AGND are provided for voltage reference.

OPERATING MODES

The A/D converter supports ratiometric measurements. It supports both Single Ended and Differential modes of operation.

Four specific analog channel selection modes are supported. These are as follows:

Allow any specific channel to be selected at one time. The A/D converter performs the specific conversion requested and stops.

Allow any specific channel to be scanned continuously. In other words, the user will specify the channel and the A/D converter will keep on scanning it continuously. The user can come in at any arbitrary time and immediately read the result of the last conversion. The user does not have to wait for the current conversion to be completed.

Allow any differential channel pair to be selected at one time. The A/D converter performs the specific differential conversion requested and stops.

Allow any differential channel pair to be scanned continuously. In other words, the user will specify the differential channel pair and the A/D converter will keep on scanning it continuously. The user can come in at any arbitrary time and immediately read the result of the last differential conversion. The user does not have to wait for the current conversion to be completed.

The A/D converter is supported by two memory mapped registers, the result register and the mode control register. When the COP888CF is reset, the control register is cleared and the A/D is powered down. The A/D result register has unknown data following reset.

A/D Control Register

A control register, Reg: ENAD, contains 3 bits for channel selection, 3 bits for prescaler selection, and 2 bits for mode selection. An A/D conversion is initiated by writing to the ENAD control register. The result of the conversion is available to the user from the A/D result register, Reg: ADRSLT.

Reg: ENAD

CHANNEL SELECT	MODE SELECT	PRESCALER SELECT
Bits 7, 6, 5	Bits 4,3	Bits 2, 1, 0

CHANNEL SELECT

This 3-bit field selects one of eight channels to be the V_{IN+}. The mode selection determines the V_{IN-} input.

Single Ended mode:

Bit 7	Bit 6	Bit 5	Channel No.
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Differential mode:

Bit 7	Bit 6	Bit 5	Channel Pairs (+, -)
0	0	0	0, 1
0	0	1	1, 0
0	1	0	2, 3
0	1	1	3, 2
1	0	0	4, 5
1	0	1	5, 4
1	1	0	6, 7
1	1	1	7, 6

MODE SELECT

This 2-bit field is used to select the mode of operation (single conversion, continuous conversions, differential, single ended) as shown in the following table.

Bit 4	Bit 3	Mode
0	0	Single Ended mode, single conversion
0	1	Single Ended mode, continuous scan of a single channel into the result register
1	0	Differential mode, single conversion
1	1	Differential mode, continuous scan of a channel pair into the result register

A/D Converter (Continued)

PRESCALER SELECT

This 3-bit field is used to select one of the seven prescaler clocks for the A/D converter. The prescaler also allows the A/D clock inhibit power saving mode to be selected. The following table shows the various prescaler options.

Bit 2	Bit 1	Bit 0	Clock Select
0	0	0	Inhibit A/D clock
0	0	1	Divide by 1
0	1	0	Divide by 2
0	1	1	Divide by 4
1	0	0	Divide by 6
1	0	1	Divide by 12
1	1	0	Divide by 8
1	1	1	Divide by 16

ADC Operation

The A/D converter interface works as follows. Writing to the A/D control register ENAD initiates an A/D conversion unless the prescaler value is set to 0, in which case the ADC clock is stopped and the ADC is powered down. The conversion sequence starts at the beginning of the write to ENAD operation powering up the ADC. At the first falling edge of the converter clock following the write operation (not counting the falling edge if it occurs at the same time as the write operation ends), the sample signal turns on for two clock cycles. The ADC is selected in the middle of the sample period. If the ADC is in single conversion mode, the conversion complete signal from the ADC will generate a power down for the A/D converter. If the ADC is in continuous mode, the conversion complete signal will restart the conversion sequence by deselecting the ADC for one converter clock cycle before starting the next sample. The ADC 8-bit result is loaded into the A/D result register (ADRSLT) except during LOAD clock high, which prevents transient data (resulting from the ADC writing a new result over an old one) being read from ADRSLT.

PRESCALER

The COP888CF A/D Converter (ADC) contains a prescaler option which allows seven different clock selections. The A/D clock frequency is equal to CKI divided by the prescaler value. Note that the prescaler value must be chosen such that the A/D clock falls within the specified range. The maximum A/D frequency is 1.67 MHz. This equates to a 600 ns ADC clock cycle.

The A/D converter takes 12 ADC clock cycles to complete a conversion. Thus the minimum ADC conversion time for the COP888CF is $7.2 \mu\text{s}$ when a prescaler of 6 has been selected. These 12 ADC clock cycles necessary for a conversion consist of 1 cycle at the beginning for reset, 2 cycles for sampling, 8 cycles for converting, and 1 cycle for loading the result into the COP888CF A/D result register (ADRSLT). This A/D result register is a read-only register. The COP888CF cannot write into ADRSLT.

The prescaler also allows an A/D clock inhibit option, which saves power by powering down the A/D when it is not in use.

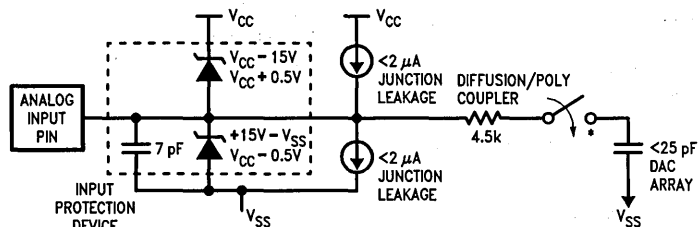
Note: The A/D converter is also powered down when the COP888CF is in either the HALT or IDLE modes. If the ADC is running when the COP888CF enters the HALT or IDLE modes, the ADC will power down during the HALT or IDLE, and then will reinitialize the conversion when the COP888CF comes out of the HALT or IDLE modes.

Analog Input and Source Resistance Considerations

Figure 12 shows the A/D pin model for the COP888CF in single ended mode. The differential mode has similar A/D pin model. The leads to the analog inputs should be kept as short as possible. Both noise and digital clock coupling to an A/D input can cause conversion errors. The clock lead should be kept away from the analog input line to reduce coupling. The A/D channel input pins do not have any internal output driver circuitry connected to them because this circuitry would load the analog input signals due to output buffer leakage current.

Source impedances greater than $1 \text{ k}\Omega$ on the analog input lines will adversely affect internal RC charging time during input sampling. As shown in Figure 12, the analog switch to the DAC array is closed only during the 2 A/D cycle sample time. Large source impedances on the analog inputs may result in the DAC array not being charged to the correct voltage levels, causing scale errors.

If large source resistance is necessary, the recommended solution is to slow down the A/D clock speed in proportion to the source resistance. The A/D converter may be operated at the maximum speed for R_S less than $1 \text{ k}\Omega$. For R_S greater than $1 \text{ k}\Omega$, A/D clock speed needs to be reduced. For example, with $R_S = 2 \text{ k}\Omega$, the A/D converter may be operated at half the maximum speed. A/D converter clock speed may be slowed down by either increasing the A/D prescaler divide-by or decreasing the CKI clock frequency. The A/D clock speed may be reduced to its minimum frequency of 100 kHz.



*The analog switch is closed only during the sample time.

FIGURE 12. A/D Pin Model (Single Ended Mode)

TL/DD/9425-28

Interrupts

The COP888CF supports a vectored interrupt scheme. It supports a total of ten interrupt sources. The following table lists all the possible COP888CF interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If $GIE = 1$ and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes $7 t_c$ cycles to execute.

At this time, since $GIE = 0$, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	Underflow	0yF8–0yF9
(4)	Timer T1	T1A/Underflow	0yF6–0yF7
	Timer T1	T1B	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
	Reserved	for UART	0yEE–0yEF
	Reserved	for UART	0yEC–0yED
(7)	Timer T2	T2A/Underflow	0yEA–0yEB
(8)	Timer T2	T2B	0yE8–0yE9
	Reserved	for Future Use	0yE6–0yE7
	Reserved	for Future Use	0yE4–0yE5
(9)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(10) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page, y ≠ 0

Interrupts (Continued)

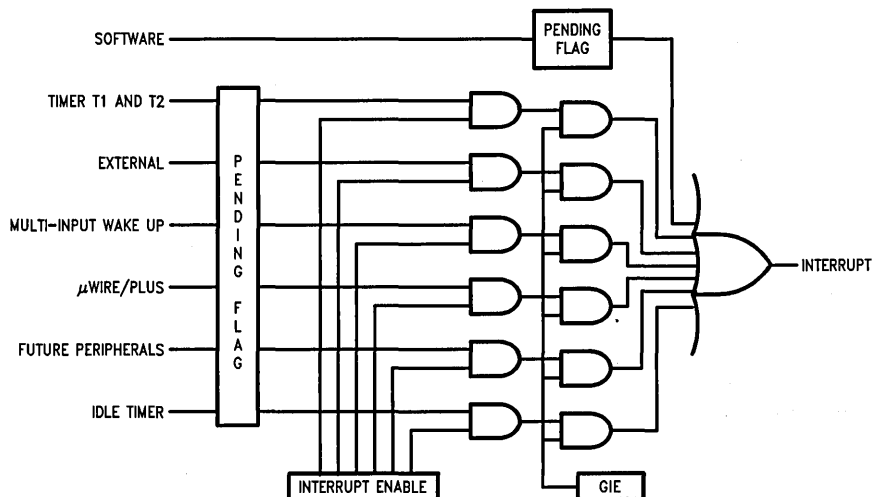


FIGURE 13. COP888CF Interrupt Block Diagram

TL/DD/9425-18

maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0-0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 13 shows the COP888CF Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to RESET, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit (**not accessible by the user**) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WatchDog

The COP888CF contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to

detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table I shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table II shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE I. WatchDog Service Register

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE II. WatchDog Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k-8k t_c Cycles
0	1	2k-16k t_c Cycles
1	0	2k-32k t_c Cycles
1	1	2k-64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the COP888CF can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WATCHDOG Operation

The WATCHDOG and Clock Monitor are disabled during reset. The COP888CF comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCHDOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. Table III shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOOUT pin, on pin 1 of the port G. WDOOUT is active low. The WDOOUT pin is in the high impedance state in the inactive state. Upon triggering the WATCHDOG, the logic will pull the WDOOUT (G1) pin low for an additional $16 t_c$ – $32 t_c$ cycles after the signal level on WDOOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CF will stop forcing the WDOOUT output low.

The WATCHDOG service window will restart when the WDOOUT pin goes high. It is recommended that the user tie the WDOOUT pin back to V_{CC} through a resistor in order to pull WDOOUT high.

A WATCHDOG service while the WDOOUT signal is active will be ignored. The state of the WDOOUT pin is not guaranteed on reset, but if it powers up low then the WATCHDOG will time out and WDOOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following $16 t_c$ – $32 t_c$ clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10$ kHz—No clock rejection.

$1/t_c < 10$ Hz—Guaranteed clock rejection.

WATCHDOG AND CLOCK MONITOR SUMMARY

The following salient points regarding the COP888 WATCHDOG and CLOCK MONITOR should be noted:

- Both the WATCHDOG and Clock Monitor detector circuits are inhibited during RESET.
- Following RESET, the WATCHDOG and CLOCK MONITOR are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and Clock Monitor enable/disable option can only be changed once, during the initial WATCHDOG service following RESET.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.
- The Clock Monitor detector circuit is active during both the HALT and IDLE modes. Consequently, the COP888 inadvertently entering the HALT mode will be detected as a Clock Monitor error (provided that the Clock Monitor enable option has been selected by the program).
- With the single-pin R/C oscillator mask option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.

TABLE III. WATCHDOG Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

WATCHDOG Operation (Continued)

- With the crystal oscillator mask option selected, or with the single-pin R/C oscillator mask option selected and the CLKDLY bit set, the WATCHDOG service window will be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.
- The IDLE timer T0 is not initialized with RESET.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the T0PND flag. The T0PND flag is set whenever the thirteenth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the T0PND flag.
- A hardware WATCHDOG service occurs just as the COP888 exits the IDLE mode. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following RESET, the initial WATCHDOG service (where the service window and the CLOCK MONITOR enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 2048 instruction cycles without causing a WATCHDOG error.

Detection of Illegal Conditions

The COP888CF can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F Hex is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

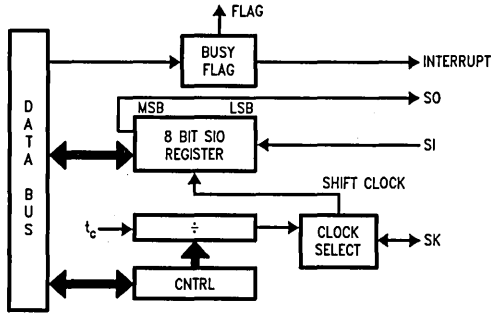
Thus, the chip can detect the following illegal conditions:

- Executing from undefined ROM
- Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures).

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CF to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E²PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 14 shows a block diagram of the MICROWIRE/PLUS logic.



TL/DD/9425-20

FIGURE 14. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. TABLE IV details the different clock rates that may be selected.

TABLE IV. MICROWIRE/PLUS Master Mode Clock Selection

SL1	SL0	SK
0	0	$2 \times t_c$
0	1	$4 \times t_c$
1	x	$8 \times t_c$

Where t_c is the instruction cycle clock

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The COP888CF may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. Figure 15 shows how two COP888CF microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock for the SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CF. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table V summarizes the bit settings required for Master mode of operation.

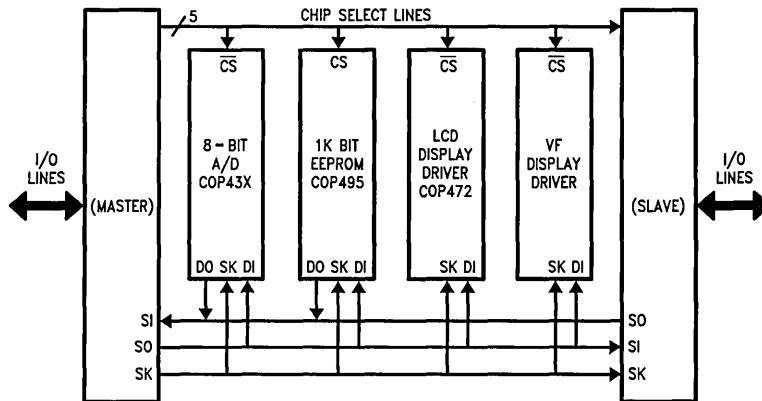


FIGURE 15. MICROWIRE/PLUS Application

TL/DD/9425-21

MICROWIRE/PLUS (Continued)**MICROWIRE/PLUS Slave Mode Operation**

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

The COP888CF allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase mode the SIO register is shifted on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE V

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space

Address	Contents
00 to 6F	On-Chip RAM bytes
70 to BF	Unused RAM Address Space
C0	Timer T2 Lower Byte
C1	Timer T2 Upper Byte
C2	Timer T2 Autoload Register T2RA Lower Byte
C3	Timer T2 Autoload Register T2RA Upper Byte
C4	Timer T2 Autoload Register T2RB Lower Byte
C5	Timer T2 Autoload Register T2RB Upper Byte
C6	Timer T2 Control Register
C7	WATCHDOG Service Register (Reg:WDSVR)
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	A/D Converter Control Register (Reg:ENAD)
CC	A/D Converter Result Register (Reg:ADRSLT)
CD to CF	Reserved
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved

Reading memory locations 70-7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

The COP888CF has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CF. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the Load B Immediate instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from -31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
MemI	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A ← A + Meml
ADC	A,Meml	ADD with Carry	A ← A + Meml + C, C ← Carry
			HC ← Half Carry
SUBC	A,Meml	Subtract with Carry	A ← A Meml + C, C ← Carry
			HC ← Half Carry
AND	A,Meml	Logical AND	A ← A and Meml
ANDSZ	A,Imml	Logical AND Immed., Skip if Zero	Skip next if (A and Imml) = 0
OR	A,Meml	Logical OR	A ← A or Meml
XOR	A,Meml	Logical EXclusive OR	A ← A xor Meml
IFEQ	MD,Imml	IF Equal	Compare MD and Imml, Do next if MD = Imml
IFEQ	A,Meml	IF Equal	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A ≠ Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	IF B Not Equal	Do next if lower 4 bits of B ≠ Imml
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg ← Reg - 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A ↔ Mem
X	A,[X]	EXchange A with Memory [X]	A ↔ [X]
LD	A,Meml	LoaD A with Memory	A ← Meml
LD	A,[X]	LoaD A with Memory [X]	A ← [X]
LD	B,Imml	LoaD B with Immed.	B ← Imml
LD	Mem,Imml	LoaD Memory Immed	Mem ← Imml
LD	Reg,Imml	LoaD Register Memory Immed.	Reg ← Imml
X	A, [B ±]	EXchange A with Memory [B]	A ↔ [B], (B ← B ± 1)
X	A, [X ±]	EXchange A with Memory [X]	A ↔ [X], (X ← X ± 1)
LD	A, [B ±]	LoaD A with Memory [B]	A ← [B], (B ← B ± 1)
LD	A, [X ±]	LoaD A with Memory [X]	A ← [X], (X ← X ± 1)
LD	[B ±], Imml	LoaD Memory [B] Immed.	[B] ← Imml, (B ← B ± 1)
CLR	A	CLear A	A ← 0
INC	A	INCrement A	A ← A + 1
DEC	A	DECrement A	A ← A - 1
LAID		Load A InDirect from ROM	A ← ROM (PU,A)
DCOR	A	Decimal CORrect A	A ← BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C → A7 → ... → A0 → C
RLC	A	Rotate A Left thru C	C ← A7 ← ... ← A0 ← C
SWAP	A	SWAP nibbles of A	A7...A4 ↔ A3...A0
SC		Set C	C ← 1, HC ← 1
RC		Reset C	C ← 0, HC ← 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP ← SP + 1, A ← [SP]
PUSH	A	PUSH A onto the stack	[SP] ← A, SP ← SP - 1
VIS		Vector to Interrupt Service Routine	PU ← [VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9...0 ← i (i = 12 bits)
JP	Disp.	Jump relative short	PC ← PC + r (r is -31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] ← PL, [SP-1] ← PU, SP-2, PC ← ii
JSR	Addr	Jump SubRoutine	[SP] ← PL, [SP-1] ← PU, SP-2, PC9...0 ← i
JID		Jump InDirect	PL ← ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL ← [SP], PU ← [SP-1]
RETSK		RETurn and SKip	SP + 2, PL ← [SP], PU ← [SP-1]
RETI		RETurn from Interrupt	SP + 2, PL ← [SP], PU ← [SP-1], GIE ← 1
INTR		Generate an Interrupt	[SP] ← PL, [SP-1] ← PU, SP-2, PC ← OFF
NOP		No Operation	PC ← PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1
------	-----

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAI	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.	
	[B]	[X]			[B+, B-]	[X+, X-]
XA,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B, Imm				1/1		
LD B, Imm				2/2		
LD Mem, Imm	2/2		3/3		2/2	
LD Reg, Imm			2/3			
IFEQ MD, Imm			3/3			

(IF B < 16)
(IF B > 15)

* = > Memory location addressed by B or X or directly.

COP888CF Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP -15	JP -31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A,[B]	0
JP -14	JP -30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP -13	JP -29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A, #i	IFEQ A,[B]	2
JP -12	JP -28	LD 0F3, # i	DRSZ 0F3	X A, [X-]	X A,[B-]	IFGT A, #i	IFGT A,[B]	3
JP -11	JP -27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A, #i	ADD A,[B]	4
JP -10	JP -26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A, #i	AND A,[B]	5
JP -9	JP -25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A, #i	XOR A,[B]	6
JP -8	JP -24	LD 0F7, # i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	7
JP -7	JP -23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A, #i	IFC	8
JP -6	JP -22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md, #i	IFNE A, #i	IFNC	9
JP -5	JP -21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+], #i	INCA	A
JP -4	JP -20	LD 0FB, # i	DRSZ 0FB	LD A,[X-]	LD A,[B-]	LD [B-], #i	DECA	B
JP -3	JP -19	LD 0FC, # i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	POPA	C
JP -2	JP -18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	D
JP -1	JP -17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B], #i	RET	E
JP -0	JP -16	LD 0FF, # i	DRSZ 0FF	*	*	LD B, #i	RETI	F

COP888CF Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B, #0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP + 17	INTR	0
IFBIT 1,[B]	*	LD B, #0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP + 18	JP + 2	1
IFBIT 2,[B]	*	LD B, #0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP + 19	JP + 3	2
IFBIT 3,[B]	*	LD B, #0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP + 20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B, #0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP + 21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B, #0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP + 22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B, #09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP + 23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B, #08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP + 24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B, #07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP + 25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B, #06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP + 26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B, #05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP + 27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B, #04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP + 28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B, #03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP + 29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B, #02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP + 30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B, #01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP + 31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B, #00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP + 32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CF mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

- = 1 Crystal Oscillator (CKI/10)
 - G7 (CKO) is clock generator output to crystal/resonator
 - CKI is the clock input
- = 2 Single-pin RC controlled oscillator (CKI/10)
 - G7 is available as a HALT restart and/or general purpose input

OPTION 2: HALT

- = 1 Enable HALT mode
- = 2 Disable HALT mode

OPTION 3: COP888CF BONDING

- = 1 44-Pin PLCC
- = 2 40-Pin DIP
- = 3 N/A
- = 4 28-Pin DIP
- = 5 28-Pin SO

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface or maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use window interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC® via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable.
MHW-PS3	Power supply 110V/60 Hz
MHW-PS4	Power supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884CF28D5PC	28 DIP	4.5V-5.5V	COP884CF
MHW-884CF28DWPC	28 DIP	2.5V-6.0V	COP884CF
MHW-888CF40D5PC	40 DIP	4.5V-5.5V	COP888CF
MHW-888CF40DWPC	40 DIP	2.5V-6.0V	COP888CF
MWH-888CF44D5PC	44 PLCC	4.5V-5.5V	COP888CF
MHW-888CF44DWPC	44 PLCC	2.5V-6.0V	COP888CF

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM®, PC/XT®, PC-AT® or compatible.	424410527-001

Development Support (Continued)

SIMULATOR

The COP8 Designer's Tool Kit is available for evaluating National Semiconductor's COP8 microcontroller family. The kit contains programmer's manuals, device datasheets, pocket reference guide, assembler and simulator, which allows the user to write, test, debug and run code on an industry standard compatible PC. The simulator has a windowed user interface and can handle script files that simulate hardware inputs, interrupts and automatic command processing. The capture file feature enables the user to record to a file current cycle count and output port changes which are caused by the program under test.

Simulator Ordering Information

Part Number	Description	Manual
COP8-TOOL-KIT	COP8 Designer's Tool Kit Assembler and Simulator	420420270-001 424420269-001

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific datasheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board, which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888CFMHEL-X	X = 1: crystal X = 3: R/C	44 LDCC	Multi-Chip Module (MCM), UV erasable	COP888CF
COP888CFMHD-X	X = 1: crystal X = 3: R/C	40 DIP	MCM, UV erasable	COP888CF
COP884CFMHD-X	X = 1: crystal X = 3: R/C	28 DIP	MCM, UV erasable	COP884CF
COP884CFMHEA-X	X = 1: crystal X = 3: R/C	28 LCC	MCM (same footprint as 28 SO), UV erasable	COP884CF

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for use with Scrambler Boards	COP884CFMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP socket	COP888CFMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888CFMHEL
COP8-SCRM-SBX	Hybrid Scrambler Board for 28 LCC Socket	COP884CFMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP884CFMHD, COP888CFMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC Scrambler Board	COP888CFMEL, COP884CFMHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package Contents:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-Up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hours, 7 Days



COP884CG/COP888CG

Single-Chip microCMOS Microcontrollers

General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOSTM process technology. The COP888CG is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μ s instruction cycle time
- 4096 bytes on-board ROM (COP888CG)
- 192 bytes on-board RAM (COP888CG)
- Single supply operation: 2.5V-6V
- Full duplex UART
- Two analog comparators
- MICROWIRE/PLUSTM serial I/O
- WATCHDOGTM and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Fourteen multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Three Timers (Each with 2 Interrupts)
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - UART (2)
 - Default VIS
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PLCC or 40 N or 28 N or 28 PLCC
 - 44 PLCC with 39 I/O pins
 - 40 N with 35 I/O pins
 - 28 N with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: -40°C to +85°C, -55°C to +125°C
- Form factor emulation devices
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram

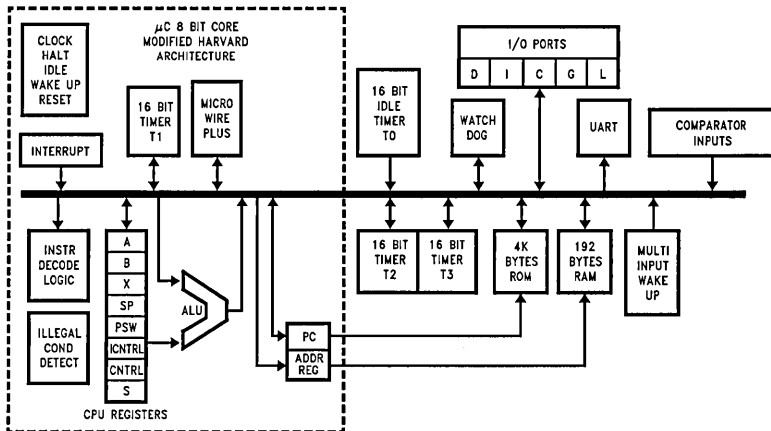


FIGURE 1. COP888CG Block Diagram

TL/DD/9765-1

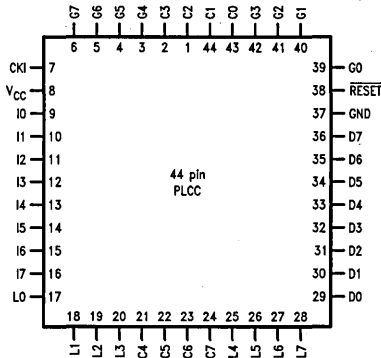
General Description (Continued)

They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes

(HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The devices operate over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

Connection Diagrams

Plastic Chip Carrier

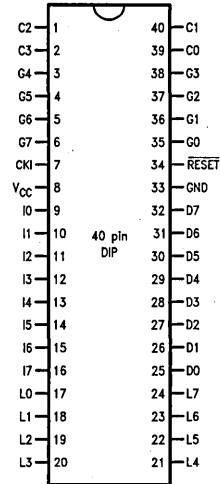


Top View

Order Number COP888CG-XXX/V
See NS Plastic Chip Package Number V44A

TL/DD/9765-2

Dual-In-Line Package

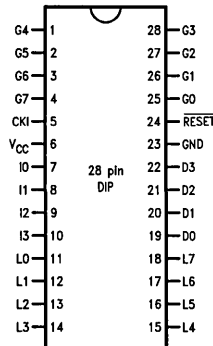


Top View

Order Number COP888CG-XXX/N
See NS Molded Package Number N40A

TL/DD/9765-4

Dual-In-Line Package



Top View

Order Number COP884CG-XXX/N
See NS Molded Package Number N28A

TL/DD/9765-5

FIGURE 2a. COP888CG Connection Diagrams

Connection Diagrams (Continued)

COP888CG Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU	T3A	17	23	27
L7	I/O	MIWU	T3B	18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
I0	I			7	9	9
I1	I	COMP1IN-		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I	COMP2IN-			13	13
I5	I	COMP2IN+			14	14
I6	I	COMP2OUT			15	15
I7	I				16	16
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V _{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			5.5	mA
CKI = 4 MHz	$V_{CC} = 3.5V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 3.5V, t_c = 10 \mu s$			1.4	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$ $V_{CC} = 3.5V, CKI = 0 MHz$		<1 <0.5	10 6	μA μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 3.5V, t_c = 10 \mu s$			0.7	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	0.4 0.2			mA mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	10 2.0			mA mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	10 2.5		100 33	μA μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	0.4 0.2			mA mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	1.6 0.7			mA mA
TRI-STATE Leakage	$V_{CC} = 6.0V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a crystal/resonator oscillator, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L, C, and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				15 3	mA mA
Maximum Input Current without Latchup (Note 5)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

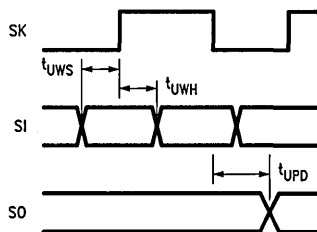
AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	1 2.5 3 7.5		DC DC DC DC	μs μs μs μs
CKI Clock Duty Cycle (Note 4) Rise Time (Note 4) Fall Time (Note 4)	$f_r = \text{Max}$ $f_r = 10\text{ MHz Ext Clock}$ $f_r = 10\text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	200 500 60 150			ns ns ns ns
Output Propagation Delay t_{PD1} , t_{PDO} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100\text{ pF}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$			0.7 1.75 1 2.5	μs μs μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Parameter sampled but not 100% tested.**Note 5:** Except pin G7: -60 mA to $+100\text{ mA}$ (sampled but not 100% tested).

Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25^\circ C$

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		± 10	± 25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current Per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



TL/DD/9765-7

FIGURE 2. MICROWIRE/PLUS Timing

Pin Descriptions

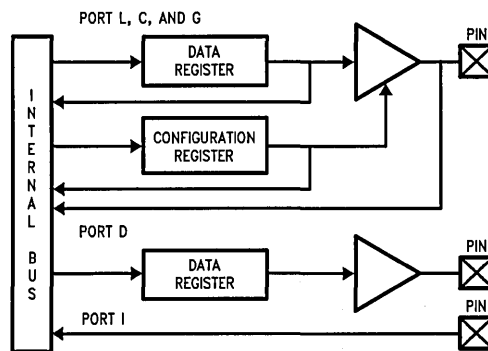
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CG contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports L and G), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the memory map for the various addresses associated with the I/O ports.) Figure 3 shows the I/O port configurations. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/9765-8

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

- L0 MIWU
- L1 MIWU or CKX
- L2 MIWU or TDX
- L3 MIWU or RDX
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU or T3A
- L7 MIWU or T3B

Port G is an 8-bit port with 5 I/O pins (G0, G2-G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2-G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2-G5) can be individually configured under software control.

Pin Descriptions (Continued)

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE™ Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOOUT WATCHDOG and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

Port C is an 8-bit I/O port. The 40-pin device does not have a full complement of Port C pins. The unavailable pins are not terminated. A read operation for these unterminated pins will return unpredictable values.

PORT I is an eight-bit Hi-Z input port. The 28-pin device does not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

- I1 COMP1–IN (Comparator 1 Negative Input)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I3 COMP1OUT (Comparator 1 Output)
- I4 COMP2–IN (Comparator 2 Negative Input)
- I5 COMP2+IN (Comparator 2 Positive Input)
- I6 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CG is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_c) cycle time.

There are six CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

S is the 8-bit Data Segment Address Register used to extend the lower half of the address range (00 to 7F) into 256 data segments of 128 bytes each.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

The program memory consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the devices vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X, SP pointers and S register.

The COP888CG has 192 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, B and S are memory mapped into this space at address locations 0FC to 0FF Hex respectively, with the other registers being available for general usage.

The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Data Memory Segment RAM Extension

Data memory address 0FF is used as a memory mapped location for the Data Segment Address Register (S).

The data store memory is either addressed directly by a single byte address within the instruction, or indirectly relative to the reference of the B, X, or SP pointers (each contains a single-byte address). This single-byte address allows an addressing range of 256 locations from 00 to FF hex. The upper bit of this single-byte address divides the data store memory into two separate sections as outlined previously. With the exception of the RAM register memory from address locations 00F0 to 00FF, all RAM memory is memory mapped with the upper bit of the single-byte address being equal to zero. This allows the upper bit of the single-byte address to determine whether or not the base address range (from 0000 to 00FF) is extended. If this upper bit equals one (representing address range 0080 to 00FF), then address extension does not take place. Alternatively, if this upper bit equals zero, then the data segment extension register S is used to extend the base address range (from 0000 to 007F) from XX00 to XX7F, where XX represents the 8 bits from the S register. Thus the 128-byte data segment extensions are located from addresses 0100 to 017F for data segment 1, 0200 to 027F for data segment 2, etc., up to FF00 to FF7F for data segment 255. The base address range from 0000 to 007F represents data segment 0.

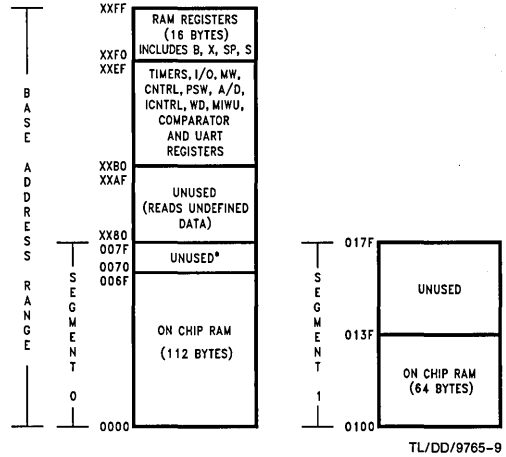
Figure 4 illustrates how the S register data memory extension is used in extending the lower half of the base address range (00 to 7F hex) into 256 data segments of 128 bytes each, with a total addressing range of 32 kbytes from XX00 to XX7F. This organization allows a total of 256 data segments of 128 bytes each with an additional upper base segment of 128 bytes. Furthermore, all addressing modes are available for all data segments. The S register must be changed under program control to move from one data segment (128 bytes) to another. However, the upper base segment (containing the 16 memory registers, I/O registers, control registers, etc.) is always available regardless of the contents of the S register, since the upper base segment (address range 0080 to 00FF) is independent of data segment extension.

The instructions that utilize the stack pointer (SP) always reference the stack as part of the base segment (Segment 0), regardless of the contents of the S register. The S register is not changed by these instructions. Consequently, the stack (used with subroutine linkage and interrupts) is always located in the base segment. The stack pointer will be initialized to point at data memory location 006F as a result of reset.

The 128 bytes of RAM contained in the base segment are split between the lower and upper base segments. The first 116 bytes of RAM are resident from address 0000 to 006F in the lower base segment, while the remaining 16 bytes of RAM represent the 16 data memory registers located at addresses 00F0 to 00FF of the upper base segment. No RAM is located at the upper sixteen addresses (0070 to 007F) of the lower base segment.

Additional RAM beyond these initial 128 bytes, however, will always be memory mapped in groups of 128 bytes (or less) at the data segment address extensions (XX00 to XX7F) of the lower base segment. The additional 64 bytes of RAM in

the COP888CG (beyond the initial 128 bytes) are memory mapped at address locations 0100 to 013F hex.



TL/DD/9765-9

*Reads as all ones.

FIGURE 4. RAM Organization

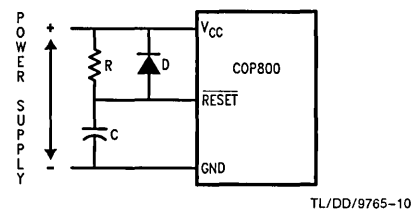
Reset

The **RESET** input when pulled low initializes the microcontroller. Initialization will occur whenever the **RESET** input is pulled low. Upon initialization, the data and configuration registers for ports L, G and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WATCHDOG and/or Clock Monitor error output pin. Port D is set high. The PC, PSW, ICNTRL, CNTRL, T2CNTRL and T3CNTRL control registers are cleared. The UART registers PSR, ENU (except that TBMT bit is set), ENUR and ENUL are cleared. The Comparator Select Register is cleared. The S register is initialized to zero. The Multi-Input Wakeup registers WKEN, WKEDG and WKPND are cleared. The stack pointer, SP, is initialized to 6F Hex.

The device comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, with the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor circuits are inhibited during reset. The WATCHDOG service window bits being initialized high default to the maximum WATCHDOG service window of 64k t_C clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16 t_C –32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in Figure 5 should be used to ensure that the **RESET** pin is held low until the power supply to the chip stabilizes.

Reset (Continued)



TL/DD/9765-10
 $RC > 5 \times \text{Power Supply Rise Time}$
FIGURE 5. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 6 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

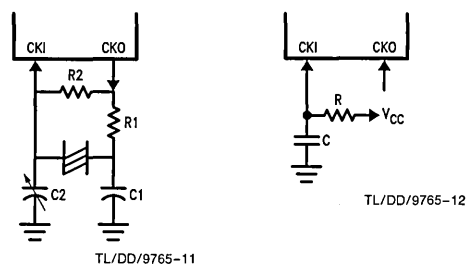
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9765-11
FIGURE 6. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4	$V_{CC} = 5.0V$
0	1	200	100-150	0.455	$V_{CC} = 5V$

TABLE B. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Note: $3k \leq R \leq 200k$
 $50 \text{ pF} \leq C \leq 200 \text{ pF}$

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. Comparator DC supply current when enabled—I6
7. Clock Monitor current when enabled—I7

Thus the total current drain, I_t , is given as
 $I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

where C = equivalent capacitance of the chip
 V = operating voltage
 f = CKI frequency

Some sample current drain values at $V_{CC} = 5V$ are:

CKI (MHz)	Inst. Cycle (μs)	I_t (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)	—	<0.001 (typ.)

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
- T1C0 Timer T1 Start/Stop control in timer modes 1 and 2
Timer T1 Underflow Interrupt Pending Flag in timer mode 3
- T1C1 Timer T1 mode control bit
- T1C2 Timer T1 mode control bit
- T1C3 Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7						Bit 0	

Control Registers (Continued)

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt
BUSY	MICROWIRE/PLUS busy shifting flag
EXPND	External interrupt pending
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
T1PNDA	Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
C	Carry Flag
HC	Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE
Bit 7				Bit 0			

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB	Timer T1 Interrupt Enable for T1B Input capture edge
T1PNDB	Timer T1 Interrupt Pending Flag for T1B capture edge
μ WEN	Enable MICROWIRE/PLUS interrupt
μ WPND	MICROWIRE/PLUS interrupt pending
TOEN	Timer T0 Interrupt Enable (Bit 12 toggle)
T0PND	Timer T0 Interrupt pending
LPEN	L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)
Bit 7 could be used as a flag	

Unused	LPEN	T0PND	TOEN	μ WPND	μ WEN	T1PNDB	T1ENB
Bit 7				Bit 0			

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB	Timer T2 Interrupt Enable for T2B Input capture edge
T2PNDB	Timer T2 Interrupt Pending Flag for T2B capture edge
T2ENA	Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
T2PNDA	Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
T2C0	Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

T2C1	Timer T2 mode control bit
T2C2	Timer T2 mode control bit
T2C3	Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

T3CNTRL Register (Address X'00B6)

The T3CNTRL register contains the following bits:

T3ENB	Timer T3 Interrupt Enable for T3B
T3PNDB	Timer T3 Interrupt Pending Flag for T3B pin (T3B capture edge)
T3ENA	Timer T3 Interrupt Enable for Timer Underflow or T3A pin
T3PNDA	Timer T3 Interrupt Pending Flag (Autoload RA in mode 1, T3 Underflow in mode 2, T3a capture edge in mode 3)
T3C0	Timer T3 Start/Stop control in timer modes 1 and 2 Timer T3 Underflow Interrupt Pending Flag in timer mode 3
T3C1	Timer T3 mode control bit
T3C2	Timer T3 mode control bit
T3C3	Timer T3 mode control bit

T3C3	T3C2	T3C1	T3C0	T3PNDA	T3ENA	T3PNDB	T3ENB
Bit 7							Bit 0

Timers

The COP888CG contains a very versatile set of timers (T0, T1, T2, T3). All timers and associated autoreload/capture registers power up containing random data.

TIMER T0 (IDLE TIMER)

The devices support applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c . The user cannot read or write to the IDLE Timer T0, which is a count down timer. The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)
WATCHDOG logic (See WATCHDOG description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_c = 1 \mu s$). A control flag TOEN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting TOEN will enable the interrupt, while resetting it will disable the interrupt.

Timers (Continued)

TIMER T1, TIMER T2 AND TIMER T3

The devices have a set of three powerful timer/counter blocks, T1, T2 and T3. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the three timer blocks, T1, T2 and T3 are identical, all comments are equally applicable to any of the three timer blocks.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the device to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the device to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_c . Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 7 shows a block diagram of the timer in PWM mode. The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

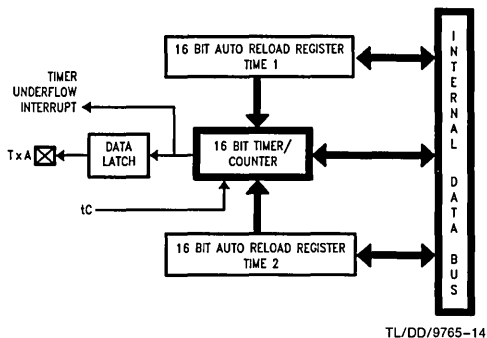


FIGURE 7. Timer in PWM Mode

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 8 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

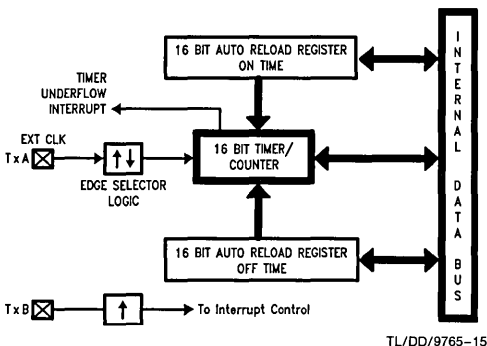


FIGURE 8. Timer in External Event Counter Mode

Mode 3. Input Capture Mode

The device can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_c rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

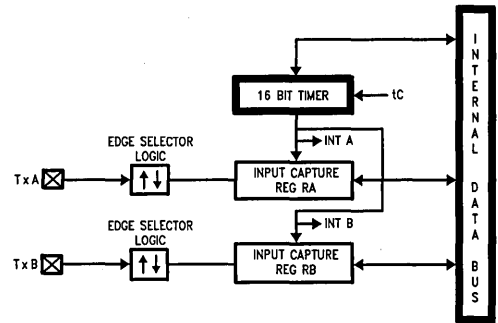
Timers (Continued)

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxCO pending flag (the TxCO control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxCO control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is generated when the selected trigger condition occurs on the TxA pin. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxCO pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 9 shows a block diagram of the timer in Input Capture mode.



TL/DD/9765-16

FIGURE 9. Timer In Input Capture Mode

TIMER CONTROL FLAGS

The timers T1, T2 and T3 have identical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop
TxC0	Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag
	1 = Timer Interrupt Enabled
	0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c

Power Save Modes

The devices offer the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry the WATCHDOG logic, the Clock Monitor and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The devices can be placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock and timers, are stopped. The WATCHDOG logic on the device is disabled during the HALT mode. However, the clock monitor circuitry if enabled remains active and will cause the WATCHDOG output pin (WDOUT) to go low. If the HALT mode is used and the user does not want to activate the WDOUT pin, the Clock Monitor should be disabled after the device comes out of reset (resetting the Clock Monitor control bit with the first write to the WDSVR register). In the HALT mode, the power requirements of the device are minimal and the applied voltage (V_{CC}) may be decreased to V_f ($V_f = 2.0V$) without altering the state of the machine.

The devices support three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock con-

figuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

Power Save Modes (Continued)

The devices have two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the device will enter and exit the HALT mode as described above. With the HALT disable mask option, the device cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WATCHDOG detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the device inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The device is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activities, except the associated on-board oscillator circuitry, the WATCHDOG logic, the clock monitor and the IDLE Timer T0, are stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the device can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_c = 1 \mu\text{s}$) of the IDLE Timer T0 is reached.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the TOPND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the TOPND bit gets set, the device will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the device will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the device from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 10 shows the Multi-Input Wakeup logic for the COP888CG microcontroller.

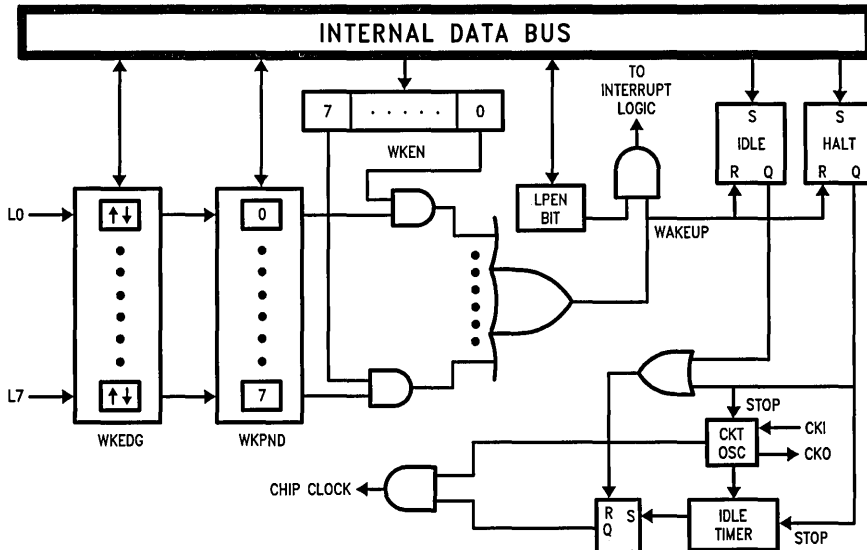


FIGURE 10. Multi-Input Wake Up Logic

TL/DD/9765-17

Multi-Input Wakeup (Continued)

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the device to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```

RBIT 5, WKEN
SBIT 5, WKEDG
RBIT 5, WKPND
SBIT 5, WKEN

```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the device will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the device out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the device will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the device will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T₀) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the device to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T₀ are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

UART

The COP888CG contains a full-duplex software programmable UART. The UART (Figure 11) consists of a transmit shift register, a receiver shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR), a UART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framing, data overrun and parity errors while the UART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the UART's attention mode of operation and providing additional receiver/transmitter status information via RCVG and XMTG bits. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts. A control flag in this register can also select the UART mode of operation: asynchronous or synchronous.

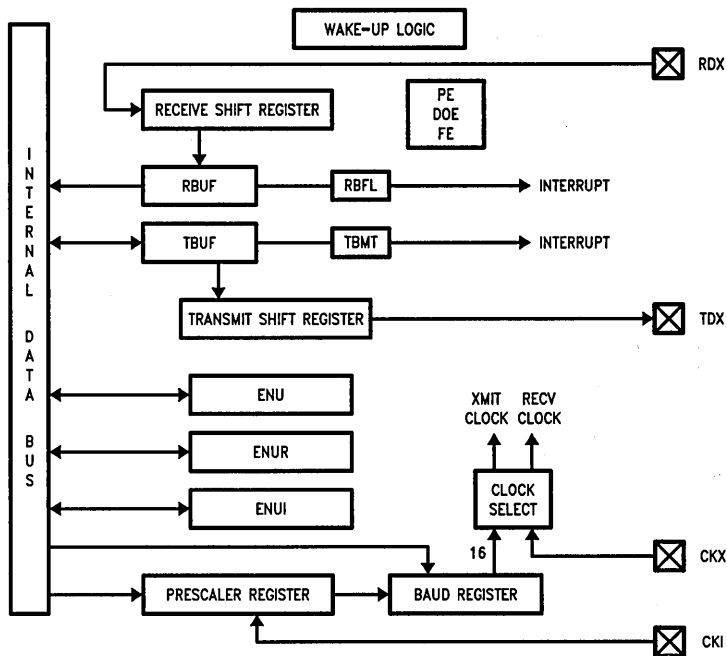


FIGURE 11. UART Block Diagram

TL/DD/9765-18

UART (Continued)

UART CONTROL AND STATUS REGISTERS

The operation of the UART is programmed through three registers: ENU, ENUR and ENUI. The function of the individual bits in these registers is as follows:

ENU-UART Control and Status Register (Address at 0BA)

PEN	PSEL1	XBIT9/ PSEL0	CHL1	CHL0	ERR	RBFL	TBMT
0RW	0RW	0RW	0RW	0RW	0R	0R	1R

Bit 7 Bit 0

ENUR-UART Receive Control and Status Register (Address at 0BB)

DOE	FE	PE	SPARE	RBIT9	ATTN	XMTG	RCVG
0RD	0RD	0RD	0RW*	0R	0RW	0R	0R

Bit7 Bit0

ENUI-UART Interrupt and Clock Source Register (Address at 0BC)

STP2	STP78	ETDX	SSEL	XRCLK	XTCLK	ERI	ETI
0RW	0RW	0RW	0RW	0RW	0RW	0RW	0RW

Bit7 Bit0

*Bit is not used.

- 0 Bit is cleared on reset.
- 1 Bit is set to one on reset.
- R Bit is read-only; it cannot be written by software.
- RW Bit is read/writes.
- D Bit is cleared on read; when read by software as a one, it is cleared automatically. Writing to the bit does not affect its state.

DESCRIPTION OF UART REGISTER BITS

ENU—UART CONTROL AND STATUS REGISTER

TBMT: This bit is set when the UART transfers a byte of data from the TBUF register into the TSFT register for transmission. It is automatically reset when software writes into the TBUF register.

RBFL: This bit is set when the UART has received a complete character and has copied it into the RBUF register. It is automatically reset when software reads the character from RBUF.

ERR: This bit is a global UART error flag which gets set if any or a combination of the errors (DOE, FE, PE) occur.

CHL1, CHL0: These bits select the character frame format. Parity is not included and is generated/verified by hardware.
 CHL1 = 0, CHL0 = 0 The frame contains eight data bits.
 CHL1 = 0, CHL0 = 1 The frame contains seven data bits.

CHL1 = 1, CHL0 = 0 The frame contains nine data bits.
 CHL1 = 1, CHL0 = 1 Loopback Mode selected. Transmitter output internally looped back to receiver input. Nine bit framing format is used.

XBIT9/PSEL0: Programs the ninth bit for transmission when the UART is operating with nine data bits per frame. For seven or eight data bits per frame, this bit in conjunction with PSEL1 selects parity.

PSEL1, PSEL0: Parity select bits.

- PSEL1 = 0, PSEL0 = 0 Odd Parity (if Parity enabled)
- PSEL1 = 0, PSEL0 = 1 Even Parity (if Parity enabled)

- PSEL1 = 1, PSEL0 = 0 Mark(1) (if Parity enabled)
- PSEL1 = 1, PSEL0 = 1 Space(0) (if Parity enabled)

PEN: This bit enables/disables Parity (7- and 8-bit modes only).
 PEN = 0 Parity disabled.
 PEN = 1 Parity enabled.

ENUR—UART RECEIVE CONTROL AND STATUS REGISTER

RCVG: This bit is set high whenever a framing error occurs and goes low when RDX goes high.

XMTG: This bit is set to indicate that the UART is transmitting. It gets reset at the end of the last frame (end of last Stop bit).

ATTN: ATTENTION Mode is enabled while this bit is set. This bit is cleared automatically on receiving a character with data bit nine set.

RBIT9: Contains the ninth data bit received when the UART is operating with nine data bits per frame.

SPARE: Reserved for future use.

PE: Flags a Parity Error.

- PE = 0 Indicates no Parity Error has been detected since the last time the ENUR register was read.
- PE = 1 Indicates the occurrence of a Parity Error.

FE: Flags a Framing Error.

- FE = 0 Indicates no Framing Error has been detected since the last time the ENUR register was read.
- FE = 1 Indicates the occurrence of a Framing Error.

DOE: Flags a Data Overrun Error.

- DOE = 0 Indicates no Data Overrun Error has been detected since the last time the ENUR register was read.
- DOE = 1 Indicates the occurrence of a Data Overrun Error.

ENUI—UART INTERRUPT AND CLOCK SOURCE REGISTER

ETI: This bit enables/disables interrupt from the transmitter section.

- ETI = 0 Interrupt from the transmitter is disabled.
- ETI = 1 Interrupt from the transmitter is enabled.

ERI: This bit enables/disables interrupt from the receiver section.

- ERI = 0 Interrupt from the receiver is disabled.
- ERI = 1 Interrupt from the receiver is enabled.

XTCLK: This bit selects the clock source for the transmitter-section.

- XTCLK = 0 The clock source is selected through the PSR and BAUD registers.
- XTCLK = 1 Signal on CKX (L1) pin is used as the clock.

XRCLK: This bit selects the clock source for the receiver section.

- XRCLK = 0 The clock source is selected through the PSR and BAUD registers.
- XRCLK = 1 Signal on CKX (L1) pin is used as the clock.

SSEL: UART mode select.

- SSEL = 0 Asynchronous Mode.
- SSEL = 1 Synchronous Mode.

UART (Continued)

ETDX: TDX (UART Transmit Pin) is the alternate function assigned to Port L pin L2; it is selected by setting ETDX bit. To simulate line break generation, software should reset ETDX bit and output logic zero to TDX pin through Port L data and configuration registers.

STP78: This bit is set to program the last Stop bit to be 7/8th of a bit in length.

STP2: This bit programs the number of Stop bits to be transmitted.

STP2 = 0 One Stop bit transmitted.

STP2 = 1 Two Stop bits transmitted.

Associated I/O Pins

Data is transmitted on the TDX pin and received on the RDX pin. TDX is the alternate function assigned to Port L pin L2; it is selected by setting ETDX (in the ENUI register) to one. RDX is an inherent function of Port L pin L3, requiring no setup.

The baud rate clock for the UART can be generated on-chip, or can be taken from an external source. Port L pin L1 (CKX) is the external clock I/O pin. The CKX pin can be either an input or an output, as determined by Port L Configuration and Data registers (Bit 1). As an input, it accepts a clock signal which may be selected to drive the transmitter and/or receiver. As an output, it presents the internal Baud Rate Generator output.

UART Operation

The UART has two modes of operation: asynchronous mode and synchronous mode.

ASYNCHRONOUS MODE

This mode is selected by resetting the SSEL (in the ENUI register) bit to zero. The input frequency to the UART is 16 times the baud rate.

The TSFT and TBUF registers double-buffer data for transmission. While TSFT is shifting out the current character on the TDX pin, the TBUF register may be loaded by software with the next byte to be transmitted. When TSFT finishes transmitting the current character the contents of TBUF are transferred to the TSFT register and the Transmit Buffer Empty Flag (TBMT in the ENU register) is set. The TBMT flag is automatically reset by the UART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the UART is transmitting. This bit gets reset at the end of the last frame (end of last Stop bit). TBUF is a read/write register.

The RSFT and RBUF registers double-buffer data being received. The UART receiver continually monitors the signal on the RDX pin for a low level to detect the beginning of a Start bit. Upon sensing this low level, it waits for half a bit time and samples again. If the RDX pin is still low, the receiver considers this to be a valid Start bit, and the remaining bits in the character frame are each sampled a single time, at the mid-bit position. Serial data input on the RDX pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the RBUF register and the Received Buffer Full Flag (RBFL) is set. RBFL is automatically reset when software reads the character from the RBUF register. RBUF is a read only register. There is also the RCVG bit which is set high

when a framing error occurs and goes low once RDX goes high. TBMT, XMTG, RBFL and RCVG are read only bits.

SYNCHRONOUS MODE

In this mode data is transferred synchronously with the clock. Data is transmitted on the rising edge and received on the falling edge of the synchronous clock.

This mode is selected by setting SSEL bit in the ENUI register. The input frequency to the UART is the same as the baud rate.

When an external clock input is selected at the CKX pin, data transmit and receive are performed synchronously with this clock through TDX/RDX pins.

If data transmit and receive are selected with the CKX pin as clock output, the device generates the synchronous clock output at the CKX pin. The internal baud rate generator is used to produce the synchronous clock. Data transmit and receive are performed synchronously with this clock.

FRAMING FORMATS

The UART supports several serial framing formats (*Figure 12*). The format is selected using control bits in the ENU, ENUR and ENUI registers.

The first format (1, 1a, 1b, 1c) for data transmission (CHL0 = 1, CHL1 = 0) consists of Start bit, seven Data bits (excluding parity) and 7/8, one or two Stop bits. In applications using parity, the parity bit is generated and verified by hardware.

The second format (CHL0 = 0, CHL1 = 0) consists of one Start bit, eight Data bits (excluding parity) and 7/8, one or two Stop bits. Parity bit is generated and verified by hardware.

The third format for transmission (CHL0 = 0, CHL1 = 1) consists of one Start bit, nine Data bits and 7/8, one or two Stop bits. This format also supports the UART "ATTENTION" feature. When operating in this format, all eight bits of TBUF and RBUF are used for data. The ninth data bit is transmitted and received using two bits in the ENU and ENUR registers, called XBIT9 and RBIT9. RBIT9 is a read only bit. Parity is not generated or verified in this mode.

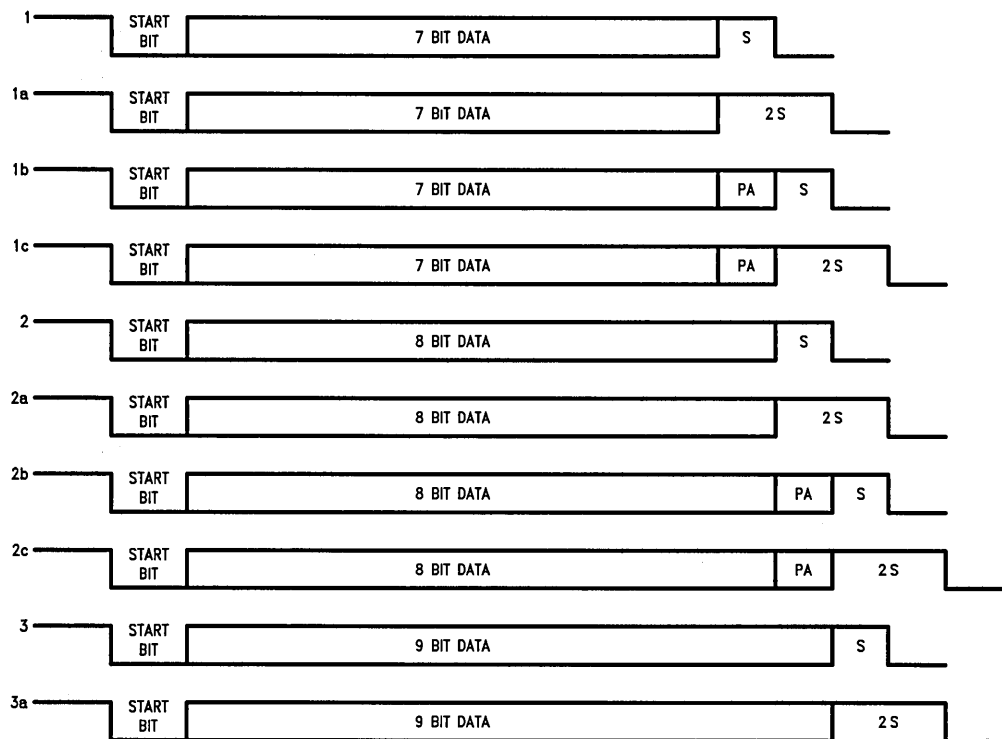
For any of the above framing formats, the last Stop bit can be programmed to be 7/8th of a bit in length. If two Stop bits are selected and the 7/8th bit is set (selected), the second Stop bit will be 7/8th of a bit in length.

The parity is enabled/disabled by PEN bit located in the ENU register. Parity is selected for 7- and 8-bit modes only. If parity is enabled (PEN = 1), the parity selection is then performed by PSEL0 and PSEL1 bits located in the ENU register.

Note that the XBIT9/PSEL0 bit located in the ENU register serves two mutually exclusive functions. This bit programs the ninth bit for transmission when the UART is operating with nine data bits per frame. There is no parity selection in this framing format. For other framing formats XBIT9 is not needed and the bit is PSEL0 used in conjunction with PSEL1 to select parity.

The frame formats for the receiver differ from the transmitter in the number of Stop bits required. The receiver only requires one Stop bit in a frame, regardless of the setting of the Stop bit selection bits in the control register. Note that an implicit assumption is made for full duplex UART operation that the framing formats are the same for the transmitter and receiver.

UART Operation (Continued)



TL/DD/9765-19

FIGURE 12. Framing Formats

UART INTERRUPTS

The UART is capable of generating interrupts. Interrupts are generated on Receive Buffer Full and Transmit Buffer Empty. Both interrupts have individual interrupt vectors. Two bytes of program memory space are reserved for each interrupt vector. The two vectors are located at addresses 0xEC to 0xEF Hex in the program memory space. The interrupts can be individually enabled or disabled using Enable Transmit Interrupt (ETI) and Enable Receive Interrupt (ERI) bits in the ENUI register.

The interrupt from the Transmitter is set pending, and remains pending, as long as both the TBMT and ETI bits are set. To remove this interrupt, software must either clear the ETI bit or write to the TBUF register (thus clearing the TBMT bit).

The interrupt from the receiver is set pending, and remains pending, as long as both the RBFL and ERI bits are set. To remove this interrupt, software must either clear the ERI bit or read from the RBUF register (thus clearing the RBFL bit).

Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the UART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a

source selected in the PSR and BAUD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1-16 (increments of 0.5) prescaler and an 11-bit binary counter. (Figure 13) The divide factors are specified through two read/write registers shown in Figure 14. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

As shown in Table I, a Prescaler Factor of 0 corresponds to NO CLOCK. NO CLOCK condition is the UART power down mode where the UART clock is turned off for power saving purpose. The user must also turn the UART clock off when a different baud rate is chosen.

The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in Table I. There are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a x16 clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (Table II). Other baud rates may be created by using appropriate divisors. The x16 clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.

Baud Clock Generation (Continued)

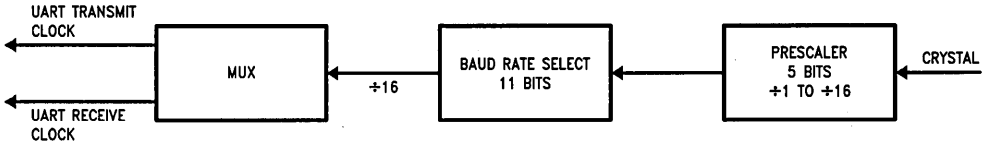


FIGURE 13. UART BAUD Clock Generation

TL/DD/9765-20

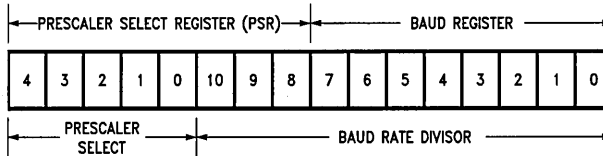


FIGURE 14. UART BAUD Clock Divisor Registers

TL/DD/9765-21

TABLE I. Prescaler Factors

Prescaler Select	Prescaler Factor
00000	NO CLOCK
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11001	13
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

TABLE II. Baud Rate Divisors (1.8432 MHz Prescaler Output)

Baud Rate	Baud Rate Divisor - 1 (N-1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5
38400	2

The entries in Table II assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 625k.

As an example, considering the Asynchronous Mode and a CKI clock of 4.608 MHz, the prescaler factor selected is:

$$4.608 / 1.8432 = 2.5$$

The 2.5 entry is available in Table I. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (Table II) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in Table II is 5.

$$N - 1 = 5 \quad (N - 1 \text{ is the value from Table II})$$

$$N = 6 \quad (N \text{ is the Baud Rate Divisor})$$

$$\text{Baud Rate} = 1.8432 \text{ MHz} / (16 \times 6) = 19200$$

The divide by 16 is performed because in the asynchronous mode, the input frequency to the UART is 16 times the baud rate. The equation to calculate baud rates is given below.

The actual Baud Rate may be found from:

$$BR = Fc / (16 \times N \times P)$$

Baud Clock Generation (Continued)

Where:

BR is the Baud Rate

F_c is the CKI frequency

N is the Baud Rate Divisor (Table II).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (Table I)

Note: In the Synchronous Mode, the divisor 16 is replaced by two.

Example:

Asynchronous Mode:

$$\text{Crystal Frequency} = 5 \text{ MHz}$$

$$\text{Desired baud rate} = 9600$$

Using the above equation $N \times P$ can be calculated first.

$$N \times P = (5 \times 10^6) / (16 \times 9600) = 32.552$$

Now 32.552 is divided by each Prescaler Factor (Table II) to obtain a value closest to an integer. This factor happens to be 6.5 (P = 6.5).

$$N = 32.552 / 6.5 = 5.008 \quad (N = 5)$$

The programmed value (from Table II) should be 4 (N - 1).

Using the above values calculated for N and P:

$$\text{BR} = (5 \times 10^6) / (16 \times 5 \times 6.5) = 9615.384$$

$$\% \text{ error} = (9615.385 - 9600) / 9600 = 0.16$$

Effect of HALT/IDLE

The UART logic is reinitialized when either the HALT or IDLE modes are entered. This reinitialization sets the TBMT flag and resets all read only bits in the UART control and status registers. Read/Write bits remain unchanged. The Transmit Buffer (TBUF) is not affected, but the Transmit Shift register (TSFT) bits are set to one. The receiver registers RBUF and RSFT are not affected.

The device will exit from the HALT/IDLE modes when the Start bit of a character is detected at the RDX (L3) pin. This feature is obtained by using the Multi-Input Wakeup scheme provided on the device.

Before entering the HALT or IDLE modes the user program must select the Wakeup source to be on the RDX pin. This selection is done by setting bit 3 of WKEN (Wakeup Enable) register. The Wakeup trigger condition is then selected to be high to low transition. This is done via the WKEDG register (Bit 3 is zero.)

If the device is halted and crystal oscillator is used, the Wakeup signal will not start the chip running immediately because of the finite start up time requirement of the crystal oscillator. The idle timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the device to execute code. The user has to consider this delay when data transfer is expected immediately after exiting the HALT mode.

Diagnostic

Bits CHARL0 and CHARL1 in the ENU register provide a loopback feature for diagnostic testing of the UART. When these bits are set to one, the following occur: The receiver input pin (RDX) is internally connected to the transmitter output pin (TDX); the output of the Transmitter Shift Register is "looped back" into the Receive Shift Register input. In this mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the UART.

Note that the framing format for this mode is the nine bit format; one Start bit, nine data bits, and 7/8, one or two Stop bits. Parity is not generated or verified in this mode.

Attention Mode

The UART Receiver section supports an alternate mode of operation, referred to as ATTENTION Mode. This mode of operation is selected by the ATTN bit in the ENUR register. The data format for transmission must also be selected as having nine Data bits and either 7/8, one or two Stop bits.

The ATTENTION mode of operation is intended for use in networking the device with other processors. Typically in such environments the messages consists of device addresses, indicating which of several destinations should receive them, and the actual data. This Mode supports a scheme in which addresses are flagged by having the ninth bit of the data field set to a 1. If the ninth bit is reset to a zero the byte is a Data byte.

While in ATTENTION mode, the UART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the UART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if UART Receiver interrupts are enabled. The ATTN bit is also cleared automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

Operation of the UART Transmitter is not affected by selection of this Mode. The value of the ninth bit to be transmitted is programmed by setting XBIT9 appropriately. The value of the ninth bit received is obtained by reading RBIT9. Since this bit is located in ENUR register where the error flags reside, a bit operation on it will reset the error flags.

Comparators

The devices contain two differential comparators, each with a pair of inputs (positive and negative) and an output. Ports I1-I3 and I4-I6 are used for the comparators. The following is the Port I assignment:

- I1 Comparator1 negative input
- I2 Comparator1 positive input
- I3 Comparator1 output
- I4 Comparator2 negative input
- I5 Comparator2 positive input
- I6 Comparator2 output

A Comparator Select Register (CMPSL) is used to enable the comparators, read the outputs of the comparators internally, and enable the outputs of the comparators to the pins. Two control bits (enable and output enable) and one result bit are associated with each comparator. The comparator result bits (CMP1RD and CMP2RD) are read only bits which will read as zero if the associated comparator is not enabled. The Comparator Select Register is cleared with reset, resulting in the comparators being disabled. The comparators should also be disabled before entering either the HALT or IDLE modes in order to save power. The configuration of the CMPSL register is as follows:

Comparators (Continued)

CMPSL REGISTER (ADDRESS X'00B7)

The CMPSL register contains the following bits:

- CMP1EN Enable comparator 1
- CMP1RD Comparator 1 result (this is a read only bit, which will read as 0 if the comparator is not enabled)
- CMP10E Selects pin 13 as comparator 1 output provided that CMP1EN is set to enable the comparator
- CMP2EN Enable comparator 2
- CMP2RD Comparator 2 result (this is a read only bit, which will read as 0 if the comparator is not enabled)
- CMP20E Selects pin 16 as comparator 2 output provided that CMP2EN is set to enable the comparator

Unused	CMP20E	CMP2RD	CMP2EN	CMP10E	CMP1RD	CMP1EN	Unused
Bit 7							Bit 0

Note that the two unused bits of CMPSL may be used as software flags.

Comparator outputs have the same spec as Ports L and G except that the rise and fall times are symmetrical.

Interrupts

The devices support a vectored interrupt scheme. It supports a total of fourteen interrupt sources. The following table lists all the possible device interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_c cycles to execute.

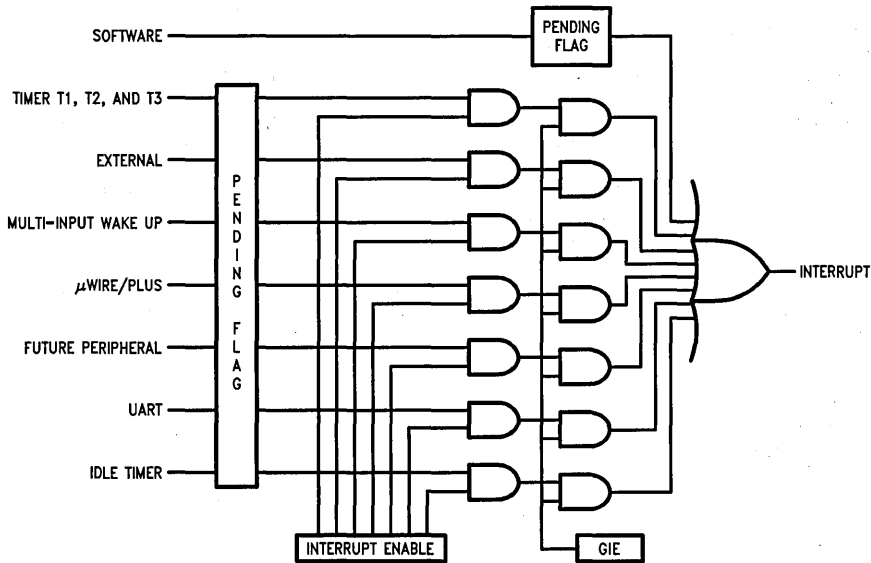


FIGURE 15. Interrupt Block Diagram

TL/DD/9765-22

Interrupts (Continued)

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	Underflow	0yF8–0yF9
(4)	Timer T1	T1A/Underflow	0yF6–0yF7
(5)	Timer T1	T1B	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
(7)	UART	Receive	0yEE–0yEF
(8)	UART	Transmit	0yEC–0yED
(9)	Timer T2	T2A/Underflow	0yEA–0yEB
(10)	Timer T2	T2B	0yE8–0yE9
(11)	Timer T3	T3A/Underflow	0yE6–0yE7
(12)	Timer T3	T3B	0yE4–0yE5
(13)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(14) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page, y ≠ 0.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block (y ≠ 0).

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 15 shows the Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

Interrupts (Continued)

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit (**not accessible by the user**) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WATCHDOG

The devices contain a WATCHDOG and clock monitor. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WATCHDOG consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WATCHDOG consists of writing a specific value to a WATCHDOG Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table III shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table IV shows the four possible combinations of lower and upper limits for the WATCHDOG service window. This flexibility in choosing the WATCHDOG service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE III. WATCHDOG Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE IV. WATCHDOG Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k–8k t_c Cycles
0	1	2k–16k t_c Cycles
1	0	2k–32k t_c Cycles
1	1	2k–64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the device can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WATCHDOG Operation

The WATCHDOG and Clock Monitor are disabled during reset. The device comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCHDOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. Table V shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WATCHDOG, the logic will pull the WDOUT (G1) pin low for an additional $16 t_c - 32 t_c$ cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the device will stop forcing the WDOUT output low.

The WATCHDOG service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WATCHDOG service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WATCHDOG will time out and WDOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following $16 t_c - 32 t_c$ clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10 \text{ kHz}$ —No clock rejection.

$1/t_c < 10 \text{ Hz}$ —Guaranteed clock rejection.

Watchdog and Clock Monitor Summary

The following salient points regarding the COP888CG WATCHDOG and CLOCK MONITOR should be noted:

- Both the WATCHDOG and CLOCK MONITOR detector circuits are inhibited during RESET.
- Following RESET, the WATCHDOG and CLOCK MONITOR are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and CLOCK MONITOR enable/disable option can only be changed once, during the initial WATCHDOG service following RESET.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.
- The CLOCK MONITOR detector circuit is active during both the HALT and IDLE modes. Consequently, the COP888 inadvertently entering the HALT mode will be detected as a CLOCK MONITOR error (provided that the CLOCK MONITOR enable option has been selected by the program).
- With the single-pin R/C oscillator mask option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.
- With the crystal oscillator mask option selected, or with the single-pin R/C oscillator mask option selected and the CLKDLY bit set, the WATCHDOG service window will be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.
- The IDLE timer T0 is not initialized with RESET.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the TOPND flag. The TOPND flag is set whenever the thirteenth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the TOPND flag.
- A hardware WATCHDOG service occurs just as the device exits the IDLE mode. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following RESET, the initial WATCHDOG service (where the service window and the CLOCK MONITOR enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 2048 instruction cycles without causing a WATCHDOG error.

Detection of Illegal Conditions

The device can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F (Segment 0), 140 to 17F (Segment 1), and all other segments (i.e., Segments 3 . . . etc.) is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

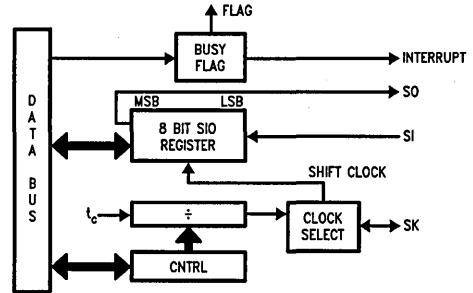
Thus, the chip can detect the following illegal conditions:

- Executing from undefined ROM
- Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before re-starting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the device to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E²PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 12 shows a block diagram of the MICROWIRE/PLUS logic.



TL/DD/9765-23

FIGURE 16. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table VI details the different clock rates that may be selected.

TABLE V. WATCHDOG Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

TABLE VI. MICROWIRE/PLUS
Master Mode Clock Select

SL1	SL0	SK
0	0	$2 \times t_c$
0	1	$4 \times t_c$
1	x	$8 \times t_c$

Where t_c is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The device may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 13* shows how two devices, microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock for the SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the device. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table VII summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table VII summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

The device allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE VII

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave

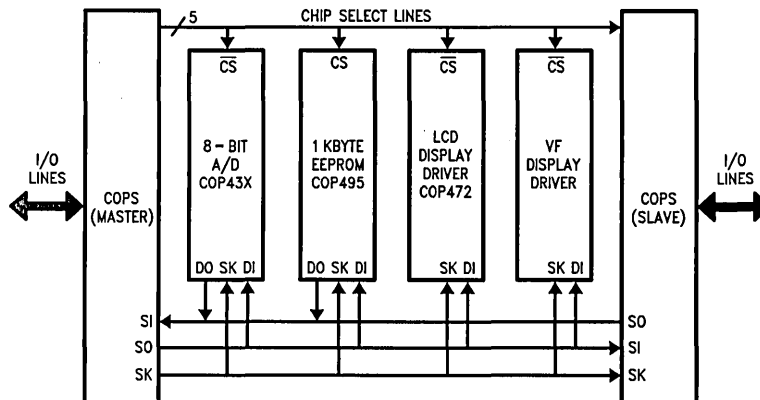


FIGURE 17. MICROWIRE/PLUS Application

TL/DD/9765-24

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address S/ADD REG	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads As All Ones)
xx80 to xxAF	Unused RAM Address Space (Reads Undefined Data)
xxB0	Timer T3 Lower Byte
XXB1	Timer T3 Upper Byte
xxB2	Timer T3 Autoload Register T3RA Lower Byte
xxB3	Timer T3 Autoload Register T3RA Upper Byte
xxB4	Timer T3 Autoload Register T3RB Lower Byte
xxB5	Timer T3 Autoload Register T3RB Upper Byte
xxB6	Timer T3 Control Register
xxB7	Comparator Select Register (CMPSL)
xxB8	UART Transmit Buffer (TBUF)
xxB9	UART Receive Buffer (RBUF)
xxBA	UART Control and Status Register (ENU)
xxBB	UART Receive Control and Status Register (ENUR)
xxBC	UART Interrupt and Clock Source Register (ENUI)
xxBD	UART Baud Register (BAUD)
xxBE	UART Prescale Select Register (PSR)
xxBF	Reserved for UART
xxC0	Timer T2 Lower Byte
xxC1	Timer T2 Upper Byte
xxC2	Timer T2 Autoload Register T2RA Lower Byte
xxC3	Timer T2 Autoload Register T2RA Upper Byte
xxC4	Timer T2 Autoload Register T2RB Lower Byte
xxC5	Timer T2 Autoload Register T2RB Upper Byte
xxC6	Timer T2 Control Register
xxC7	WATCHDOG Service Register (Reg:WDSVR)
xxC8	MIWU Edge Select Register (Reg:WKEDG)
xxC9	MIWU Enable Register (Reg:WKEN)
xxCA	MIWU Pending Register (Reg:WKPND)
xxCB	Reserved
xxCC	Reserved
xxCD to xxCF	Reserved

Address S/ADD REG	Contents
xxD0	Port L Data Register
xxD1	Port L Configuration Register
xxD2	Port L Input Pins (Read Only)
xxD3	Reserved for Port L
xxD4	Port G Data Register
xxD5	Port G Configuration Register
xxD6	Port G Input Pins (Read Only)
xxD7	Port I Input Pins (Read Only)
xxD8	Port C Data Register
xxD9	Port C Configuration Register
xxDA	Port C Input Pins (Read Only)
xxDB	Reserved for Port C
xxDC	Port D
xxDD to DF	Reserved for Port D
xxE0 to xxE5	Reserved for EE Control Registers
xxE6	Timer T1 Autoload Register T1RB Lower Byte
xxE7	Timer T1 Autoload Register T1RB Upper Byte
xxE8	ICNTRL Register
xxE9	MICROWIRE/PLUS Shift Register
xxEA	Timer T1 Lower Byte
xxEB	Timer T1 Upper Byte
xxEC	Timer T1 Autoload Register T1RA Lower Byte
xxED	Timer T1 Autoload Register T1RA Upper Byte
xxEE	CNTRL Control Register
xxEF	PSW Register
xxF0 to FB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	S Register
0100–013F	On-Chip 64 RAM Bytes

Reading memory locations 0070H–007FH (Segment 0) will return all ones. Reading unused memory locations 0080H–00AFH (Segment 0) will return undefined data. Reading unused memory locations 0140–017F (Segment 1) will return all ones. Reading memory locations from other Segments (i.e., Segment 2, Segment 3, ... etc.) will return all ones.

Addressing Modes

There are ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the Load B Immediate instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from -31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	$A \leftarrow A + \text{Meml}$
ADC	A,Meml	ADD with Carry	$A \leftarrow A + \text{Meml} + C$, $C \leftarrow \text{Carry}$
SUBC	A,Meml	Subtract with Carry	$HC \leftarrow \text{Half Carry}$ $A \leftarrow A - \text{Meml} + C$, $C \leftarrow \text{Carry}$ $HC \leftarrow \text{Half Carry}$
AND	A,Meml	Logical AND	$A \leftarrow A \text{ and Meml}$
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if $(A \text{ and Imm}) = 0$
OR	A,Meml	Logical OR	$A \leftarrow A \text{ or Meml}$
XOR	A,Meml	Logical EXclusive OR	$A \leftarrow A \text{ xor Meml}$
IFEQ	MD,Imm	IF EQUAL	Compare MD and Imm, Do next if $MD = \text{Imm}$
IFEQ	A,Meml	IF EQUAL	Compare A and Meml, Do next if $A = \text{Meml}$
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if $A \neq \text{Meml}$
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if $A > \text{Meml}$
IFBNE	#	If B Not Equal	Do next if lower 4 bits of $B \neq \text{Imm}$
DRSZ	Reg	Decrement Reg., Skip if Zero	$\text{Reg} \leftarrow \text{Reg} - 1$, Skip if $\text{Reg} = 0$
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	$A \leftrightarrow \text{Mem}$
X	A,[X]	EXchange A with Memory [X]	$A \leftrightarrow [X]$
LD	A,Meml	LoAD A with Memory	$A \leftarrow \text{Meml}$
LD	A,[X]	LoAD A with Memory [X]	$A \leftarrow [X]$
LD	B,Imm	LoAD B with Immed.	$B \leftarrow \text{Imm}$
LD	Mem,Imm	LoAD Memory Immed	$\text{Mem} \leftarrow \text{Imm}$
LD	Reg,Imm	LoAD Register Memory Immed.	$\text{Reg} \leftarrow \text{Imm}$
X	A, [B ±]	EXchange A with Memory [B]	$A \leftrightarrow [B]$, $(B \leftarrow B \pm 1)$
X	A, [X ±]	EXchange A with Memory [X]	$A \leftrightarrow [X]$, $(X \leftarrow X \pm 1)$
LD	A, [B ±]	LoAD A with Memory [B]	$A \leftarrow [B]$, $(B \leftarrow B \pm 1)$
LD	A, [X ±]	LoAD A with Memory [X]	$A \leftarrow [X]$, $(X \leftarrow X \pm 1)$
LD	[B ±],Imm	LoAD Memory [B] Immed.	$[B] \leftarrow \text{Imm}$, $(B \leftarrow B \pm 1)$
CLR	A	CLear A	$A \leftarrow 0$
INC	A	INCRement A	$A \leftarrow A + 1$
DEC	A	DECRementA	$A \leftarrow A - 1$
LAI		LoAD A INDirect from ROM	$A \leftarrow \text{ROM}(\text{PU},A)$
DCOR	A	DecImal CORrection A	$A \leftarrow \text{BCD correction of } A \text{ (follows ADC, SUBC)}$
RRC	A	RotatE A Right thru C	$C \rightarrow A7 \rightarrow \dots \rightarrow A0 \rightarrow C$
RLC	A	RotatE A Left thru C	$C \leftarrow A7 \leftarrow \dots \leftarrow A0 \leftarrow C$
SWAP	A	SWAP nibbles of A	$A7 \dots A4 \leftrightarrow A3 \dots A0$
SC		Set C	$C \leftarrow 1$, $HC \leftarrow 1$
RC		Reset C	$C \leftarrow 0$, $HC \leftarrow 0$
IFC		IF C	If C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	$\text{SP} \leftarrow \text{SP} + 1$, $A \leftarrow [\text{SP}]$
PUSH	A	PUSH A onto the stack	$[\text{SP}] \leftarrow A$, $\text{SP} \leftarrow \text{SP} - 1$
VIS		Vector to Interrupt Service Routine	$\text{PU} \leftarrow [\text{VU}]$, $\text{PL} \leftarrow [\text{VL}]$
JMPL	Addr.	Jump absolute Long	$\text{PC} \leftarrow ii$ ($ii = 15$ bits, 0 to 32k)
JMP	Addr.	Jump absolute	$\text{PC9} \dots 0 \leftarrow i$ ($i = 12$ bits)
JP	Disp.	Jump relative short	$\text{PC} \leftarrow \text{PC} + r$ (r is -31 to $+32$, except 1)
JSRL	Addr.	Jump SubRoutine Long	$[\text{SP}] \leftarrow \text{PL}$, $[\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2$, $\text{PC} \leftarrow ii$
JSR	Addr	Jump SubRoutine	$[\text{SP}] \leftarrow \text{PL}$, $[\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2$, $\text{PC9} \dots 0 \leftarrow i$
JID		Jump INDirect	$\text{PL} \leftarrow \text{ROM}(\text{PU},A)$
RET		RETurn from subroutine	$\text{SP} + 2$, $\text{PL} \leftarrow [\text{SP}]$, $\text{PU} \leftarrow [\text{SP}-1]$
RETSK		RETurn and SKip	$\text{SP} + 2$, $\text{PL} \leftarrow [\text{SP}]$, $\text{PU} \leftarrow [\text{SP}-1]$
RETI		RETurn from Interrupt	$\text{SP} + 2$, $\text{PL} \leftarrow [\text{SP}]$, $\text{PU} \leftarrow [\text{SP}-1]$, $\text{GIE} \leftarrow 1$
INTR		Generate an Interrupt	$[\text{SP}] \leftarrow \text{PL}$, $[\text{SP}-1] \leftarrow \text{PU}$, $\text{SP}-2$, $\text{PC} \leftarrow \text{OFF}$
NOP		No OPeration	$\text{PC} \leftarrow \text{PC} + 1$

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAI	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

RPND	1/1
------	-----

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.	
	[B]	[X]			[B+, B-]	[X+, X-]
XA,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B, Imm				1/1		
LD B, Imm				2/2		
LD Mem, Imm	2/2		3/3		2/2	
LD Reg, Imm			2/3			
IFEQ MD, Imm			3/3			

(IF B < 16)
(IF B > 15)

* = > Memory location addressed by B or X or directly.

Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP -15	JP -31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A,[B]	0
JP -14	JP -30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBCA, #i	SUB A,[B]	1
JP -13	JP -29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A, #i	IFEQ A,[B]	2
JP -12	JP -28	LD 0F3, # i	DRSZ 0F3	X A, [X-]	X A,[B-]	IFGT A, #i	IFGT A,[B]	3
JP -11	JP -27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A, #i	ADD A,[B]	4
JP -10	JP -26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A, #i	AND A,[B]	5
JP -9	JP -25	LD 0F6, # i	DRSZ 0F6	X A, [X]	X A,[B]	XOR A, #i	XOR A,[B]	6
JP -8	JP -24	LD 0F7, # i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	7
JP -7	JP -23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A, #i	IFC	8
JP -6	JP -22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md, #i	IFNE A, #i	IFNC	9
JP -5	JP -21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+], #i	INCA	A
JP -4	JP -20	LD 0FB, # i	DRSZ 0FB	LD A,[X-]	LD A,[B-]	LD [B-], #i	DECA	B
JP -3	JP -19	LD 0FC, # i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	POPA	C
JP -2	JP -18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	D
JP -1	JP -17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B], #i	RET	E
JP -0	JP -16	LD 0FF, # i	DRSZ 0FF	*	*	LD B, #i	RETI	F

Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B, #0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP + 17	INTR	0
IFBIT 1,[B]	*	LD B, #0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP + 18	JP + 2	1
IFBIT 2,[B]	*	LD B, #0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP + 19	JP + 3	2
IFBIT 3,[B]	*	LD B, #0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP + 20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B, #0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP + 21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B, #0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP + 22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B, #09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP + 23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B, #08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP + 24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B, #07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP + 25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B, #06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP + 26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B, #05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP + 27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B, #04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP + 28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B, #03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP + 29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B, #02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP + 30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B, #01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP + 31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B, #00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP + 32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A**Mask Options**

The COP888CG mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

- = 1 Crystal Oscillator (CKI/10)
 - G7 (CKO) is clock generator output to crystal/resonator
 - CKI is the clock input
- = 2 Single-pin RC controlled oscillator (CKI/10)
 - G7 is available as a HALT restart and/or general purpose input

OPTION 2: HALT

- = 1 Enable HALT mode
- = 2 Disable HALT mode

OPTION 3: BONDING OPTIONS

- = 1 44-Pin PLCC
- = 2 40-Pin DIP
- = 3 N/A
- = 4 28-Pin DIP

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock (1/t_c).

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface or maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use window interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC® via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS-232 serial interface cable
MHW-PS3	Power supply 110V/60 MHz
MHW-PS4	Power supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884CG28D5PC	28 DIP	4.5V-5.5V	COP884CG
MHW-884CG28DWPC	28 DIP	2.5V-6.0V	COP884CG
MHW-888CG40D5PC	40 DIP	4.5V-5.5V	COP888CG
MHW-888CG40DWPC	40 DIP	2.5V-6.0V	COP888CG
MWH-888CG44D5PC	44 PLCC	4.5V-5.5V	COP888CG
MHW-888CG44DWPC	44 PLCC	2.5V-6.0V	COP888CG

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM®, PC-/XT®, PC-AT® or compatible	424410527-001

Development Support (Continued)

SIMULATOR

The COP8 Designer's Tool Kit is available for evaluating National Semiconductor's COP8 microcontroller family. The kit contains programmer's manuals, device datasheets, pocket reference guide, assembler and simulator, which allows the user to write, test, debug and run code on an industry standard compatible PC. The simulator has a windowed user interface and can handle script files that simulate hardware inputs, interrupts and automatic command processing. The capture file feature enables the user to record to a file current cycle count and output port changes which are caused by the program under test.

Simulator Ordering Information

Part Number	Description	Manual
COP8-TOOL-KIT	COP8 Designer's Tool Kit Assembler and Simulator	420420270-001 424420269-001

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulator device specific datasheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888CGMHEL-X	X = 1: crystal X = 3: R/C	44 LDCC	Multi-Chip Module (MCM), UV erasable	COP888CG
COP888CGMHD-X	X = 1: crystal X = 3: R/C	40 DIP	MCM, UV erasable	COP888CG
COP884CGMHD-X	X = 1: crystal X = 3: R/C	28 DIP	MCM, UV erasable	COP884CG
COP884CGMHEA-X	X = 1: crystal X = 3: R/C	28 LCC	MCM (same footprint as 28 SO), UV erasable	COP884CG

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for use with Scrambler Boards	COP884CGMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP socket	COP888CGMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888CGMHEL
COP8-SCRM-SBX	MCM Scrambler board for 28 LCC socket	COP884CGMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP scrambler board	COP884CGMHD, COP888CGMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC scrambler board	COP888CGMEL, COP884CGMHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days



COP688EG/COP684EG/COP888EG/COP884EG

Single-Chip microCMOS Microcontrollers

General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. The COP888EG/COP884EG is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 8k bytes on-board ROM
- 256 bytes on-board RAM
- Single supply operation: 2.5V–6V
- Full duplex UART
- Two analog comparators
- MICROWIRE/PLUS™ serial I/O
- WATCHDOG™ and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Three 16-bit timers, each with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Fourteen multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Three Timers (Each with 2 Interrupts)
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - UART (2)
 - Default VIS
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PLCC or 40 N or 28 N or 28 SO
 - 44 PLCC with 39 I/O pins
 - 40 N with 35 I/O pins
 - 28 SO or 28 N, each with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: –40°C to +85°C, –55°C to +125°C
- Form factor emulation devices
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram

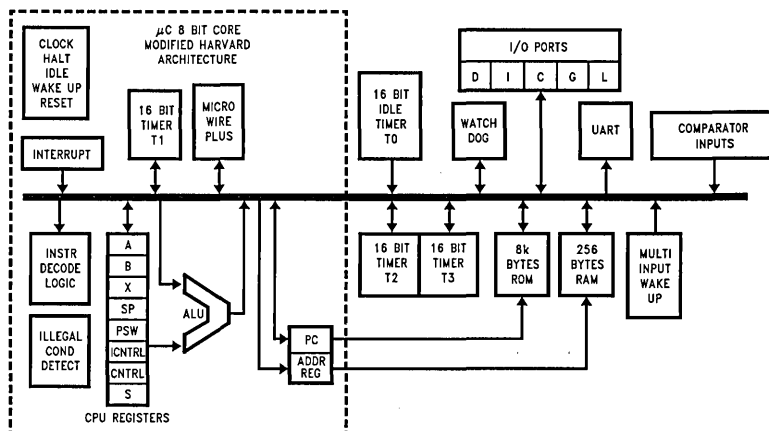


FIGURE 1. COP888EG Block Diagram

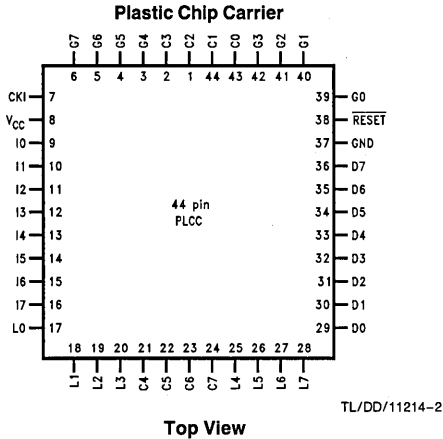
TL/DD/11214-1

General Description (Continued)

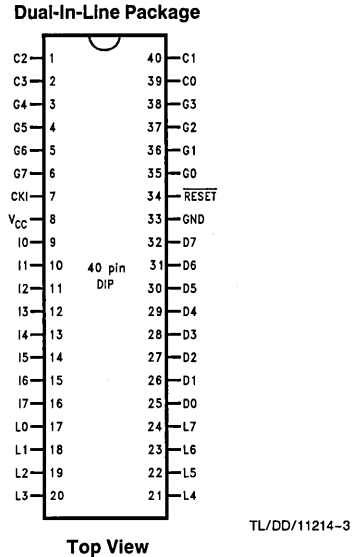
They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes

(HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The devices operate over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

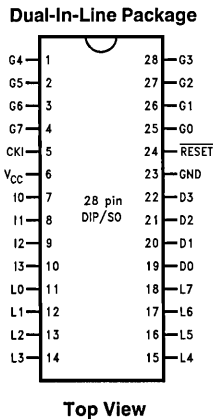
Connection Diagrams



Top View
Order Number COP888EG-XXX/V
See NS Plastic Chip Package Number V44A



Top View
Order Number COP888EG-XXX/N
See NS Molded Package Number N40A



Top View
Order Number COP884CG-XXX/WM or COP884EG-XXX/N
See NS Molded Package Number M28B or N28A

FIGURE 2a. COP888CG Connection Diagrams

Connection Diagrams (Continued)

COP888EG Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU	T3A	17	23	27
L7	I/O	MIWU	T3B	18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
I0	I			7	9	9
I1	I	COMP1IN-		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I	COMP2IN-			13	13
I5	I	COMP2IN+			14	14
I6	I	COMP2OUT			15	15
I7	I				16	16
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V _{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 888EG: -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			5.5	mA
CKI = 4 MHz	$V_{CC} = 3.5V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 3.5V, t_c = 10 \mu s$			1.4	mA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$ $V_{CC} = 3.5V, CKI = 0 MHz$		<1 <0.5	10 6	μA μA
Idle Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 3.5V, t_c = 10 \mu s$			0.7	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	0.4 0.2			mA mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	10 2.0			mA mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	10 2.5		100 33	μA μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$ $V_{CC} = 2.5V, V_{OH} = 1.8V$	0.4 0.2			mA mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$ $V_{CC} = 2.5V, V_{OL} = 0.4V$	1.6 0.7			mA mA
TRI-STATE Leakage		-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a crystal/resonator oscillator, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L, C, and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

DC Electrical Characteristics 888EG: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				15 3	mA mA
Maximum Input Current without Latchup	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics 888EG: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	1 2.5 3 7.5		DC DC DC DC	μs μs μs μs
CKI Clock Duty Cycle (Note 4) Rise Time (Note 4) Fall Time (Note 4)	$f_r = \text{Max}$ $f_r = 10\text{ MHz Ext Clock}$ $f_r = 10\text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	200 500 60 150			ns ns ns ns
Output Propagation Delay t_{PD1} , t_{PDO} SO, SK All Others	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$			0.7 1.75 1 2.5	μs μs μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Parameter sampled but not 100% tested.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 688EG: -55°C ≤ T_A ≤ +125°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu s$			5.5	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0$ MHz		<10	30	μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu s$			2.5	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-5		+5	μA
Input Pullup Current	$V_{CC} = 5.5V$	35		400	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.3V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1V$	9			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V, V_{OH} = 2.7V$	9		140	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.3V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.4			mA
TRI-STATE Leakage		-5		+5	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a crystal/resonator oscillator, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L, C, and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

DC Electrical Characteristics 688EG: $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				12 2.5	mA mA
Maximum Input Current without Latchup	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

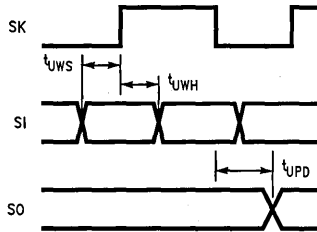
AC Electrical Characteristics 688EG: $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
CKI Clock Duty Cycle (Note 4) Rise Time (Note 5) Fall Time (Note 5)	$f_r = \text{Max}$ $f_r = 10 \text{ MHz Ext Clock}$ $f_r = 10 \text{ MHz Ext Clock}$	45		55 12 8	% ns ns
Inputs t_{SETUP} t_{HOLD}	$V_{CC} \geq 4.5\text{V}$ $V_{CC} \geq 4.5\text{V}$	200 60			ns ns
Output Propagation Delay $t_{\text{PD1}}, t_{\text{PD0}}$ SO, SK All Others	$R_L = 2.2\text{k}, C_L = 100 \text{ pF}$ $V_{CC} \geq 4.5\text{V}$ $V_{CC} \geq 4.5\text{V}$			0.7 1	μs μs
MICROWIRE Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Parameter sampled but not 100% tested.

Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25^\circ C$

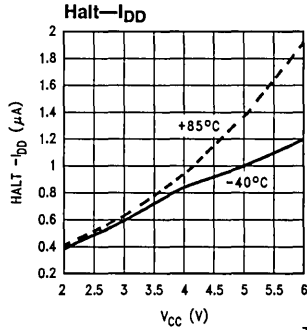
Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		± 10	± 25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current Per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



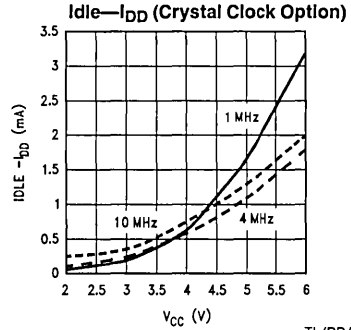
TL/DD/11214-5

FIGURE 2. MICROWIRE/PLUS Timing

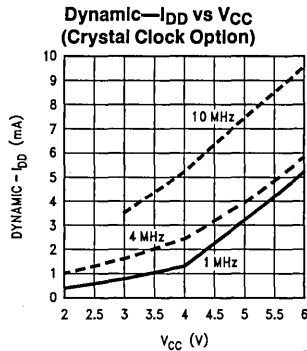
Typical Performance Characteristics ($-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$)



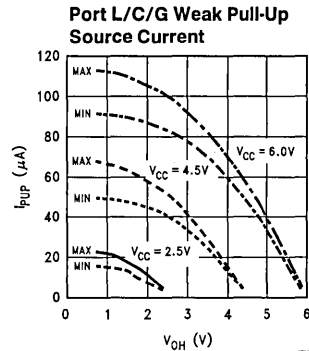
TL/DD/11214-7



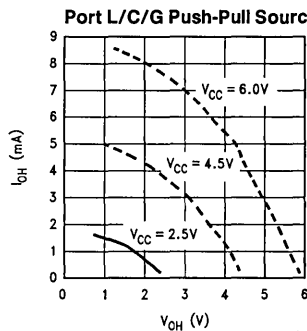
TL/DD/11214-8



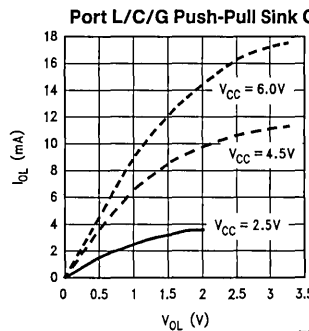
TL/DD/11214-9



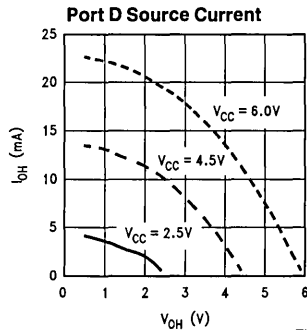
TL/DD/11214-10



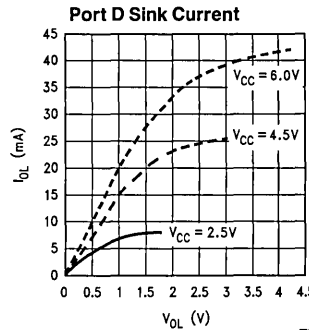
TL/DD/11214-11



TL/DD/11214-12



TL/DD/11214-13



TL/DD/11214-14

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The device contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports L and G), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the memory map for the various addresses associated with the I/O ports.) Figure 3 shows the I/O port configurations. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

L0	MIWU
L1	MIWU or CKX
L2	MIWU or TDx
L3	MIWU or RDX
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU or T3A
L7	MIWU or T3B

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

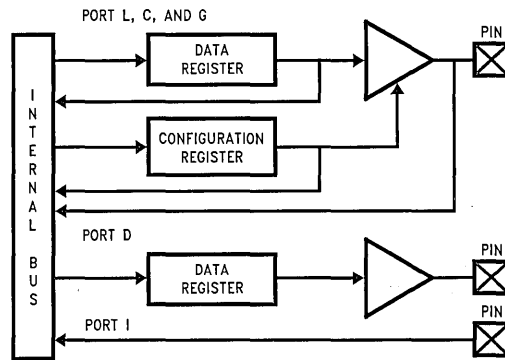


FIGURE 3. I/O Port Configurations

TL/DD/11214-6

Pin Descriptions (Continued)

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE™ Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT WATCHDOG and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

Port C is an 8-bit I/O port. The 40-pin device does not have a full complement of Port C pins. The unavailable pins are not terminated. A read operation for these unterminated pins will return unpredictable values.

PORT I is an eight-bit Hi-Z input port. The 28-pin device does not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

- I1 COMP1–IN (Comparator 1 Negative Input)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I3 COMP1OUT (Comparator 1 Output)
- I4 COMP2–IN (Comparator 2 Negative Input)
- I5 COMP2+IN (Comparator 2 Positive Input)
- I6 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the device is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_c) cycle time.

There are six CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

S is the 8-bit Data Segment Address Register used to extend the lower half of the address range (00 to 7F) into 256 data segments of 128 bytes each.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

The program memory consists of 8092 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the devices vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X, SP pointers and S register.

The data memory consists of 256 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, B and S are memory mapped into this space at address locations 0FC to 0FF Hex respectively, with the other registers being available for general usage. The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Data Memory Segment RAM Extension

Data memory address 0FF is used as a memory mapped location for the Data Segment Address Register (S).

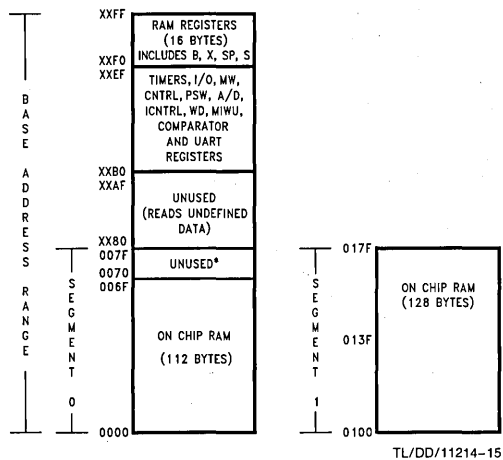
The data store memory is either addressed directly by a single byte address within the instruction, or indirectly relative to the reference of the B, X, or SP pointers (each contains a single-byte address). This single-byte address allows an addressing range of 256 locations from 00 to FF hex. The upper bit of this single-byte address divides the data store memory into two separate sections as outlined previously. With the exception of the RAM register memory from address locations 00F0 to 00FF, all RAM memory is memory mapped with the upper bit of the single-byte address being equal to zero. This allows the upper bit of the single-byte address to determine whether or not the base address range (from 0000 to 00FF) is extended. If this upper bit equals one (representing address range 0080 to 00FF), then address extension does not take place. Alternatively, if this upper bit equals zero, then the data segment extension register S is used to extend the base address range (from 0000 to 007F) from XX00 to XX7F, where XX represents the 8 bits from the S register. Thus the 128-byte data segment extensions are located from addresses 0100 to 017F for data segment 1, 0200 to 027F for data segment 2, etc., up to FF00 to FF7F for data segment 255. The base address range from 0000 to 007F represents data segment 0.

Figure 4 illustrates how the S register data memory extension is used in extending the lower half of the base address range (00 to 7F hex) into 256 data segments of 128 bytes each, with a total addressing range of 32 kbytes from XX00 to XX7F. This organization allows a total of 256 data segments of 128 bytes each with an additional upper base segment of 128 bytes. Furthermore, all addressing modes are available for all data segments. The S register must be changed under program control to move from one data segment (128 bytes) to another. However, the upper base segment (containing the 16 memory registers, I/O registers, control registers, etc.) is always available regardless of the contents of the S register, since the upper base segment (address range 0080 to 00FF) is independent of data segment extension.

The instructions that utilize the stack pointer (SP) always reference the stack as part of the base segment (Segment 0), regardless of the contents of the S register. The S register is not changed by these instructions. Consequently, the stack (used with subroutine linkage and interrupts) is always located in the base segment. The stack pointer will be initialized to point at data memory location 006F as a result of reset.

The 128 bytes of RAM contained in the base segment are split between the lower and upper base segments. The first 116 bytes of RAM are resident from address 0000 to 006F in the lower base segment, while the remaining 16 bytes of RAM represent the 16 data memory registers located at addresses 00F0 to 00FF of the upper base segment. No RAM is located at the upper sixteen addresses (0070 to 007F) of the lower base segment.

Additional RAM beyond these initial 128 bytes, however, will always be memory mapped in groups of 128 bytes (or less) at the data segment address extensions (XX00 to XX7F) of the lower base segment. The additional 128 bytes of RAM are memory mapped at address locations 0100 to 017F hex.



*Reads as all ones.

FIGURE 4. RAM Organization

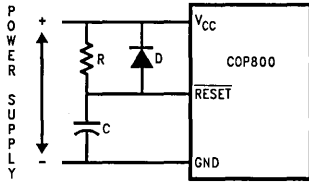
Reset

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the data and configuration registers for ports L, G and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WATCHDOG and/or Clock Monitor error output pin. Port D is set high. The PC, PSW, ICNTRL, CNTRL, T2CNTRL and T3CNTRL control registers are cleared. The UART registers PSR, ENU (except that TBMT bit is set), ENUR and ENUI are cleared. The Comparator Select Register is cleared. The S register is initialized to zero. The Multi-Input Wakeup registers WKEN, WKEDG and WKPND are cleared. The stack pointer, SP, is initialized to 6F Hex.

The device comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, with the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor circuits are inhibited during reset. The WATCHDOG service window bits being initialized high default to the maximum WATCHDOG service window of $64k t_C$ clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until $16 t_C - 32 t_C$ clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in Figure 5 should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.

Reset (Continued)



TL/DD/11214-16

$RC > 5 \times$ Power Supply Rise Time

FIGURE 5. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 6 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

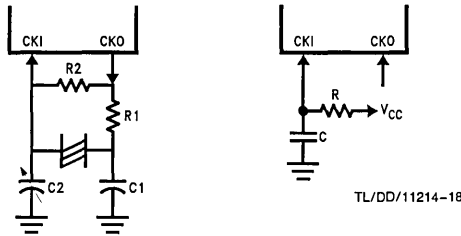
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/11214-18

TL/DD/11214-17

FIGURE 6. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4	$V_{CC} = 5V$
0	1	200	100-150	0.455	$V_{CC} = 5V$

TABLE B. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Note: $3k \leq R \leq 200k$

$50 \text{ pF} \leq C \leq 200 \text{ pF}$

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. Comparator DC supply current when enabled—I6
7. Clock Monitor current when enabled—I7

Thus the total current drain, I_t is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

where C = equivalent capacitance of the chip

V = operating voltage

f = CKI frequency

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
- T1C0 Timer T1 Start/Stop control in timer modes 1 and 2
Timer T1 Underflow Interrupt Pending Flag in timer mode 3
- T1C1 Timer T1 mode control bit
- T1C2 Timer T1 mode control bit
- T1C3 Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
------	------	------	------	------	------	-----	-----

Bit 7

Bit 0

Control Registers (Continued)

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt
BUSY	MICROWIRE/PLUS busy shifting flag
EXPND	External interrupt pending
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
T1PND	Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
C	Carry Flag
HC	Half Carry Flag

HC	C	T1PND	T1ENA	EXPND	BUSY	EXEN	GIE
----	---	-------	-------	-------	------	------	-----

Bit 7

Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB	Timer T1 Interrupt Enable for T1B Input capture edge
T1PNDB	Timer T1 Interrupt Pending Flag for T1B capture edge
μ WEN	Enable MICROWIRE/PLUS interrupt
μ WPND	MICROWIRE/PLUS interrupt pending
T0EN	Timer T0 Interrupt Enable (Bit 12 toggle)
T0PND	Timer T0 Interrupt pending
LPEN	L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)
Bit 7 could be used as a flag	

Unused	LPEN	T0PND	T0EN	μ WPND	μ WEN	T1PNDB	T1ENB
--------	------	-------	------	------------	-----------	--------	-------

Bit 7

Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB	Timer T2 Interrupt Enable for T2B Input capture edge
T2PNDB	Timer T2 Interrupt Pending Flag for T2B capture edge
T2ENA	Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
T2PND	Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
T2C0	Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

T2C1	Timer T2 mode control bit
T2C2	Timer T2 mode control bit
T2C3	Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PND	T2ENA	T2PNDB	T2ENB
------	------	------	------	-------	-------	--------	-------

Bit 7

Bit 0

T3CNTRL Register (Address X'00B6)

The T3CNTRL register contains the following bits:

T3ENB	Timer T3 Interrupt Enable for T3B
T3PNDB	Timer T3 Interrupt Pending Flag for T3B pin (T3B capture edge)
T3ENA	Timer T3 Interrupt Enable for Timer Underflow or T3A pin
T3PND	Timer T3 Interrupt Pending Flag (Autoload RA in mode 1, T3 Underflow in mode 2, T3a capture edge in mode 3)
T3C0	Timer T3 Start/Stop control in timer modes 1 and 2 Timer T3 Underflow Interrupt Pending Flag in timer mode 3
T3C1	Timer T3 mode control bit
T3C2	Timer T3 mode control bit
T3C3	Timer T3 mode control bit

T3C3	T3C2	T3C1	T3C0	T3PND	T3ENA	T3PNDB	T3ENB
------	------	------	------	-------	-------	--------	-------

Bit 7

Bit 0

Timers

The device contains a very versatile set of timers (T0, T1, T2, T3). All timers and associated autoreload/capture registers power up containing random data.

TIMER T0 (IDLE TIMER)

The devices support applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c . The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)

WATCHDOG logic (See WATCHDOG description)

Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_c = 1 \mu s$). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

Timers (Continued)

TIMER T1, TIMER T2 AND TIMER T3

The devices have a set of three powerful timer/counter blocks, T1, T2 and T3. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the three timer blocks, T1, T2 and T3 are identical, all comments are equally applicable to any of the three timer blocks.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the device to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the device to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_c . Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 7 shows a block diagram of the timer in PWM mode. The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPND A and TxPND B. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

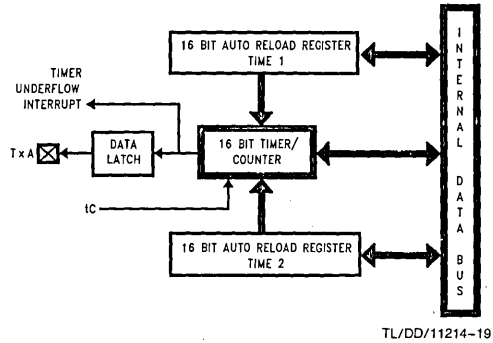


FIGURE 7. Timer in PWM Mode

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPND A pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPND B flag.

Figure 8 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

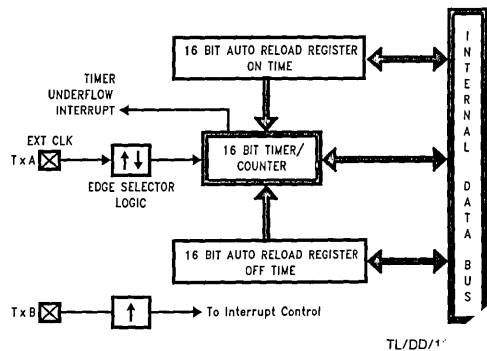


FIGURE 8. Timer in External Event Counter Mode

Mode 3. Input Capture Mode

The device can precisely measure external time external events by placing the timer in input capture mode.

In this mode, the timer Tx is clocked at the t_c rate. The two registers, RxA and RxB, are used to capture the time of an event. Each register, RxA and RxB, acts as an input capture register. RxA acts as the rising edge capture register and RxB acts as the falling edge capture register.

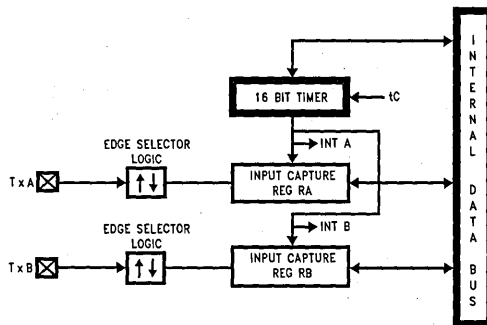
Timers (Continued)

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPND A and TxPND B. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxCO pending flag (the TxCO control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxCO control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPND A and TxCO pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 9 shows a block diagram of the timer in Input Capture mode.



TL/DD/11214-21

FIGURE 9. Timer in Input Capture Mode

TIMER CONTROL FLAGS

The timers T1, T2 and T3 have identical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPND A	Timer Interrupt Pending Flag
TxPND B	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c

Power Save Modes

The devices offer the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry the WATCHDOG logic, the Clock Monitor and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The devices can be placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock and timers, are stopped. The WATCHDOG logic on the device is disabled during the HALT mode. However, the clock monitor circuitry if enabled remains active and will cause the WATCHDOG output pin (WDOUT) to go low. If the HALT mode is used and the user does not want to activate the WDOUT pin, the Clock Monitor should be disabled after the device comes out of reset (resetting the Clock Monitor control bit with the first write to the WDSVR register). In the HALT mode, the power requirements of the device are minimal and the applied voltage (V_{CC}) may be decreased to V_r ($V_r = 2.0V$) without altering the state of the machine.

The devices support three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock con-

figuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

Power Save Modes (Continued)

The devices have two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the device will enter and exit the HALT mode as described above. With the HALT disable mask option, the device cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WATCHDOG detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the device inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The device is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activities, except the associated on-board oscillator circuitry, the WATCHDOG logic, the clock monitor and the IDLE Timer T0, are stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the device can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_c = 1 \mu s$) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the TOPND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the TOPND bit gets set, the device will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the device will resume normal operation with the instruction immediately following the "Enter Idle Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the device from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 10 shows the Multi-Input Wakeup logic.

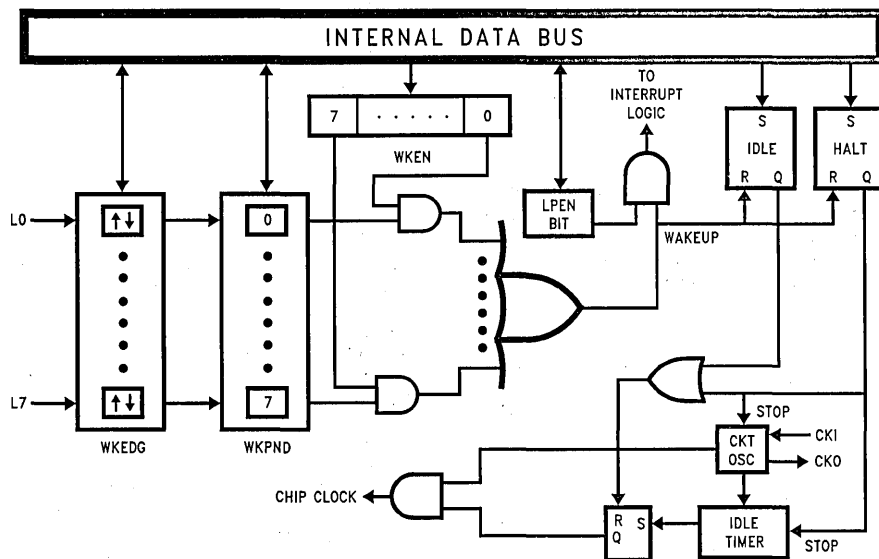


FIGURE 10. Multi-Input Wake Up Logic

TL/DD/11214-22

Multi-Input Wakeup (Continued)

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the device to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```

RBIT 5, WKEN
SBIT 5, WKEDG
RBIT 5, WKPND
SBIT 5, WKEN

```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the device will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the device out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the device will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the device will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T₀) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the device to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T₀ are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

UART

The device contains a full-duplex software programmable UART. The UART (*Figure 11*) consists of a transmit shift register, a receiver shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR), a UART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framing, data overrun and parity errors while the UART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the UART's attention mode of operation and providing additional receiver/transmitter status information via RCVG and XMTG bits. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts. A control flag in this register can also select the UART mode of operation: asynchronous or synchronous.

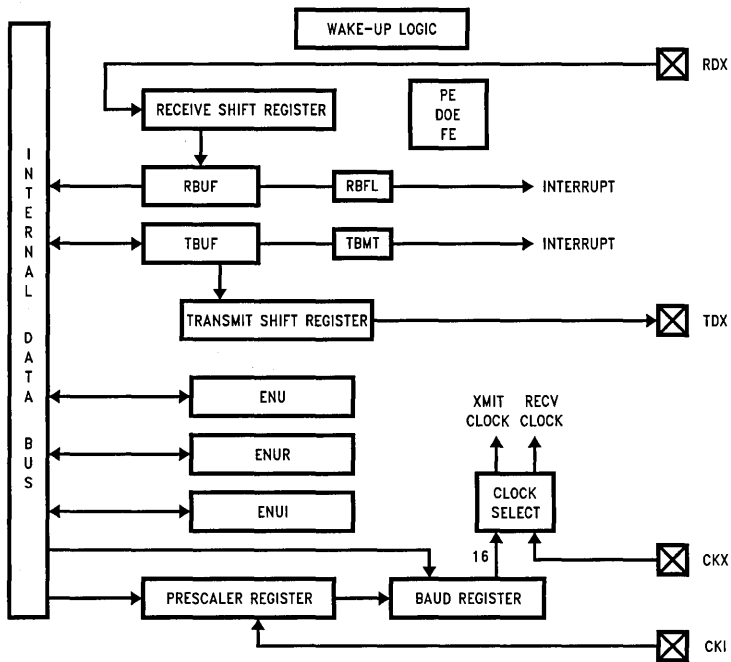


FIGURE 11. UART Block Diagram

TL/DD/11214-23

UART (Continued)

UART CONTROL AND STATUS REGISTERS

The operation of the UART is programmed through three registers: ENU, ENUR and ENUI. The function of the individual bits in these registers is as follows:

ENU-UART Control and Status Register (Address at 0BA)

PEN	PSEL1	XBIT9/ PSEL0	CHL1	CHL0	ERR	RBFL	TBMT
ORW	ORW	ORW	ORW	ORW	OR	OR	1R

Bit 7 Bit 0

ENUR-UART Receive Control and Status Register
(Address at 0BB)

DOE	FE	PE	SPARE	RBIT9	ATTN	XMTG	RCVG
ORD	ORD	ORD	ORW*	OR	ORW	OR	OR

Bit7 Bit0

ENUI-UART Interrupt and Clock Source Register
(Address at 0BC)

STP2	STP78	ETDX	SSEL	XRCLK	XTCLK	ERI	ETI
ORW	ORW	ORW	ORW	ORW	ORW	ORW	ORW

Bit7 Bit0

*Bit is not used.

- 0 Bit is cleared on reset.
- 1 Bit is set to one on reset.
- R Bit is read-only; it cannot be written by software.
- RW Bit is read/write.
- D Bit is cleared on read; when read by software as a one, it is cleared automatically. Writing to the bit does not affect its state.

DESCRIPTION OF UART REGISTER BITS

ENU—UART CONTROL AND STATUS REGISTER

TBMT: This bit is set when the UART transfers a byte of data from the TBUF register into the TSFT register for transmission. It is automatically reset when software writes into the TBUF register.

RBFL: This bit is set when the UART has received a complete character and has copied it into the RBUF register. It is automatically reset when software reads the character from RBUF.

ERR: This bit is a global UART error flag which gets set if any or a combination of the errors (DOE, FE, PE) occur.

CHL1, CHL0: These bits select the character frame format. Parity is not included and is generated/verified by hardware.
 CHL1 = 0, CHL0 = 0 The frame contains eight data bits.
 CHL1 = 0, CHL0 = 1 The frame contains seven data bits.

CHL1 = 1, CHL0 = 0 The frame contains nine data bits.
 CHL1 = 1, CHL0 = 1 Loopback Mode selected. Transmitter output internally looped back to receiver input. Nine bit framing format is used.

XBIT9/PSEL0: Programs the ninth bit for transmission when the UART is operating with nine data bits per frame. For seven or eight data bits per frame, this bit in conjunction with PSEL1 selects parity.

PSEL1, PSEL0: Parity select bits.

- PSEL1 = 0, PSEL0 = 0 Odd Parity (if Parity enabled)
- PSEL1 = 0, PSEL0 = 1 Even Parity (if Parity enabled)

- PSEL1 = 1, PSEL0 = 0 Mark(1) (if Parity enabled)
- PSEL1 = 1, PSEL0 = 1 Space(0) (if Parity enabled)

PEN: This bit enables/disables Parity (7- and 8-bit modes only).

- PEN = 0 Parity disabled.
- PEN = 1 Parity enabled.

ENUR—UART RECEIVE CONTROL AND STATUS REGISTER

RCVG: This bit is set high whenever a framing error occurs and goes low when RDX goes high.

XMTG: This bit is set to indicate that the UART is transmitting. It gets reset at the end of the last frame (end of last Stop bit).

ATTN: ATTENTION Mode is enabled while this bit is set. This bit is cleared automatically on receiving a character with data bit nine set.

RBIT9: Contains the ninth data bit received when the UART is operating with nine data bits per frame.

SPARE: Reserved for future use.

PE: Flags a Parity Error.

- PE = 0 Indicates no Parity Error has been detected since the last time the ENUR register was read.
- PE = 1 Indicates the occurrence of a Parity Error.

FE: Flags a Framing Error.

- FE = 0 Indicates no Framing Error has been detected since the last time the ENUR register was read.
- FE = 1 Indicates the occurrence of a Framing Error.

DOE: Flags a Data Overrun Error.

- DOE = 0 Indicates no Data Overrun Error has been detected since the last time the ENUR register was read.
- DOE = 1 Indicates the occurrence of a Data Overrun Error.

ENUI—UART INTERRUPT AND CLOCK SOURCE REGISTER

ETI: This bit enables/disables interrupt from the transmitter section.

- ETI = 0 Interrupt from the transmitter is disabled.
- ETI = 1 Interrupt from the transmitter is enabled.

ERI: This bit enables/disables interrupt from the receiver section.

- ERI = 0 Interrupt from the receiver is disabled.
- ERI = 1 Interrupt from the receiver is enabled.

XTCLK: This bit selects the clock source for the transmitter section.

- XTCLK = 0 The clock source is selected through the PSR and BAUD registers.
- XTCLK = 1 Signal on CKX (L1) pin is used as the clock.

XRCLK: This bit selects the clock source for the receiver section.

- XRCLK = 0 The clock source is selected through the PSR and BAUD registers.
- XRCLK = 1 Signal on CKX (L1) pin is used as the clock.

SSEL: UART mode select.

- SSEL = 0 Asynchronous Mode.
- SSEL = 1 Synchronous Mode.

UART (Continued)

ETDX: TDX (UART Transmit Pin) is the alternate function assigned to Port L pin L2; it is selected by setting ETDX bit. To simulate line break generation, software should reset ETDX bit and output logic zero to TDX pin through Port L data and configuration registers.

STP78: This bit is set to program the last Stop bit to be 7/8th of a bit in length.

STP2: This bit programs the number of Stop bits to be transmitted.

STP2 = 0 One Stop bit transmitted.

STP2 = 1 Two Stop bits transmitted.

Associated I/O Pins

Data is transmitted on the TDX pin and received on the RDX pin. TDX is the alternate function assigned to Port L pin L2; it is selected by setting ETDX (in the ENUI register) to one. RDX is an inherent function of Port L pin L3, requiring no setup.

The baud rate clock for the UART can be generated on-chip, or can be taken from an external source. Port L pin L1 (CKX) is the external clock I/O pin. The CKX pin can be either an input or an output, as determined by Port L Configuration and Data registers (Bit 1). As an input, it accepts a clock signal which may be selected to drive the transmitter and/or receiver. As an output, it presents the internal Baud Rate Generator output.

UART Operation

The UART has two modes of operation: asynchronous mode and synchronous mode.

ASYNCHRONOUS MODE

This mode is selected by resetting the SSEL (in the ENUI register) bit to zero. The input frequency to the UART is 16 times the baud rate.

The TSFT and TBUF registers double-buffer data for transmission. While TSFT is shifting out the current character on the TDX pin, the TBUF register may be loaded by software with the next byte to be transmitted. When TSFT finishes transmitting the current character the contents of TBUF are transferred to the TSFT register and the Transmit Buffer Empty Flag (TBMT in the ENU register) is set. The TBMT flag is automatically reset by the UART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the UART is transmitting. This bit gets reset at the end of the last frame (end of last Stop bit). TBUF is a read/write register.

The RSFT and RBUF registers double-buffer data being received. The UART receiver continually monitors the signal on the RDX pin for a low level to detect the beginning of a Start bit. Upon sensing this low level, it waits for half a bit time and samples again. If the RDX pin is still low, the receiver considers this to be a valid Start bit, and the remaining bits in the character frame are each sampled a single time, at the mid-bit position. Serial data input on the RDX pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the RBUF register and the Received Buffer Full Flag (RBFL) is set. RBFL is automatically reset when software reads the character from the RBUF register. RBUF is a read only register. There is also the RCVG bit which is set high

when a framing error occurs and goes low once RDX goes high. TBMT, XMTG, RBFL and RCVG are read only bits.

SYNCHRONOUS MODE

In this mode data is transferred synchronously with the clock. Data is transmitted on the rising edge and received on the falling edge of the synchronous clock.

This mode is selected by setting SSEL bit in the ENUI register. The input frequency to the UART is the same as the baud rate.

When an external clock input is selected at the CKX pin, data transmit and receive are performed synchronously with this clock through TDX/RDX pins.

If data transmit and receive are selected with the CKX pin as clock output, the device generates the synchronous clock output at the CKX pin. The internal baud rate generator is used to produce the synchronous clock. Data transmit and receive are performed synchronously with this clock.

FRAMING FORMATS

The UART supports several serial framing formats (*Figure 12*). The format is selected using control bits in the ENU, ENUR and ENUI registers.

The first format (1, 1a, 1b, 1c) for data transmission (CHL0 = 1, CHL1 = 0) consists of Start bit, seven Data bits (excluding parity) and 7/8, one or two Stop bits. In applications using parity, the parity bit is generated and verified by hardware.

The second format (CHL0 = 0, CHL1 = 0) consists of one Start bit, eight Data bits (excluding parity) and 7/8, one or two Stop bits. Parity bit is generated and verified by hardware.

The third format for transmission (CHL0 = 0, CHL1 = 1) consists of one Start bit, nine Data bits and 7/8, one or two Stop bits. This format also supports the UART "ATTENTION" feature. When operating in this format, all eight bits of TBUF and RBUF are used for data. The ninth data bit is transmitted and received using two bits in the ENU and ENUR registers, called XBIT9 and RBIT9. RBIT9 is a read only bit. Parity is not generated or verified in this mode.

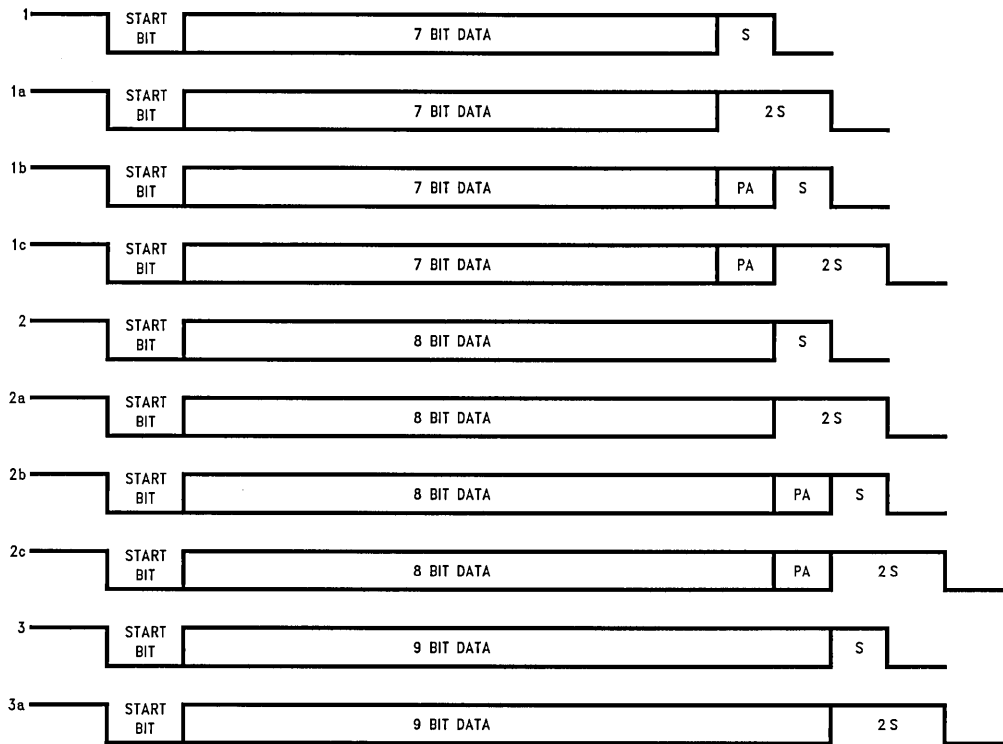
For any of the above framing formats, the last Stop bit can be programmed to be 7/8th of a bit in length. If two Stop bits are selected and the 7/8th bit is set (selected), the second Stop bit will be 7/8th of a bit in length.

The parity is enabled/disabled by PEN bit located in the ENU register. Parity is selected for 7- and 8-bit modes only. If parity is enabled (PEN = 1), the parity selection is then performed by PSEL0 and PSEL1 bits located in the ENU register.

Note that the XBIT9/PSEL0 bit located in the ENU register serves two mutually exclusive functions. This bit programs the ninth bit for transmission when the UART is operating with nine data bits per frame. There is no parity selection in this framing format. For other framing formats XBIT9 is not needed and the bit is PSEL0 used in conjunction with PSEL1 to select parity.

The frame formats for the receiver differ from the transmitter in the number of Stop bits required. The receiver only requires one Stop bit in a frame, regardless of the setting of the Stop bit selection bits in the control register. Note that an implicit assumption is made for full duplex UART operation that the framing formats are the same for the transmitter and receiver.

UART Operation (Continued)



TL/DD/11214-24

FIGURE 12. Framing Formats

UART INTERRUPTS

The UART is capable of generating interrupts. Interrupts are generated on Receive Buffer Full and Transmit Buffer Empty. Both interrupts have individual interrupt vectors. Two bytes of program memory space are reserved for each interrupt vector. The two vectors are located at addresses 0xEC to 0xEF Hex in the program memory space. The interrupts can be individually enabled or disabled using Enable Transmit Interrupt (ETI) and Enable Receive Interrupt (ERI) bits in the ENUI register.

The interrupt from the Transmitter is set pending, and remains pending, as long as both the TBMT and ETI bits are set. To remove this interrupt, software must either clear the ETI bit or write to the TBUF register (thus clearing the TBMT bit).

The interrupt from the receiver is set pending, and remains pending, as long as both the RBFL and ERI bits are set. To remove this interrupt, software must either clear the ERI bit or read from the RBUF register (thus clearing the RBFL bit).

Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the UART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a

source selected in the PSR and BAUD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1-16 (increments of 0.5) prescaler and an 11-bit binary counter. (Figure 13) The divide factors are specified through two read/write registers shown in Figure 14. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

As shown in Table I, a Prescaler Factor of 0 corresponds to NO CLOCK. NO CLOCK condition is the UART power down mode where the UART clock is turned off for power saving purpose. The user must also turn the UART clock off when a different baud rate is chosen.

The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in Table I. There are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a x16 clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (Table II). Other baud rates may be created by using appropriate divisors. The x16 clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.

Baud Clock Generation (Continued)

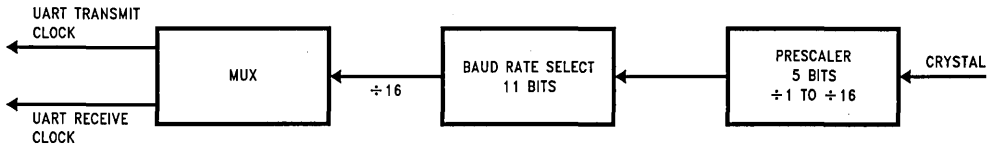


FIGURE 13. UART BAUD Clock Generation

TL/DD/11214-25

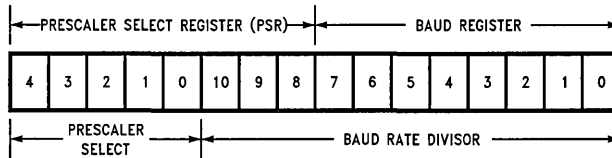


FIGURE 14. UART BAUD Clock Divisor Registers

TL/DD/11214-26

TABLE I. Prescaler Factors

Prescaler Select	Prescaler Factor
00000	NO CLOCK
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11001	13
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

TABLE II. Baud Rate Divisors
(1.8432 MHz Prescaler Output)

Baud Rate	Baud Rate Divisor - 1 (N-1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5
38400	2

The entries in Table II assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 625k.

As an example, considering the Asynchronous Mode and a CKI clock of 4.608 MHz, the prescaler factor selected is:

$$4.608 / 1.8432 = 2.5$$

The 2.5 entry is available in Table I. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (Table II) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in Table II is 5.

$$N - 1 = 5 \text{ (N - 1 is the value from Table II)}$$

$$N = 6 \text{ (N is the Baud Rate Divisor)}$$

$$\text{Baud Rate} = 1.8432 \text{ MHz} / (16 \times 6) = 19200$$

The divide by 16 is performed because in the asynchronous mode, the input frequency to the UART is 16 times the baud rate. The equation to calculate baud rates is given below.

The actual Baud Rate may be found from:

$$BR = Fc / (16 \times N \times P)$$

Baud Clock Generation (Continued)

Where:

BR is the Baud Rate

Fc is the CKI frequency

N is the Baud Rate Divisor (Table II).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (Table I)

Note: In the Synchronous Mode, the divisor 16 is replaced by two.

Example:

Asynchronous Mode:

$$\text{Crystal Frequency} = 5 \text{ MHz}$$

$$\text{Desired baud rate} = 9600$$

Using the above equation $N \times P$ can be calculated first.

$$N \times P = (5 \times 10^6) / (16 \times 9600) = 32.552$$

Now 32.552 is divided by each Prescaler Factor (Table II) to obtain a value closest to an integer. This factor happens to be 6.5 ($P = 6.5$).

$$N = 32.552 / 6.5 = 5.008 \quad (N = 5)$$

The programmed value (from Table II) should be 4 ($N - 1$).

Using the above values calculated for N and P:

$$BR = (5 \times 10^6) / (16 \times 5 \times 6.5) = 9615.384$$

$$\% \text{ error} = (9615.385 - 9600) / 9600 = 0.16$$

Effect of HALT/IDLE

The UART logic is reinitialized when either the HALT or IDLE modes are entered. This reinitialization sets the TBMT flag and resets all read only bits in the UART control and status registers. Read/Write bits remain unchanged. The Transmit Buffer (TBUF) is not affected, but the Transmit Shift register (TSFT) bits are set to one. The receiver registers RBUF and RSFT are not affected.

The device will exit from the HALT/IDLE modes when the Start bit of a character is detected at the RDX (L3) pin. This feature is obtained by using the Multi-Input Wakeup scheme provided on the device.

Before entering the HALT or IDLE modes the user program must select the Wakeup source to be on the RDX pin. This selection is done by setting bit 3 of WKEN (Wakeup Enable) register. The Wakeup trigger condition is then selected to be high to low transition. This is done via the WKEDG register (Bit 3 is zero.)

If the device is halted and crystal oscillator is used, the Wakeup signal will not start the chip running immediately because of the finite start up time requirement of the crystal oscillator. The idle timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the device to execute code. The user has to consider this delay when data transfer is expected immediately after exiting the HALT mode.

Diagnostic

Bits CHARL0 and CHARL1 in the ENU register provide a loopback feature for diagnostic testing of the UART. When these bits are set to one, the following occur: The receiver input pin (RDX) is internally connected to the transmitter output pin (TDX); the output of the Transmitter Shift Register is "looped back" into the Receive Shift Register input. In this mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the UART.

Note that the framing format for this mode is the nine bit format; one Start bit, nine data bits, and 7/8, one or two Stop bits. Parity is not generated or verified in this mode.

Attention Mode

The UART Receiver section supports an alternate mode of operation, referred to as ATTENTION Mode. This mode of operation is selected by the ATTN bit in the ENUR register. The data format for transmission must also be selected as having nine Data bits and either 7/8, one or two Stop bits.

The ATTENTION mode of operation is intended for use in networking the device with other processors. Typically in such environments the messages consists of device addresses, indicating which of several destinations should receive them, and the actual data. This Mode supports a scheme in which addresses are flagged by having the ninth bit of the data field set to a 1. If the ninth bit is reset to a zero the byte is a Data byte.

While in ATTENTION mode, the UART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the UART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if UART Receiver interrupts are enabled. The ATTN bit is also cleared automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

Operation of the UART Transmitter is not affected by selection of this Mode. The value of the ninth bit to be transmitted is programmed by setting XBIT9 appropriately. The value of the ninth bit received is obtained by reading RBIT9. Since this bit is located in ENUR register where the error flags reside, a bit operation on it will reset the error flags.

Comparators

The devices contain two differential comparators, each with a pair of inputs (positive and negative) and an output. Ports I1-I3 and I4-I6 are used for the comparators. The following is the Port I assignment:

- I1 Comparator1 negative input
- I2 Comparator1 positive input
- I3 Comparator1 output
- I4 Comparator2 negative input
- I5 Comparator2 positive input
- I6 Comparator2 output

A Comparator Select Register (CMPSL) is used to enable the comparators, read the outputs of the comparators internally, and enable the outputs of the comparators to the pins. Two control bits (enable and output enable) and one result bit are associated with each comparator. The comparator result bits (CMP1RD and CMP2RD) are read only bits which will read as zero if the associated comparator is not enabled. The Comparator Select Register is cleared with reset, resulting in the comparators being disabled. The comparators should also be disabled before entering either the HALT or IDLE modes in order to save power. The configuration of the CMPSL register is as follows:

Comparators (Continued)

CMPSL REGISTER (ADDRESS X'00B7)

The CMPSL register contains the following bits:

- CMP1EN Enable comparator 1
- CMP1RD Comparator 1 result (this is a read only bit, which will read as 0 if the comparator is not enabled)
- CMP10E Selects pin I3 as comparator 1 output provided that CMPIEN is set to enable the comparator
- CMP2EN Enable comparator 2
- CMP2RD Comparator 2 result (this is a read only bit, which will read as 0 if the comparator is not enabled)
- CMP20E Selects pin I6 as comparator 2 output provided that CMP2EN is set to enable the comparator

Unused	CMP20E	CMP2RD	CMP2EN	CMP10E	CMP1RD	CMP1EN	Unused
--------	--------	--------	--------	--------	--------	--------	--------

Bit 7

Bit 0

Note that the two unused bits of CMPSL may be used as software flags.

Comparator outputs have the same spec as Ports L and G except that the rise and fall times are symmetrical.

Interrupts

The devices support a vectored interrupt scheme. It supports a total of fourteen interrupt sources. The following table lists all the possible device interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_c cycles to execute.

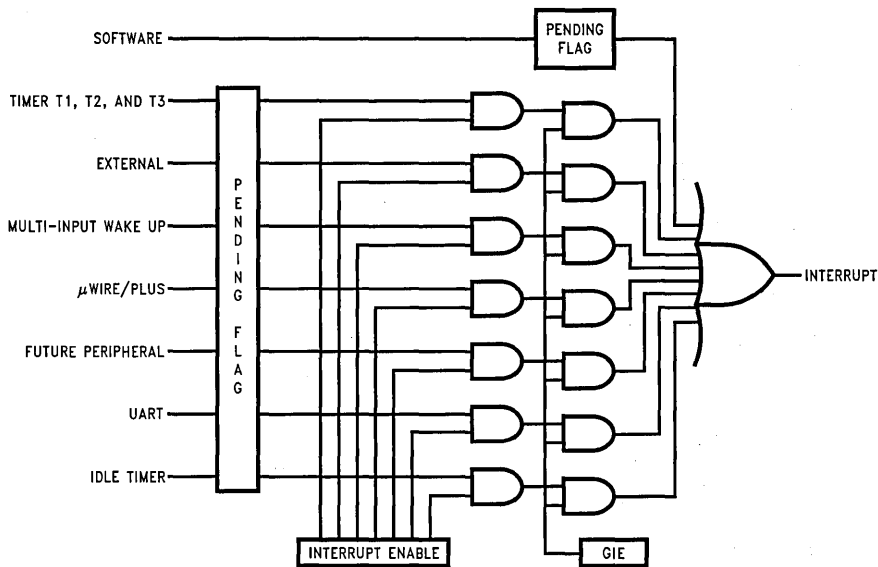


FIGURE 15. Interrupt Block Diagram

TL/DD/11214-27

Interrupts (Continued)

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	Underflow	0yF8–0yF9
(4)	Timer T1	T1A/Underflow	0yF6–0yF7
(5)	Timer T1	T1B	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
(7)	UART	Receive	0yEE–0yEF
(8)	UART	Transmit	0yEC–0yED
(9)	Timer T2	T2A/Underflow	0yEA–0yEB
(10)	Timer T2	T2B	0yE8–0yE9
(11)	Timer T3	T3A/Underflow	0yE6–0yE7
(12)	Timer T3	T3B	0yE4–0yE5
(13)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(14) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page, y ≠ 0.

At this time, since $GIE = 0$, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block (y ≠ 0).

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 15 shows the Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

Interrupts (Continued)

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit (**not accessible by the user**) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WATCHDOG

The devices contain a WATCHDOG and clock monitor. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WATCHDOG consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WATCHDOG consists of writing a specific value to a WATCHDOG Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table III shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table IV shows the four possible combinations of lower and upper limits for the WATCHDOG service window. This flexibility in choosing the WATCHDOG service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE III. WATCHDOG Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE IV. WATCHDOG Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k–8k t_c Cycles
0	1	2k–16k t_c Cycles
1	0	2k–32k t_c Cycles
1	1	2k–64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the device can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WATCHDOG Operation

The WATCHDOG and Clock Monitor are disabled during reset. The device comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCHDOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. Table V shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WATCHDOG, the logic will pull the WDOUT (G1) pin low for an additional 16 t_c –32 t_c cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the device will stop forcing the WDOUT output low.

The WATCHDOG service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WATCHDOG service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WATCHDOG will time out and WDOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_c –32 t_c clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10$ kHz—No clock rejection.

$1/t_c < 10$ Hz—Guaranteed clock rejection.

WATCHDOG Operation (Continued)

WATCHDOG AND CLOCK MONITOR SUMMARY

The following salient points regarding the COP888EG WATCHDOG and CLOCK MONITOR should be noted:

- Both the WATCHDOG and CLOCK MONITOR detector circuits are inhibited during RESET.
- Following RESET, the WATCHDOG and CLOCK MONITOR are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and CLOCK MONITOR enable/disable option can only be changed once, during the initial WATCHDOG service following RESET.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.
- The CLOCK MONITOR detector circuit is active during both the HALT and IDLE modes. Consequently, the COP888 inadvertently entering the HALT mode will be detected as a CLOCK MONITOR error (provided that the CLOCK MONITOR enable option has been selected by the program).
- With the single-pin R/C oscillator mask option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.
- With the crystal oscillator mask option selected, or with the single-pin R/C oscillator mask option selected and the CLKDLY bit set, the WATCHDOG service window will be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.
- The IDLE timer T0 is not initialized with RESET.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the TOPND flag. The TOPND flag is set whenever the thirteenth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the TOPND flag.
- A hardware WATCHDOG service occurs just as the device exits the IDLE mode. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following RESET, the initial WATCHDOG service (where the service window and the CLOCK MONITOR enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 2048 instruction cycles without causing a WATCHDOG error.

Detection of Illegal Conditions

The device can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F (Segment 0), 140 to 17F (Segment 1), and all other segments (i.e., Segments 3 . . . etc.) is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

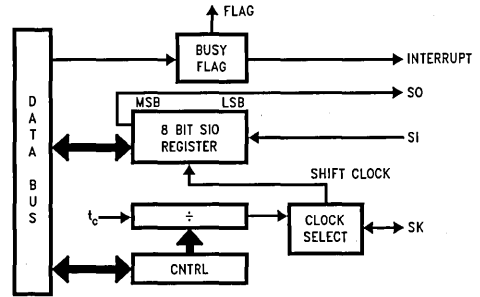
Thus, the chip can detect the following illegal conditions:

- a. Executing from undefined ROM
- b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before re-starting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the device to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E²PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 12 shows a block diagram of the MICROWIRE/PLUS logic.



TL/DD/11214-28

FIGURE 16. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table VI details the different clock rates that may be selected.

TABLE V. WATCHDOG Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

TABLE VI. MICROWIRE/PLUS Master Mode Clock Select

SL1	SL0	SK
0	0	$2 \times t_c$
0	1	$4 \times t_c$
1	x	$8 \times t_c$

Where t_c is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The device may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. Figure 13 shows how two devices, microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock for the SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the device. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table VII summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table VII summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

The device allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE VII

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave

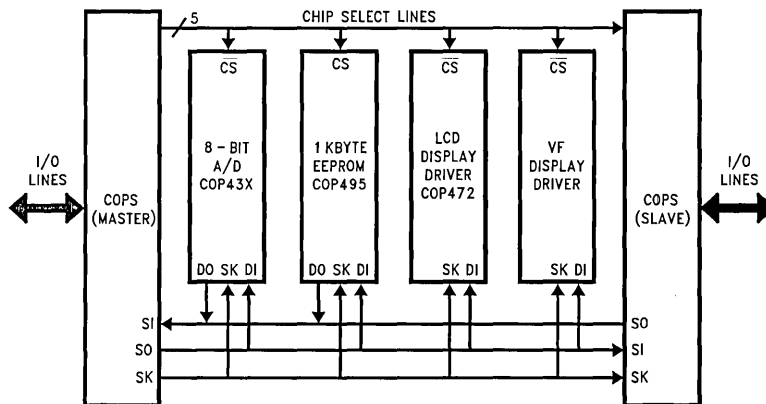


FIGURE 17. MICROWIRE/PLUS Application

TL/DD/11214-29

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address S/ADD REG	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads As All Ones)
xx80 to xxAF	Unused RAM Address Space (Reads Undefined Data)
xxB0	Timer T3 Lower Byte
XXB1	Timer T3 Upper Byte
xxB2	Timer T3 Autoload Register T3RA Lower Byte
xxB3	Timer T3 Autoload Register T3RA Upper Byte
xxB4	Timer T3 Autoload Register T3RB Lower Byte
xxB5	Timer T3 Autoload Register T3RB Upper Byte
xxB6	Timer T3 Control Register
xxB7	Comparator Select Register (CMPSL)
xxB8	UART Transmit Buffer (TBUF)
xxB9	UART Receive Buffer (RBUF)
xxBA	UART Control and Status Register (ENU)
xxBB	UART Receive Control and Status Register (ENUR)
xxBC	UART Interrupt and Clock Source Register (ENUI)
xxBD	UART Baud Register (BAUD)
xxBE	UART Prescale Select Register (PSR)
xxBF	Reserved for UART
xxC0	Timer T2 Lower Byte
xxC1	Timer T2 Upper Byte
xxC2	Timer T2 Autoload Register T2RA Lower Byte
xxC3	Timer T2 Autoload Register T2RA Upper Byte
xxC4	Timer T2 Autoload Register T2RB Lower Byte
xxC5	Timer T2 Autoload Register T2RB Upper Byte
xxC6	Timer T2 Control Register
xxC7	WATCHDOG Service Register (Reg:WDSVR)
xxC8	MIWU Edge Select Register (Reg:WKEDG)
xxC9	MIWU Enable Register (Reg:WKEN)
xxCA	MIWU Pending Register (Reg:WKPND)
xxCB	Reserved
xxCC	Reserved
xxCD to xxCF	Reserved

Address S/ADD REG	Contents
xxD0	Port L Data Register
xxD1	Port L Configuration Register
xxD2	Port L Input Pins (Read Only)
xxD3	Reserved for Port L
xxD4	Port G Data Register
xxD5	Port G Configuration Register
xxD6	Port G Input Pins (Read Only)
xxD7	Port I Input Pins (Read Only)
xxD8	Port C Data Register
xxD9	Port C Configuration Register
xxDA	Port C Input Pins (Read Only)
xxDB	Reserved for Port C
xxDC	Port D
xxDD to DF	Reserved for Port D
xxE0 to xxE5	Reserved for EE Control Registers
xxE6	Timer T1 Autoload Register T1RB Lower Byte
xxE7	Timer T1 Autoload Register T1RB Upper Byte
xxE8	ICNTRL Register
xxE9	MICROWIRE/PLUS Shift Register
xxEA	Timer T1 Lower Byte
xxEB	Timer T1 Upper Byte
xxEC	Timer T1 Autoload Register T1RA Lower Byte
xxED	Timer T1 Autoload Register T1RA Upper Byte
xxEE	CNTRL Control Register
xxEF	PSW Register
xxF0 to FB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	S Register
0100-017F	On-Chip 128 RAM Bytes

Reading memory locations 0070H-007FH (Segment 0) will return all ones. Reading unused memory locations 0080H-00AFH (Segment 0) will return undefined data. Reading memory locations from other Segments (i.e., Segment 2, Segment 3, ... etc.) will return all ones.

Addressing Modes

There are ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the Load B Immediate instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from -31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD ADC	A,Meml A,Meml	ADD ADD with Carry	$A \leftarrow A + Meml$ $A \leftarrow A + Meml + C, C \leftarrow Carry$ HC ← Half Carry
SUBC	A,Meml	Subtract with Carry	$A \leftarrow A - Meml + C, C \leftarrow Carry$ HC ← Half Carry
AND ANDSZ OR XOR	A,Meml A,Imm A,Meml A,Meml	Logical AND Logical AND Immed., Skip if Zero Logical OR Logical EXclusive OR	$A \leftarrow A \text{ and } Meml$ Skip next if $(A \text{ and } Imm) = 0$ $A \leftarrow A \text{ or } Meml$ $A \leftarrow A \text{ xor } Meml$
IFEQ IFEQ IFNE IFGT IFBNE	MD,Imm A,Meml A,Meml A,Meml #	IF Equal IF Equal IF Not Equal IF Greater Than If B Not Equal	Compare MD and Imm, Do next if MD = Imm Compare A and Meml, Do next if A = Meml Compare A and Meml, Do next if A ≠ Meml Compare A and Meml, Do next if A > Meml Do next if lower 4 bits of B ≠ Imm
DRSZ SBIT RBIT IFBIT RPND	Reg #,Mem #,Mem #,Mem #,Mem	Decrement Reg., Skip if Zero Set BIT Reset BIT IF BIT Reset PeNDing Flag	Reg ← Reg - 1, Skip if Reg = 0 1 to bit, Mem (bit = 0 to 7 immediate) 0 to bit, Mem If bit in A or Mem is true do next instruction Reset Software Interrupt Pending Flag
X X LD LD LD LD LD	A,Mem A,[X] A,Meml A,[X] B,Imm Mem,Imm Reg,Imm	EXchange A with Memory EXchange A with Memory [X] LoAD A with Memory LoAD A with Memory [X] LoAD B with Immed. LoAD Memory Immed LoAD Register Memory Immed.	$A \leftrightarrow Mem$ $A \leftrightarrow [X]$ $A \leftarrow Meml$ $A \leftarrow [X]$ $B \leftarrow Imm$ $Mem \leftarrow Imm$ $Reg \leftarrow Imm$
X X LD LD LD	A, [B ±] A, [X ±] A, [B ±] A, [X ±] [B ±],Imm	EXchange A with Memory [B] EXchange A with Memory [X] LoAD A with Memory [B] LoAD A with Memory [X] LoAD Memory [B] Immed.	$A \leftrightarrow [B], (B \leftarrow B \pm 1)$ $A \leftrightarrow [X], (X \leftarrow \pm 1)$ $A \leftarrow [B], (B \leftarrow B \pm 1)$ $A \leftarrow [X], (X \leftarrow X \pm 1)$ $[B] \leftarrow Imm, (B \leftarrow B \pm 1)$
CLR INC DEC LAID DCOR RRC RLC SWAP SC RC IFC IFNC POP PUSH	A A A A A A A A A A A A A A	CLear A INCrement A DECrementA LoAD A InDirect from ROM Decimal CORrect A Rotate A Right thru C Rotate A Left thru C SWAP nibbles of A Set C Reset C IF C IF Not C POP the stack into A PUSH A onto the stack	$A \leftarrow 0$ $A \leftarrow A + 1$ $A \leftarrow A - 1$ $A \leftarrow ROM (PU,A)$ A ← BCD correction of A (follows ADC, SUBC) $C \rightarrow A7 \rightarrow \dots \rightarrow A0 \rightarrow C$ $C \leftarrow A7 \leftarrow \dots \leftarrow A0 \leftarrow C$ $A7 \dots A4 \leftrightarrow A3 \dots A0$ $C \leftarrow 1, HC \leftarrow 1$ $C \leftarrow 0, HC \leftarrow 0$ If C is true, do next instruction If C is not true, do next instruction $SP \leftarrow SP + 1, A \leftarrow [SP]$ $[SP] \leftarrow A, SP \leftarrow SP - 1$
VIS JMPL JMP JP JSRL JSR JID RET RETSK RETI INTR NOP	Addr. Addr. Disp. Addr. Addr	Vector to Interrupt Service Routine Jump absolute Long Jump absolute Jump relative short Jump SubRoutine Long Jump SubRoutine Jump InDirect RETURN from subroutine RETURN and Skip RETURN from Interrupt Generate an Interrupt No OPERATION	$PU \leftarrow [VU], PL \leftarrow [VL]$ $PC \leftarrow ii (ii = 15 \text{ bits}, 0 \text{ to } 32k)$ $PC9 \dots 0 \leftarrow i (i = 12 \text{ bits})$ $PC \leftarrow PC + r (r \text{ is } -31 \text{ to } +32, \text{ except } 1)$ $[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC \leftarrow ii$ $[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC9 \dots 0 \leftarrow i$ $PL \leftarrow ROM (PU,A)$ $SP + 2, PL \leftarrow [SP], PU \leftarrow [SP-1]$ $SP + 2, PL \leftarrow [SP], PU \leftarrow [SP-1]$ $SP + 2, PL \leftarrow [SP], PU \leftarrow [SP-1], GIE \leftarrow 1$ $[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC \leftarrow 0FF$ $PC \leftarrow PC + 1$

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.	Instructions Using A & C		Transfer of Control Instructions	
ADD	1/1	3/4	2/2	CLRA	1/1	JMPL	3/4
ADC	1/1	3/4	2/2	INCA	1/1	JMP	2/3
SUBC	1/1	3/4	2/2	DECA	1/1	JP	1/3
AND	1/1	3/4	2/2	LAID	1/3	JSRL	3/5
OR	1/1	3/4	2/2	DCOR	1/1	JSR	2/5
XOR	1/1	3/4	2/2	RRCA	1/1	JID	1/3
IFEQ	1/1	3/4	2/2	RLCA	1/1	VIS	1/5
IFNE	1/1	3/4	2/2	SWAPA	1/1	RET	1/5
IFGT	1/1	3/4	2/2	SC	1/1	RETSK	1/5
IFBNE	1/1			RC	1/1	RETI	1/5
DRSZ		1/3		IFC	1/1	INTR	1/7
				IFNC	1/1	NOP	1/1
SBIT	1/1	3/4		PUSHA	1/3		
RBIT	1/1	3/4		POPA	1/3		
IFBIT	1/1	3/4		ANDSZ	2/2		

RPND	1/1
------	-----

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.	
	[B]	[X]			[B+, B-]	[X+, X-]
XA,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B, Imm				1/1		
LD B, Imm				2/2		
LD Mem, Imm	2/2		3/3		2/2	
LD Reg, Imm			2/3			
IFEQ MD, Imm			3/3			

(IF B < 16)
(IF B > 15)

* = > Memory location addressed by B or X or directly.

Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP -15	JP -31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A,[B]	0
JP -14	JP -30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP -13	JP -29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A, #i	IFEQ A,[B]	2
JP -12	JP -28	LD 0F3, # i	DRSZ 0F3	X A, [X-]	X A,[B-]	IFGT A, #i	IFGT A,[B]	3
JP -11	JP -27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A, #i	ADD A,[B]	4
JP -10	JP -26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A, #i	AND A,[B]	5
JP -9	JP -25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A, #i	XOR A,[B]	6
JP -8	JP -24	LD 0F7, # i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	7
JP -7	JP -23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A, #i	IFC	8
JP -6	JP -22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md, #i	IFNE A, #i	IFNC	9
JP -5	JP -21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+], #i	INCA	A
JP -4	JP -20	LD 0FB, # i	DRSZ 0FB	LD A,[X-]	LD A,[B-]	LD [B-], #i	DECA	B
JP -3	JP -19	LD 0FC, # i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	POPA	C
JP -2	JP -18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	D
JP -1	JP -17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B], #i	RET	E
JP -0	JP -16	LD 0FF, # i	DRSZ 0FF	*	*	LD B, #i	RETI	F

Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B, #0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP + 17	INTR	0
IFBIT 1,[B]	*	LD B, #0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP + 18	JP + 2	1
IFBIT 2,[B]	*	LD B, #0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP + 19	JP + 3	2
IFBIT 3,[B]	*	LD B, #0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP + 20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B, #0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP + 21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B, #0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP + 22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B, #09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP + 23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B, #08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP + 24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B, #07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP + 25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B, #06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP + 26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B, #05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP + 27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B, #04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP + 28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B, #03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP + 29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B, #02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP + 30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B, #01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP + 31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B, #00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP + 32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A**Mask Options**

The COP888CG mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

- = 1 Crystal Oscillator (CKI/10)
G7 (CK0) is clock generator output to crystal/resonator
CKI is the clock input
- = 2 Single-pin RC controlled oscillator (CKI/10)
G7 is available as a HALT restart and/or general purpose input

OPTION 2: HALT

- = 1 Enable HALT mode
- = 2 Disable HALT mode

OPTION 3: BONDING OPTIONS

- = 1 44-Pin PLCC
- = 2 40-Pin DIP
- = 3 N/A
- = 4 28-Pin DIP/SO

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface or maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use window interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC® via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS-232 serial interface cable
MHW-PS3	Power supply 110V/60 MHz
MHW-PS4	Power supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884EG28D5PC	28 DIP	4.5V-5.5V	COP884EG
MHW-884EG28DWPC	28 DIP	2.5V-6.0V	COP884EG
MHW-888EG40D5PC	40 DIP	4.5V-5.5V	COP888EG
MHW-888EG40DWPC	40 DIP	2.5V-6.0V	COP888EG
MWH-888EG44D5PC	44 PLCC	4.5V-5.5V	COP888EG
MHW-888EG44DWPC	44 PLCC	2.5V-6.0V	COP888EG

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM®, PC-/XT®, PC-AT® or compatible	424410527-001

Development Support (Continued)

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific datasheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code

from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888EGMHXL-X	X = 1: crystal X = 3: R/C	44 LDCC	Multi-Chip Module (MCM), UV erasable	COP888EG
COP888EGMHD-X	X = 1: crystal X = 3: R/C	40 DIP	MCM, UV erasable	COP888EG

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP socket	COP888EGMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888EGMHXL
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP scrambler board	COP888EGMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC scrambler board	COP888EGMEL

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

COP688CS/COP684CS/COP888CS/COP884CS/ COP988CS/COP984CS Single-Chip microCMOS Microcontroller

General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²C²MOS™ process technology. The COP888CS is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 4096 bytes on-board ROM
- 192 bytes on-board RAM
- Single supply operation: 2.5V–6V
- Full duplex UART
- One analog comparator
- MICROWIRE/PLUS™ serial I/O
- WATCHDOG™ and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- One 16-bit timer, with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Ten multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Timer (2)
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - UART (2)
 - Default VIS
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PLCC or 40 N or 28 N or 28 SOIC
 - 44 PLCC with 39 I/O pins
 - 40 N with 35 I/O pins
 - 28 SO or 28 N, each with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form factor emulation devices
- Real time emulation and full program debug offered by National's Development Systems
- For other COP800 devices with a UART see COP888CG and COP888EG

Block Diagram

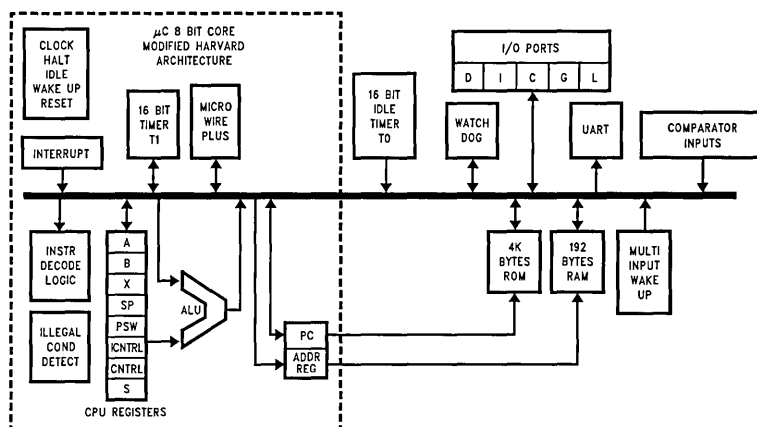


FIGURE 1. COP888CS Block Diagram

TL/DD/10830-1

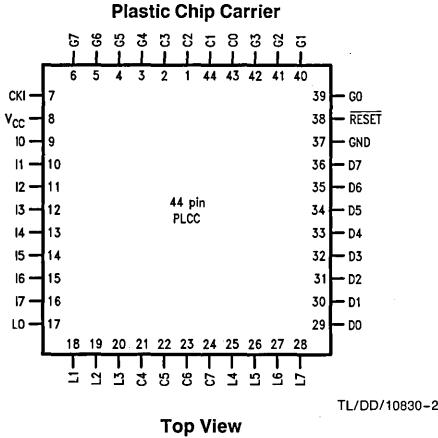
COP688CS/COP684CS/COP888CS/COP884CS/COP988CS/COP984CS

General Description (Continued)

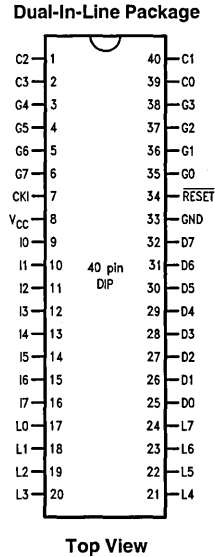
It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, one 16-bit timer/counter supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, one comparator, and two power savings modes (HALT and

IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CS operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

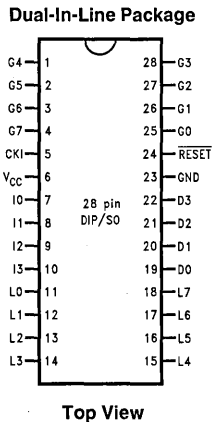
Connection Diagrams



Top View
 Order Number COP888CS-XXX/V
 See NS Package Number V44A



Top View
 Order Number COP888S-XXX/N
 See NS Package Number N40A



Top View
 Order Number COP884CS-XXX/N
 See NS Package Number N28B
 Order Number COP884CS-XXX/WM
 See NS Package Number M28B

FIGURE 2. COP888CS Connection Diagrams

Connection Diagrams (Continued)

COP888CS Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU		15	21	25
L5	I/O	MIWU		16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUIT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
I0	I			7	9	9
I1	I	COMP1IN-		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I				13	13
I5	I				14	14
I6	I				15	15
I7	I				16	16
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V _{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 7V
 Voltage at Any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 100 mA

Total Current out of GND Pin (Sink) 110 mA

Storage Temperature Range $-65^{\circ}C$ to $+140^{\circ}C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

98XCS: $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units								
Operating Voltage	COP98XCS COP98XCSH	2.5		4.0	V								
		4.0		6.0	V								
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1 V_{CC}$	V								
Supply Current (Note 2)	$V_{CC} = 6V, t_c = 1 \mu s$ $V_{CC} = 6V, t_c = 2.5 \mu s$ $V_{CC} = 4V, t_c = 2.5 \mu s$ $V_{CC} = 4V, t_c = 10 \mu s$			12.5	mA								
				5.5	mA								
				2.5	mA								
				1.4	mA								
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$ $V_{CC} = 4V, CKI = 0 MHz$		<0.7	8	μA								
			<0.3	4	μA								
IDLE Current	$V_{CC} = 6V, t_c = 1 \mu s$ $V_{CC} = 6V, t_c = 2.5 \mu s$ $V_{CC} = 4V, t_c = 10 \mu s$			3.5	mA								
				2.5	mA								
				0.7	mA								
Input Levels					RESET								
					Logic High	$0.8 V_{CC}$		V					
					Logic Low		$0.2 V_{CC}$	V					
					CKI (External and Crystal Osc. Modes)								
					Logic High	$0.7 V_{CC}$		V					
					Logic Low		$0.2 V_{CC}$	V					
All Other Inputs					Logic High	V							
					Logic Low	$0.2 V_{CC}$	V						
Hi-Z Input Leakage	$V_{CC} = 6V, V_{IN} = 0V$	-1		+1	μA								
Input Pullup Current	$V_{CC} = 6V, V_{IN} = 0V$	40		250	μA								
G and L Port Input Hysteresis				$0.35 V_{CC}$	V								
Output Current Levels					D Outputs								
					Source	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4		mA				
						$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2		mA				
					Sink	$V_{CC} = 4V, V_{OL} = 1V$	10		mA				
						$V_{CC} = 2.5V, V_{OL} = 0.4V$	2.0		mA				
					All Others					Source (Weak Pull-Up Mode)			
										$V_{CC} = 4V, V_{OH} = 2.7V$	10		μA
										$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		μA
										Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4	
						$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2		mA				
Sink (Push-Pull Mode)					$V_{CC} = 4V, V_{OL} = 0.4V$	1.6		mA					
					$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7		mA					
TRI-STATE Leakage	$V_{CC} = 6.0V$	-1		+1	μA								

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

DC Electrical Characteristics 98XCS: $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				15 3	mA mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics 98XCS: $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)	$4\text{V} \leq V_{CC} \leq 6\text{V}$	1		DC	μs
Crystal, Resonator, R/C Oscillator	$2.5\text{V} \leq V_{CC} < 4\text{V}$	2.5		DC	μs
	$4\text{V} \leq V_{CC} \leq 6\text{V}$	3		DC	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	7.5		DC	μs
CKI Clock Duty Cycle (Note 5)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 5)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Fall Time (Note 5)	$f_r = 10\text{ MHz Ext Clock}$			5	ns
Inputs					
t_{SETUP}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	200			ns
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	500			ns
t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$	60			ns
	$2.5\text{V} \leq V_{CC} < 4\text{V}$	150			ns
Output Propagation Delay	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$				
$t_{\text{PD1}}, t_{\text{PD0}}$	$4\text{V} \leq V_{CC} \leq 6\text{V}$			0.7	μs
SO, SK	$2.5\text{V} \leq V_{CC} < 4\text{V}$			1.75	μs
All Others	$4\text{V} \leq V_{CC} \leq 6\text{V}$			1	μs
	$2.5\text{V} \leq V_{CC} < 4\text{V}$			2.5	μs
MICROWIRE™ Setup Time (t_{JWS})		20			ns
MICROWIRE Hold Time (t_{JWH})		56			ns
MICROWIRE Output Propagation Delay (t_{JPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

Note 5: Parameter sampled but not 100% tested.

Note 6: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	100 mA

Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 88XCS: -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			5.5	mA
HALT Current (Note 3)					
	$V_{CC} = 6V, CKI = 0 MHz$		<1	10	μA
	$V_{CC} = 4V, CKI = 0 MHz$		<0.5	6	μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			2.5	mA
CKI = 1 MHz	$V_{CC} = 4V, t_c = 10 \mu s$			0.7	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V, V_{IN} = 0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6V, V_{IN} = 0V$	40		250	μA
G and L Port Input Hysteresis				0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$	10			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	2.0			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$	10		100	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage		-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

DC Electrical Characteristics 88XCS: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				15 3	mA mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics 88XCS: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	1 2.5 3 7.5		DC DC DC DC	μs μs μs μs
CKI Clock Duty Cycle (Note 5) Rise Time (Note 5) Fall Time (Note 5)	$f_r = \text{Max}$ $f_r = 10\text{ MHz Ext Clock}$ $f_r = 10\text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}	$4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$	200 500 60 150			ns ns ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100\text{ pF}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$ $4\text{V} \leq V_{CC} \leq 6\text{V}$ $2.5\text{V} \leq V_{CC} < 4\text{V}$			0.7 1.75 1 2.5	μs μs μs μs
MICROWIRE Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 5: Parameter sampled but not 100% tested.

Note 6: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC}) 7V
 Voltage at Any Pin $-0.3V$ to $V_{CC} + 0.3V$
 Total Current into V_{CC} Pin (Source) 100 mA

Total Current out of GND Pin (Sink) 110 mA

Storage Temperature Range -65°C to $+140^{\circ}\text{C}$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics 68XCS: $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1 V_{CC}$	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			12.5	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu s$			5.5	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, \text{CKI} = 0 \text{ MHz}$		< 10	30	μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			3.5	mA
CKI = 4 MHz	$V_{CC} = 5.5V, t_c = 2.5 \mu s$			2.5	mA
Input Levels					
RESET					
Logic High		$0.8 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
CKI (External and Crystal Osc. Modes)					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 5.5V, V_{IN} = 0V$	-5		+5	μA
Input Pullup Current	$V_{CC} = 5.5V, V_{IN} = 0V$	35		400	μA
G and L Port Input Hysteresis				$0.35 V_{CC}$	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1V$	9			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	9		140	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.4			mA
TRI-STATE Leakage	$V_{CC} = 5.5V$	-5		+5	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

DC Electrical Characteristics 68XCS: $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				12 2.5	mA mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics 68XCS: $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ unless otherwise specified

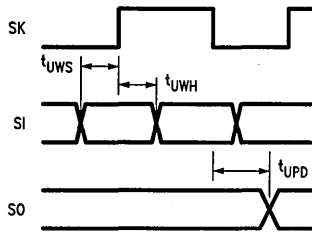
Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator	$4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$	1 3		DC DC	μs μs
CKI Clock Duty Cycle (Note 5) Rise Time (Note 5) Fall Time (Note 5)	$f_r = \text{Max}$ $f_r = 10\text{ MHz Ext Clock}$ $f_r = 10\text{ MHz Ext Clock}$	45		55 12 8	% ns ns
Inputs t_{SETUP} t_{HOLD}	$4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$	200 60			ns ns
Output Propagation Delay $t_{\text{PD1}}, t_{\text{PD0}}$ SO, SK All Others	$R_L = 2.2\text{k}, C_L = 100\text{ pF}$ $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$ $4.5\text{V} \leq V_{CC} \leq 5.5\text{V}$			0.7 1	μs μs
MICROWIRE Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 5: Parameter sampled but not 100% tested.

Note 6: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Comparator AC and DC Characteristics $V_{CC} = 5V, T_A = 25^\circ C$

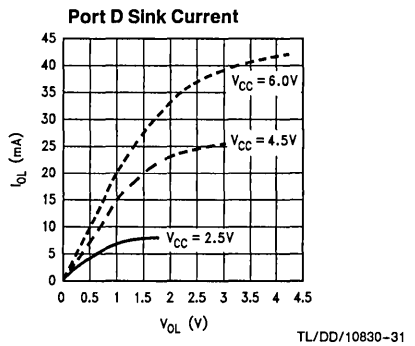
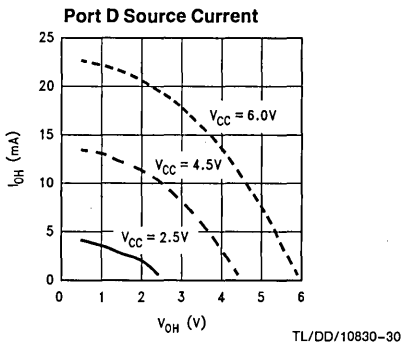
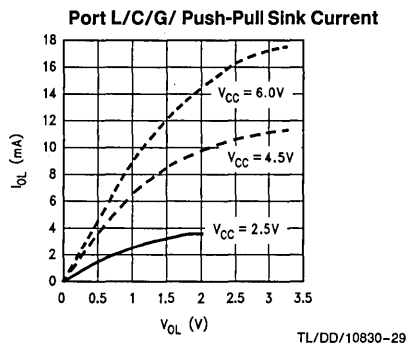
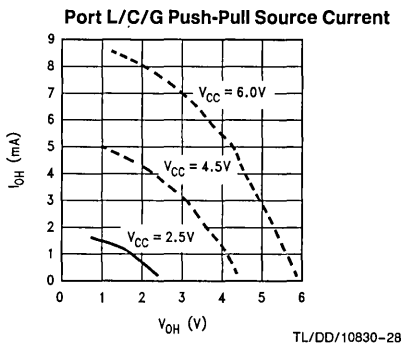
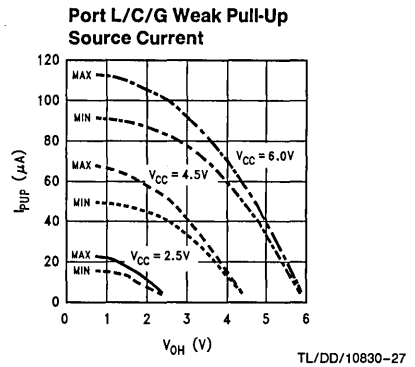
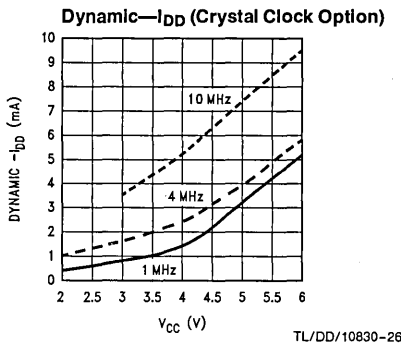
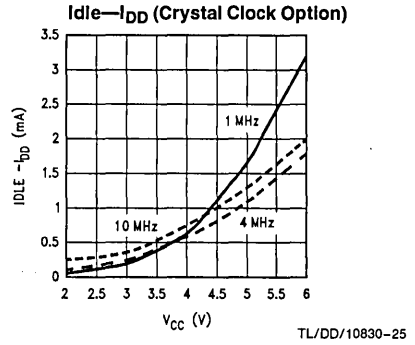
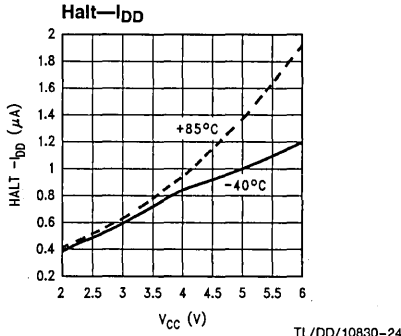
Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		± 10	± 25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



TL/DD/10830-6

FIGURE 3. MICROWIRE/PLUS Timing

Typical Performance Characteristics (-40°C to +85°C)



Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CS contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports L and G), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CS memory map for the various addresses associated with the I/O ports.) Figure 4 shows the I/O port configurations for the COP888CS. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

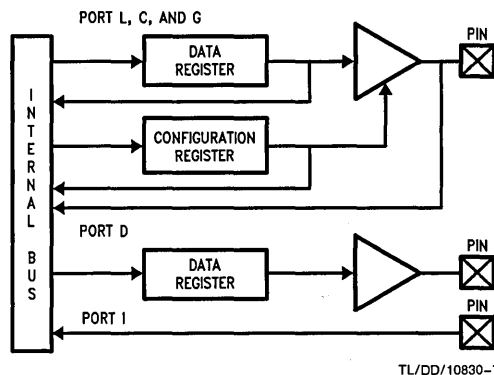


FIGURE 4. I/O Port Configurations

Port L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive.

The Port L has the following alternate features:

- L0 MIWU
- L1 MIWU or CKX
- L2 MIWU or TDX
- L3 MIWU or RDX

- L4 MIWU
- L5 MIWU
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT WATCHDOG and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

Port C is an 8-bit I/O port. The 40-pin device does not have a full complement of Port C pins. The unavailable pins are not terminated. A read operation for these unterminated pins will return unpredictable values.

Port I is an eight-bit Hi-Z input port. The 28-pin device does not have a full complement of Port I pins. The unavailable

Pin Descriptions (Continued)

pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Ports I1–I3 are used for Comparator 1.

Ports I1–I3 have the following alternate features.

- I1 COMP1–IN (Comparator 1 Negative Input)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I3 COMP1OUT (Comparator 1 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CS is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CS architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_c) cycle time.

There are six CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

S is the 8-bit Data Segment Address Register used to extend the lower half of the address range (00 to 7F) into 256 data segments of 128 bytes each.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CS consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CS vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data

and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X, SP pointers and S register.

The COP888CS has 192 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, B and S are memory mapped into this space at address locations 0FC to 0FF Hex respectively, with the other registers being available for general usage.

The instruction set of the COP888CS permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CS (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Data Memory Segment RAM Extension

Data memory address 0FF is used as a memory mapped location for the Data Segment Address Register (S) in the COP888CS.

The data store memory is either addressed directly by a single byte address within the instruction, or indirectly relative to the reference of the B, X, or SP pointers (each contains a single-byte address). This single-byte address allows an addressing range of 256 locations from 00 to FF hex. The upper bit of this single-byte address divides the data store memory into two separate sections as outlined previously. With the exception of the RAM register memory from address locations 00F0 to 00FF, all RAM memory is memory mapped with the upper bit of the single-byte address being equal to zero. This allows the upper bit of the single-byte address to determine whether or not the base address range (from 0000 to 00FF) is extended. If this upper bit equals one (representing address range 0080 to 00FF), then address extension does not take place. Alternatively, if this upper bit equals zero, then the data segment extension register S is used to extend the base address range (from 0000 to 007F) from XX00 to XX7F, where XX represents the 8 bits from the S register. Thus the 128-byte data segment extensions are located from addresses 0100 to 017F for data segment 1, 0200 to 027F for data segment 2, etc., up to FF00 to FF7F for data segment 255. The base address range from 0000 to 007F represents data segment 0.

Figure 5 illustrates how the S register data memory extension is used in extending the lower half of the base address range (00 to 7F hex) into 256 data segments of 128 bytes each, with a total addressing range of 32 kbytes from XX00 to XX7F. This organization allows a total of 256 data segments of 128 bytes each with an additional upper base segment of 128 bytes. Furthermore, all addressing modes are available for all data segments. The S register must be changed under program control to move from one data segment (128 bytes) to another. However, the upper base segment (containing the 16 memory registers, I/O registers, control registers, etc.) is always available regardless of the

Data Memory Segment RAM Extension (Continued)

contents of the S register, since the upper base segment (address range 0080 to 00FF) is independent of data segment extension.

The instructions that utilize the stack pointer (SP) always reference the stack as part of the base segment (Segment 0), regardless of the contents of the S register. The S register is not changed by these instructions. Consequently, the stack (used with subroutine linkage and interrupts) is always located in the base segment. The stack pointer will be initialized to point at data memory location 006F as a result of reset.

The 128 bytes of RAM contained in the base segment are split between the lower and upper base segments. The first 112 bytes of RAM are resident from address 0000 to 006F in the lower base segment, while the remaining 16 bytes of RAM represent the 16 data memory registers located at addresses 00F0 to 00FF of the upper base segment. No RAM is located at the upper sixteen addresses (0070 to 007F) of the lower base segment.

Additional RAM beyond these initial 128 bytes, however, will always be memory mapped in groups of 128 bytes (or less) at the data segment address extensions (XX00 to XX7F) of the lower base segment. The additional 64 bytes of RAM in the COP888CS (beyond the initial 128 bytes) are memory mapped at address locations 0100 to 013F hex.

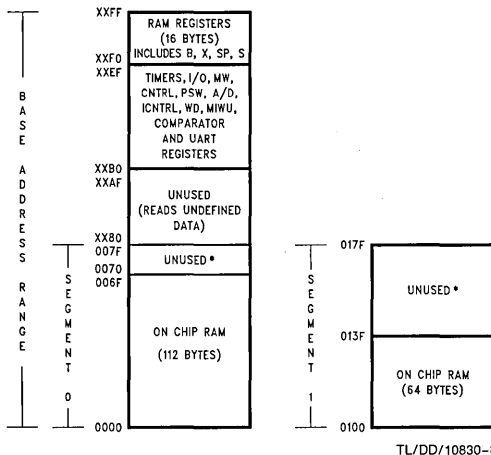


FIGURE 5. RAM Organization

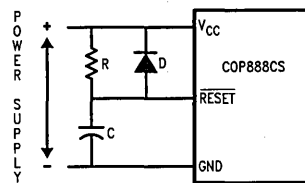
Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for ports L, G and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WATCHDOG and/or Clock Monitor error output pin. Port D is set high. The PC, PSW, ICNTRL, CNTRL, are cleared. The UART registers PSR, ENU (except that TBMT bit is set), ENUR and ENUI are cleared. The Comparator Select Register is cleared. The S register is initialized to zero. The Multi-Input Wakeup registers WKEN,

WKEDG and WKPND are cleared. The stack pointer, SP, is initialized to 6F Hex.

The COP888CS comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, with the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor circuits are inhibited during reset. The WATCHDOG service window bits being initialized high default to the maximum WATCHDOG service window of 64k t_C clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16 t_C –32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in Figure 6 should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes.



TL/DD/10830-9

$RC > 5 \times$ Power Supply Rise Time

FIGURE 6. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Figure 7 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

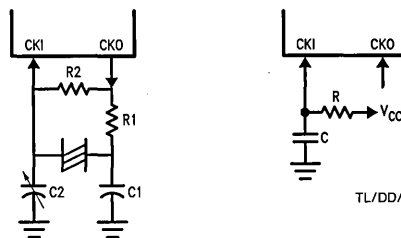
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10830-10

FIGURE 7. Crystal and R/C Oscillator Diagrams

Oscillator Circuits (Continued)

TABLE A. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	10	$V_{CC} = 5V$
0	1	30	30–36	4	$V_{CC} = 5.0V$
0	1	200	100–150	0.455	$V_{CC} = 2.5V$

TABLE B. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Note: $3k \leq R \leq 200k$
 $50 \text{ pF} \leq C \leq 200 \text{ pF}$

Current Drain

The total current drain of the chip depends on:

- Oscillator operation mode—I1
- Internal switching current—I2
- Internal leakage current—I3
- Output source current—I4
- DC current caused by external input not at V_{CC} or GND—I5
- Comparator DC supply current when enabled—I6
- Clock Monitor current when enabled—I7

Thus the total current drain, I_t , is given as

$$I_t = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I_2 = C \times V \times f$$

where C = equivalent capacitance of the chip
 V = operating voltage
 f = CKI frequency

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
- T1C0 Timer T1 Start/Stop control in timer modes 1 and 2
 Timer T1 Underflow Interrupt Pending Flag in timer mode 3
- T1C1 Timer T1 mode control bit
- T1C2 Timer T1 mode control bit
- T1C3 Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
------	------	------	------	------	------	-----	-----

Bit 7

Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable (enables interrupts)
- EXEN Enable external interrupt
- BUSY MICROWIRE/PLUS busy shifting flag
- EXPND External interrupt pending
- T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
- T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
- C Carry Flag
- HC Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE
----	---	--------	-------	-------	------	------	-----

Bit 7

Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

Control Registers (Continued)

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB	Timer T1 Interrupt Enable for T1B Input capture edge
T1PNDB	Timer T1 Interrupt Pending Flag for T1B capture edge
μ WEN	Enable MICROWIRE/PLUS interrupt
μ WPND	MICROWIRE/PLUS interrupt pending
T0EN	Timer T0 Interrupt Enable (Bit 12 toggle)
T0PND	Timer T0 Interrupt pending
LPEN	L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μ WPND	μ WEN	T1PNDB	T1ENB	
Bit 7								Bit 0

Timers

The COP888CS contains a very versatile set of timers (T0, T1). All timers and associated autoreload/capture registers power up containing random data.

TIMER T0 (IDLE TIMER)

The COP888CS supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c . The user cannot read or write to the IDLE Timer T0, which is a count down timer. The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)

WATCHDOG logic (See WATCHDOG description)

Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_c = 1 \mu\text{s}$). A control flag T0EN allows the

interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1

The COP888CS has a powerful timer/counter block.

The timer block consists of a 16-bit timer, T1, and two supporting 16-bit autoreload/capture registers, R1A and R1B. It has two pins associated with it, T1A and T1B. The pin T1A supports I/O required by the timer block, while the pin T1B is an input to the timer block. The powerful and flexible timer block allows the COP888CS to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits T1C3, T1C2, and T1C1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CS to generate a PWM signal with very minimal user intervention.

The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer T1 counts down at a fixed rate of t_c . Upon every underflow the timer is alternately reloaded with the contents of supporting registers, R1A and R1B. The very first underflow of the timer causes the timer to reload from the register R1A. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register R1B.

The T1 Timer control bits, T1C3, T1C2 and T1C1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode. The underflows can be programmed to toggle the T1A output pin. The underflows can also be programmed to generate interrupts.

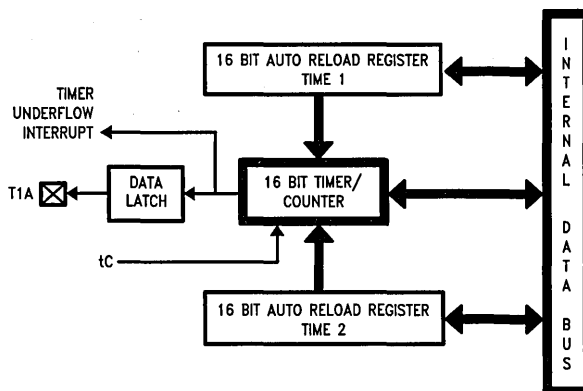


FIGURE 8. Timer in PWM Mode

TL/DD/10830-12

Timers (Continued)

Underflows from the timer are alternately latched into two pending flags, T1PNDA and T1PNDB. The user must reset these pending flags under software control. Two control enable flags, T1ENA and T1ENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag T1ENA will cause an interrupt when a timer underflow causes the R1A register to be reloaded into the timer. Setting the timer enable flag T1ENB will cause an interrupt when a timer underflow causes the R1B register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, T1, is clocked by the input signal from the T1A pin. The Tx timer control bits, T1C3, T1C2 and T1C1 allow the timer to be clocked either on a positive or negative edge from the T1A pin. Underflows from the timer are latched into the T1PNDA pending flag. Setting the T1ENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin T1B can be used as an independent positive edge sensitive interrupt input if the T1ENB control flag is set. The occurrence of a positive edge on the T1B input pin is latched into the T1PNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the T1A pin is being used as the counter input clock.

Mode 3. Input Capture Mode

The COP888CS can precisely measure external frequencies or time external events by placing the timer block, T1, in the input capture mode.

In this mode, the timer T1 is constantly running at the fixed t_c rate. The two registers, R1A and R1B, act as capture registers. Each register acts in conjunction with a pin. The register R1A acts in conjunction with the T1A pin and the register R1B acts in conjunction with the T1B pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, T1C3, T1C2 and T1C1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the T1A and T1B pins will be respectively latched into the pending flags, T1PNDA and T1PNDB. The control flag T1ENA allows the interrupt on T1A to be either enabled or disabled. Setting the T1ENA flag enables interrupts to be generated when the selected trigger condition occurs on the T1A pin. Similarly, the flag T1ENB controls the interrupts from the T1B pin.

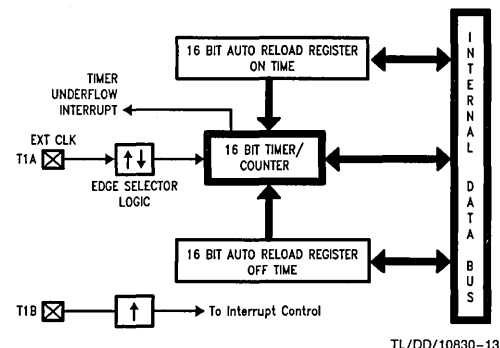
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer T1C0 pending flag (the T1C0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the T1C0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the T1ENA control flag. When a T1A interrupt occurs in the Input Capture mode, the user must check both the T1PNDA and T1C0 pending flags in order to determine whether a T1A input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

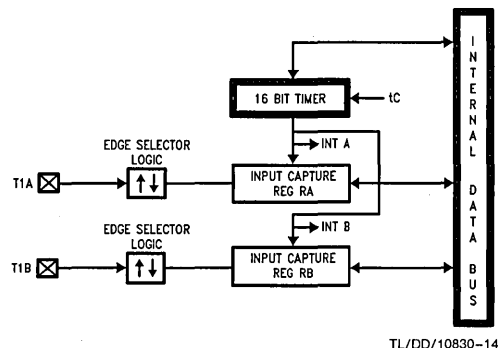
The control bits and their functions are summarized below.

- T1C0 Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop
- T1C0 Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
- T1PNDA Timer Interrupt Pending Flag
- T1PNDB Timer Interrupt Pending Flag
- T1ENA Timer Interrupt Enable Flag
- T1ENB Timer Interrupt Enable Flag
1 = Timer Interrupt Enabled
0 = Timer Interrupt Disabled
- T1C3 Timer mode control
- T1C2 Timer mode control
- T1C1 Timer mode control



TL/DD/10830-13

FIGURE 9. Timer in External Event Counter Mode



TL/DD/10830-14

FIGURE 10. Timer in Input Capture Mode

Timers (Continued)

The timer mode control bits (T1C3, T1C2 and T1C1) are detailed below:

T1C3	T1C2	T1C1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. T1B Edge	T1A Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. T1B Edge	T1A Neg. Edge
1	0	1	MODE 1 (PWM) T1A Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No T1A Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: T1A Pos. Edge T1B Pos. Edge	Pos. T1A Edge or Timer Underflow	Pos. T1B Edge	t_c
1	1	0	MODE 3 (Capture) Captures: T1A Pos. Edge T1B Neg. Edge	Pos. T1A Edge or Timer Underflow	Neg. T1B Edge	t_c
0	1	1	MODE 3 (Capture) Captures: T1A Neg. Edge T1B Pos. Edge	Neg. T1B Edge or Timer Underflow	Pos. T1B Edge	t_c
1	1	1	MODE 3 (Capture) Captures: T1A Neg. Edge T1B Neg. Edge	Neg. T1A Edge or Timer Underflow	Neg. T1B Edge	t_c

Power Save Modes

The COP888CS offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry the WATCHDOG logic, the Clock Monitor and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CS is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock and timers, are stopped. The WATCHDOG logic on the COP888CS is disabled during the HALT mode. However, the clock monitor circuitry if enabled remains active and will cause the WATCHDOG output pin (WDOUT) to go low. If the HALT mode is used and the user does not want to activate the WDOUT pin, the Clock Monitor should be disabled after the device comes out of reset (resetting the Clock Monitor control bit with the first write to the WDSVR register). In the HALT mode, the power requirements of the COP888CS are minimal and the applied voltage (V_{CC}) may be decreased to V_r ($V_r = 2.0V$) without altering the state of the machine.

The COP888CS supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is

with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

Power Save Modes (Continued)

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

The COP888CS has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CS will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CS cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WATCHDOG detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the COP888CS inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CS is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activities, except the associated on-board oscillator circuitry, the WATCHDOG logic, the clock monitor and the IDLE Timer T0, are stopped. The power supply requirements of the micro-controller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CS can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes

normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_c = 1 \mu\text{s}$) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the TOPND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the TOPND bit gets set, the COP888CS will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CS will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CS from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 11 shows the Multi-Input Wakeup logic for the COP888CS microcontroller.

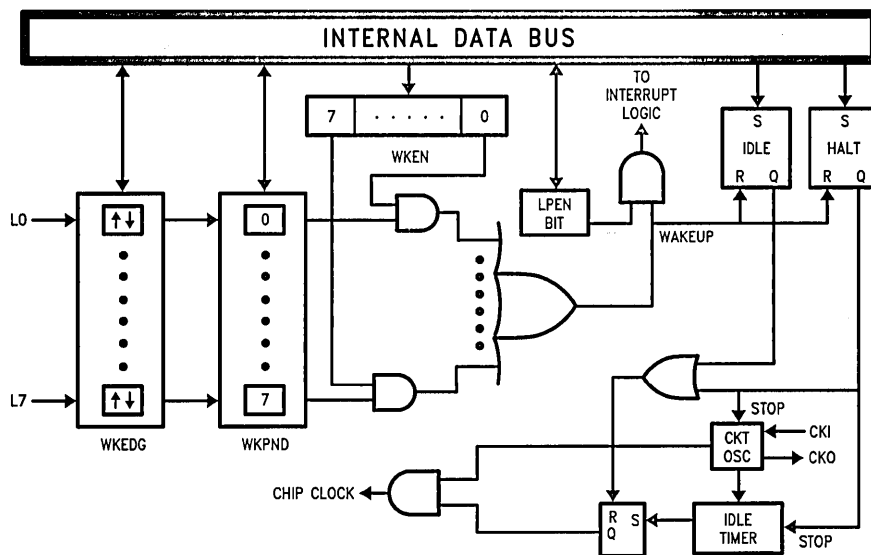


FIGURE 11. Multi-Input Wake Up Logic

TL/DD/10830-15

Multi-Input Wakeup (Continued)

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CS to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```

RBIT 5, WKEN
SBIT 5, WKEDG
RBIT 5, WKPND
SBIT 5, WKEN

```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CS will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CS out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CS will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CS will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CS to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

UART

The COP888CS contains a full-duplex software programmable UART. The UART (Figure 12) consists of a transmit shift register, a receiver shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR), a UART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framing, data overrun and parity errors while the UART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the UART's attention mode of operation and providing additional receiver/transmitter status information via RCVG and XMTG bits. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts. A control flag in this register can also select the UART mode of operation: asynchronous or synchronous.

UART CONTROL AND STATUS REGISTERS

The operation of the UART is programmed through three registers: ENU, ENUR and ENUI. The function of the individual bits in these registers is as follows:

ENU-UART Control and Status Register (Address at 0BA)

PEN	PSEL1	XBIT9/ PSEL0	CHL1	CHL0	ERR	RBFL	TBMT
0RW	0RW	0RW	0RW	0RW	0R	0R	1R

Bit 7 Bit 0

ENUR-UART Receive Control and Status Register (Address at 0BB)

DOE	FE	PE	SPARE	RBIT9	ATTN	XMTG	RCVG
0RD	0RD	0RD	0RW*	0R	0RW	0R	0R

Bit 7 Bit 0

ENUI-UART Interrupt and Clock Source Register (Address at 0BC)

STP2	STP78	ETDX	SSEL	XRCLK	XTCLK	ERI	ETI
0RW	0RW	0RW	0RW	0RW	0RW	0RW	0RW

Bit 7 Bit 0

*Bit is not used.

0 Bit is cleared on reset.

1 Bit is set to one on reset.

R Bit is read-only; it cannot be written by software.

RW Bit is read/write.

D Bit is cleared on read; when read by software as a one, it is cleared automatically. Writing to the bit does not affect its state.

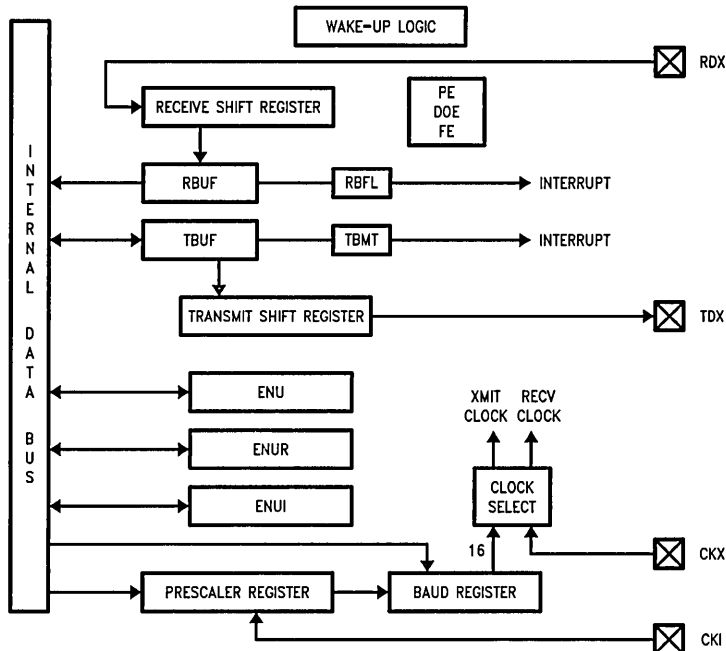


FIGURE 12. UART Block Diagram

TL/DD/10830-16

UART (Continued)**DESCRIPTION OF UART REGISTER BITS****ENU—UART CONTROL AND STATUS REGISTER**

TBMT: This bit is set when the UART transfers a byte of data from the TBUF register into the TSFT register for transmission. It is automatically reset when software writes into the TBUF register.

RBF: This bit is set when the UART has received a complete character and has copied it into the RBUF register. It is automatically reset when software reads the character from RBUF.

ERR: This bit is a global UART error flag which gets set if any or a combination of the errors (DOE, FE, PE) occur.

CHL1, CHL0: These bits select the character frame format. Parity is not included and is generated/verified by hardware.
 CHL1 = 0, CHL0 = 0 The frame contains eight data bits.
 CHL1 = 0, CHL0 = 1 The frame contains seven data bits.
 CHL1 = 1, CHL0 = 0 The frame contains nine data bits.
 CHL1 = 1, CHL0 = 1 Loopback Mode selected. Transmitter output internally looped back to receiver input. Nine bit framing format is used.

XBIT9/PSEL0: Programs the ninth bit for transmission when the UART is operating with nine data bits per frame. For seven or eight data bits per frame, this bit in conjunction with PSEL1 selects parity.

PSEL1, PSEL0: Parity select bits.

PSEL1 = 0, PSEL0 = 0 Odd Parity (if Parity enabled)
 PSEL1 = 0, PSEL0 = 1 Even Parity (if Parity enabled)
 PSEL1 = 1, PSEL0 = 0 Mark(1) (if Parity enabled)
 PSEL1 = 1, PSEL0 = 1 Space(0) (if Parity enabled)

PEN: This bit enables/disables Parity (7- and 8-bit modes only).

PEN = 0 Parity disabled.
 PEN = 1 Parity enabled.

ENUR—UART RECEIVE CONTROL AND STATUS REGISTER

RCVG: This bit is set high whenever a framing error occurs and goes low when RDX goes high.

XMTG: This bit is set to indicate that the UART is transmitting. It gets reset at the end of the last frame (end of last Stop bit).

ATTN: ATTENTION Mode is enabled while this bit is set. This bit is cleared automatically on receiving a character with data bit nine set.

RBIT9: Contains the ninth data bit received when the UART is operating with nine data bits per frame.

SPARE: Reserved for future use.

PE: Flags a Parity Error.

PE = 0 Indicates no Parity Error has been detected since the last time the ENUR register was read.
 PE = 1 Indicates the occurrence of a Parity Error.

FE: Flags a Framing Error.

FE = 0 Indicates no Framing Error has been detected since the last time the ENUR register was read.
 FE = 1 Indicates the occurrence of a Framing Error.

DOE: Flags a Data Overrun Error.

DOE = 0 Indicates no Data Overrun Error has been detected since the last time the ENUR register was read.

DOE = 1 Indicates the occurrence of a Data Overrun Error.

ENUI—UART INTERRUPT AND CLOCK SOURCE REGISTER

ETI: This bit enables/disables interrupt from the transmitter section.

ETI = 0 Interrupt from the transmitter is disabled.
 ETI = 1 Interrupt from the transmitter is enabled.

ERI: This bit enables/disables interrupt from the receiver section.

ERI = 0 Interrupt from the receiver is disabled.
 ERI = 1 Interrupt from the receiver is enabled.

XTCLK: This bit selects the clock source for the transmitter-section.

XTCLK = 0 The clock source is selected through the PSR and BAUD registers.

XTCLK = 1 Signal on CKX (L1) pin is used as the clock.

XRCLK: This bit selects the clock source for the receiver section.

XRCLK = 0 The clock source is selected through the PSR and BAUD registers.

XRCLK = 1 Signal on CKX (L1) pin is used as the clock.

SSEL: UART mode select.

SSEL = 0 Asynchronous Mode.
 SSEL = 1 Synchronous Mode.

ETDX: TDX (UART Transmit Pin) is the alternate function assigned to Port L pin L2; it is selected by setting ETDX bit. To simulate line break generation, software should reset ETDX bit and output logic zero to TDX pin through Port L data and configuration registers.

STP78: This bit is set to program the last Stop bit to be 7/8th of a bit in length.

STP2: This bit programs the number of Stop bits to be transmitted.

STP2 = 0 One Stop bit transmitted.
 STP2 = 1 Two Stop bits transmitted.

Associated I/O Pins

Data is transmitted on the TDX pin and received on the RDX pin. TDX is the alternate function assigned to Port L pin L2; it is selected by setting ETDX (in the ENUI register) to one. RDX is an inherent function of Port L pin L3, requiring no setup.

The baud rate clock for the UART can be generated on-chip, or can be taken from an external source. Port L pin L1 (CKX) is the external clock I/O pin. The CKX pin can be either an input or an output, as determined by Port L Configuration and Data registers (Bit 1). As an input, it accepts a clock signal which may be selected to drive the transmitter and/or receiver. As an output, it presents the internal Baud Rate Generator output.

UART Operation

The UART has two modes of operation: asynchronous mode and synchronous mode.

ASYNCHRONOUS MODE

This mode is selected by resetting the SSEL (in the ENUI register) bit to zero. The input frequency to the UART is 16 times the baud rate.

The TSFT and TBUF registers double-buffer data for transmission. While TSFT is shifting out the current character on the TDX pin, the TBUF register may be loaded by software with the next byte to be transmitted. When TSFT finishes transmitting the current character the contents of TBUF are transferred to the TSFT register and the Transmit Buffer Empty Flag (TBMT in the ENU register) is set. The TBMT flag is automatically reset by the UART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the UART is transmitting. This bit gets reset at the end of the last frame (end of last Stop bit). TBUF is a read/write register.

The RSFT and RBUF registers double-buffer data being received. The UART receiver continually monitors the signal on the RDX pin for a low level to detect the beginning of a Start bit. Upon sensing this low level, it waits for half a bit time and samples again. If the RDX pin is still low, the receiver considers this to be a valid Start bit, and the remaining bits in the character frame are each sampled a single time, at the mid-bit position. Serial data input on the RDX pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the RBUF register and the Received Buffer Full Flag (RBFL) is set. RBFL is automatically reset when software reads the character from the RBUF register. RBUF is a read only register. There is also the RCVG bit which is set high when a framing error occurs and goes low once RDX goes high. TBMT, XMTG, RBFL and RCVG are read only bits.

SYNCHRONOUS MODE

In this mode data is transferred synchronously with the clock. Data is transmitted on the rising edge and received on the falling edge of the synchronous clock.

This mode is selected by setting SSEL bit in the ENUI register. The input frequency to the UART is the same as the baud rate.

When an external clock input is selected at the CKX pin, data transmit and receive are performed synchronously with this clock through TDX/RDX pins.

If data transmit and receive are selected with the CKX pin as clock output, the μ C generates the synchronous clock output at the CKX pin. The internal baud rate generator is used to produce the synchronous clock. Data transmit and receive are performed synchronously with this clock.

FRAMING FORMATS

The UART supports several serial framing formats (*Figure 13*). The format is selected using control bits in the ENU, ENUR and ENUI registers.

The first format (1, 1a, 1b, 1c) for data transmission (CHL0 = 1, CHL1 = 0) consists of Start bit, seven Data bits (excluding parity) and 7/8, one or two Stop bits. In applications using parity, the parity bit is generated and verified by hardware.

The second format (CHL0 = 0, CHL1 = 0) consists of one Start bit, eight Data bits (excluding parity) and 7/8, one or two Stop bits. Parity bit is generated and verified by hardware.

The third format for transmission (CHL0 = 0, CHL1 = 1) consists of one Start bit, nine Data bits and 7/8, one or two Stop bits. This format also supports the UART "ATTENTION" feature. When operating in this format, all eight bits of TBUF and RBUF are used for data. The ninth data bit is transmitted and received using two bits in the ENU and ENUR registers, called XBIT9 and RBIT9. RBIT9 is a read only bit. Parity is not generated or verified in this mode.

For any of the above framing formats, the last Stop bit can be programmed to be 7/8th of a bit in length. If two Stop bits are selected and the 7/8th bit is set (selected), the second Stop bit will be 7/8th of a bit in length.

The parity is enabled/disabled by PEN bit located in the ENU register. Parity is selected for 7- and 8-bit modes only. If parity is enabled (PEN = 1), the parity selection is then performed by PSEL0 and PSEL1 bits located in the ENU register.

Note that the XBIT9/PSEL0 bit located in the ENU register serves two mutually exclusive functions. This bit programs the ninth bit for transmission when the UART is operating with nine data bits per frame. There is no parity selection in this framing format. For other framing formats XBIT9 is not needed and the bit is PSEL0 used in conjunction with PSEL1 to select parity.

The frame formats for the receiver differ from the transmitter in the number of Stop bits required. The receiver only requires one Stop bit in a frame, regardless of the setting of the Stop bit selection bits in the control register. Note that an implicit assumption is made for full duplex UART operation that the framing formats are the same for the transmitter and receiver.

UART Operation (Continued)

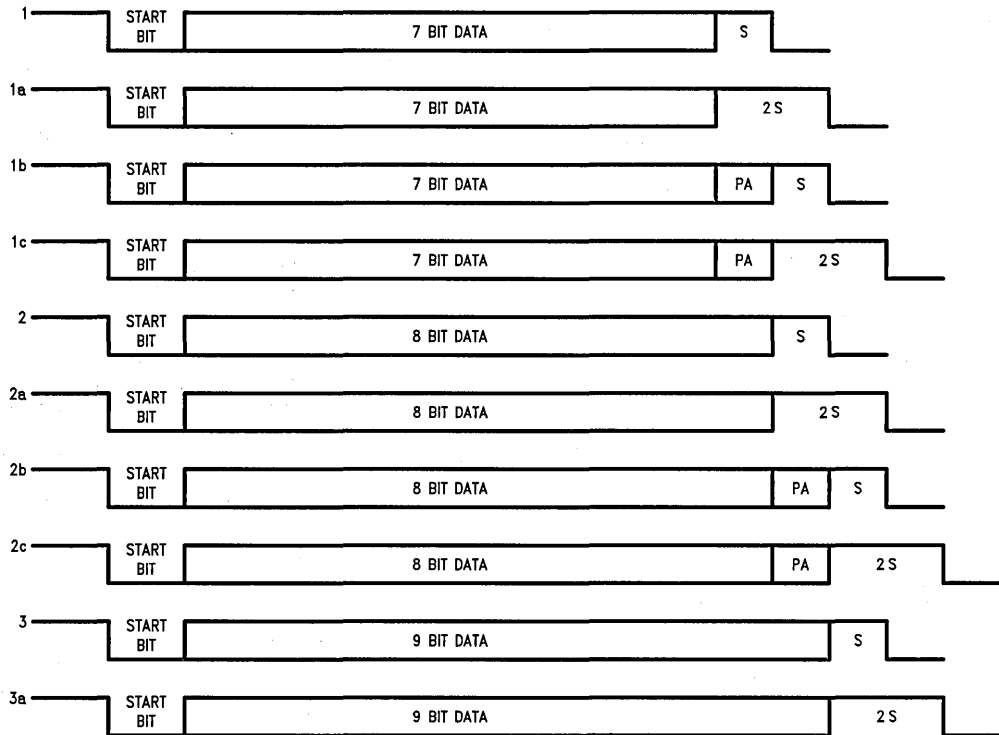


FIGURE 13. Framing Formats

TL/DD/10830-17

UART INTERRUPTS

The UART is capable of generating interrupts. Interrupts are generated on Receive Buffer Full and Transmit Buffer Empty. Both interrupts have individual interrupt vectors. Two bytes of program memory space are reserved for each interrupt vector. The two vectors are located at addresses 0xEC to 0xEF Hex in the program memory space. The interrupts can be individually enabled or disabled using Enable Transmit Interrupt (ETI) and Enable Receive Interrupt (ERI) bits in the ENUI register.

The interrupt from the Transmitter is set pending, and remains pending, as long as both the TBMT and ETI bits are set. To remove this interrupt, software must either clear the ETI bit or write to the TBUF register (thus clearing the TBMT bit).

The interrupt from the receiver is set pending, and remains pending, as long as both the RBFL and ERI bits are set. To remove this interrupt, software must either clear the ERI bit or read from the RBUF register (thus clearing the RBFL bit).

Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the UART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a

source selected in the PSR and BAUD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1-16 (increments of 0.5) prescaler and an 11-bit binary counter. (Figure 14) The divide factors are specified through two read/write registers shown in Figure 15. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

As shown in Table I, a Prescaler Factor of 0 corresponds to NO CLOCK. NO CLOCK condition is the UART power down mode where the UART clock is turned off for power saving purpose. The user must also turn the UART clock off when a different baud rate is chosen.

The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in Table I. There are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a x16 clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (Table II). Other baud rates may be created by using appropriate divisors. The x16 clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.

Baud Clock Generation (Continued)

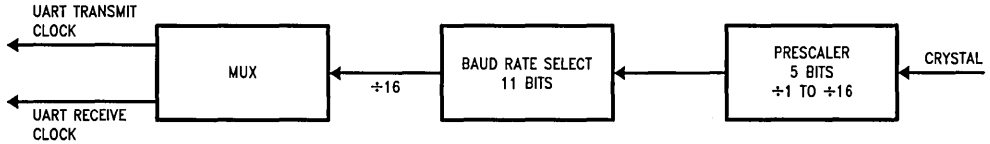
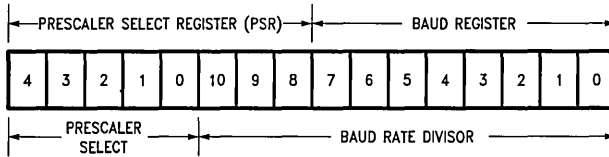


FIGURE 14. UART BAUD Clock Generation

TL/DD/10830-18



TL/DD/10830-19

FIGURE 15. UART BAUD Clock Divisor Registers

TABLE I. Prescaler Factors

Prescaler Select	Prescaler Factor
00000	NO CLOCK
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11001	13
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

TABLE II. Baud Rate Divisors (1.8432 MHz Prescaler Output)

Baud Rate	Baud Rate Divisor - 1 (N-1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5
38400	2

The entries in Table II assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 625k.

As an example, considering the Asynchronous Mode and a CKI clock of 4.608 MHz, the prescaler factor selected is:

$$4.608 / 1.8432 = 2.5$$

The 2.5 entry is available in Table I. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (Table II) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in Table II is 5.

$$N - 1 = 5 \text{ (} N - 1 \text{ is the value from Table II)}$$

$$N = 6 \text{ (} N \text{ is the Baud Rate Divisor)}$$

$$\text{Baud Rate} = 1.8432 \text{ MHz} / (16 \times 6) = 19200$$

The divide by 16 is performed because in the asynchronous mode, the input frequency to the UART is 16 times the baud rate. The equation to calculate baud rates is given below.

The actual Baud Rate may be found from:

$$BR = Fc / (16 \times N \times P)$$

Baud Clock Generation (Continued)

Where:

BR is the Baud Rate

Fc is the CKI frequency

N is the Baud Rate Divisor (Table II).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (Table I)

Note: In the Synchronous Mode, the divisor 16 is replaced by two if internal Baud Rate generator is used. Replaced by one if external clock is used.

Example:

Asynchronous Mode:

Crystal Frequency = 5 MHz

Desired baud rate = 9600

Using the above equation $N \times P$ can be calculated first.

$$N \times P = (5 \times 10^6) / (16 \times 9600) = 32.552$$

Now 32.552 is divided by each Prescaler Factor (Table II) to obtain a value closest to an integer. This factor happens to be 6.5 ($P = 6.5$).

$$N = 32.552 / 6.5 = 5.008 \quad (N = 5)$$

The programmed value (from Table II) should be 4 ($N - 1$).

Using the above values calculated for N and P:

$$BR = (5 \times 10^6) / (16 \times 5 \times 6.5) = 9615.384$$

$$\% \text{ error} = (9615.385 - 9600) / 9600 = 0.16$$

Effect of HALT/IDLE

The UART logic is reinitialized when either the HALT or IDLE modes are entered. This reinitialization sets the TBMT flag and resets all read only bits in the UART control and status registers. Read/Write bits remain unchanged. The Transmit Buffer (TBUF) is not affected, but the Transmit Shift register (TSFT) bits are set to one. The receiver registers RBUF and RSFT are not affected.

The μC will exit from the HALT/IDLE modes when the Start bit of a character is detected at the RDX (L3) pin. This feature is obtained by using the Multi-Input Wakeup scheme provided on the μC .

Before entering the HALT or IDLE modes the user program must select the Wakeup source to be on the RDX pin. This selection is done by setting bit 3 of WKEN (Wakeup Enable) register. The Wakeup trigger condition is then selected to be high to low transition. This is done via the WKEDG register (Bit 3 is zero.)

If the microcontroller is halted and crystal oscillator is used, the Wakeup signal will not start the chip running immediately because of the finite start up time requirement of the crystal oscillator. The idle timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the μC to execute code. The user has to consider this delay when data transfer is expected immediately after exiting the HALT mode.

Diagnostic

Bits CHARL0 and CHARL1 in the ENU register provide a loopback feature for diagnostic testing of the UART. When these bits are set to one, the following occur: The receiver input pin (RDX) is internally connected to the transmitter output pin (TDX); the output of the Transmitter Shift Regis-

ter is "looped back" into the Receive Shift Register input. In this mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the UART.

Note that the framing format for this mode is the nine bit format; one Start bit, nine data bits, and 7/8, one or two Stop bits. Parity is not generated or verified in this mode.

Attention Mode

The UART Receiver section supports an alternate mode of operation, referred to as ATTENTION Mode. This mode of operation is selected by the ATTN bit in the ENUR register. The data format for transmission must also be selected as having nine Data bits and either 7/8, one or two Stop bits.

The ATTENTION mode of operation is intended for use in networking the COP888CS with other processors. Typically in such environments the messages consists of device addresses, indicating which of several destinations should receive them, and the actual data. This Mode supports a scheme in which addresses are flagged by having the ninth bit of the data field set to a 1. If the ninth bit is reset to a zero the byte is a Data byte.

While in ATTENTION mode, the UART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the UART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if UART Receiver interrupts are enabled. The ATTN bit is also cleared automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

Operation of the UART Transmitter is not affected by selection of this Mode. The value of the ninth bit to be transmitted is programmed by setting XBIT9 appropriately. The value of the ninth bit received is obtained by reading RBIT9. Since this bit is located in ENUR register where the error flags reside, a bit operation on it will reset the error flags.

Comparator

The COP888CS contains one differential comparator, with a pair of inputs (positive and negative) and an output. Ports I1-I3 are used for the comparator. The following is the Port I assignment:

- I1 Comparator1 negative input
- I2 Comparator1 positive input
- I3 Comparator1 output

A Comparator Select Register (CMPSL) is used to enable the comparators, read the outputs of the comparator internally, and enable the output of the comparator to the pins. Two control bits (enable and output enable) and one result bit are associated with the comparator. The comparator result bit (CMP1RD) is read only bit which will read as zero if the comparator is not enabled. The Comparator Select Register is cleared with reset, resulting in the comparator being disabled. The comparator should also be disabled before entering either the HALT or IDLE modes in order to save power. The configuration of the CMPSL register is as follows:

Comparator (Continued)

CMPSL REGISTER (ADDRESS X'00B7)

The CMPSL register contains the following bits:

- CMP1EN Enable comparator 1
- CMP1RD Comparator 1 result (this is a read only bit, which will read as 0 if the comparator is not enabled)
- CMP10E Selects pin I3 as comparator 1 output provided that CMP1EN is set to enable the comparator

Unused	Unused	Unused	Unused	CMP10E	CMP1RD	CMP1EN	Unused
Bit 7				Bit 0			

Comparator outputs have the same spec as Ports L and G except that the rise and fall times are symmetrical.

Interrupts

The COP888CS supports a vectored interrupt scheme. It supports a total of fourteen interrupt sources. The following table lists all the possible COP888CS interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_c cycles to execute.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service rou-

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE-0yFF
	Reserved	for Future Use	0yFC-0yFD
(2)	External	Pin G0 Edge	0yFA-0yFB
(3)	Timer T0	Underflow	0yF8-0yF9
(4)	Timer T1	T1A/Underflow	0yF6-0yF7
(5)	Timer T1	T1B	0yF4-0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2-0yF3
	Reserved	for Future Use	0yF0-0yF1
(7)	UART	Receive	0yEE-0yEF
(8)	UART	Transmit	0yEC-0yED
(9)	Reserved		0yEA-0yEB
(10)	Reserved		0yE8-0yE9
(11)	Reserved		0yE6-0yE7
(12)	Reserved		0yE4-0yE5
(13)	Port L/Wakeup	Port L Edge	0yE2-0yE3
(14) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0-0yE1

y is VIS page, y ≠ 0.

Interrupts (Continued)

tine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block ($y \neq 0$).

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 16 shows the COP888CS Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit (**not accessible by the user**) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WATCHDOG

The COP888CS contains a WATCHDOG and clock monitor. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WATCHDOG consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WATCHDOG consists of writing a specific value to a WATCHDOG Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table III shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

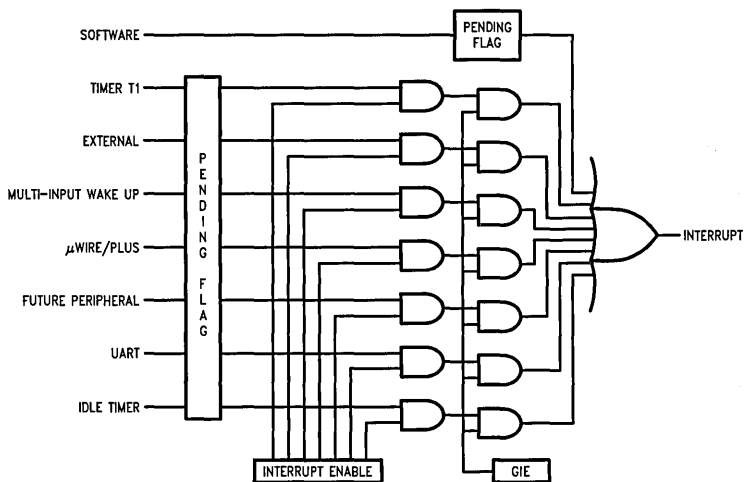


FIGURE 16. COP888CS Interrupt Block Diagram

TL/DD/10830-20

WATCHDOG (Continued)

Table IV shows the four possible combinations of lower and upper limits for the WATCHDOG service window. This flexibility in choosing the WATCHDOG service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE III. WATCHDOG Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE IV. WATCHDOG Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k–8k t_c Cycles
0	1	2k–16k t_c Cycles
1	0	2k–32k t_c Cycles
1	1	2k–64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the COP888CS can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WATCHDOG Operation

The WATCHDOG and Clock Monitor are disabled during reset. The COP888CS comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCHDOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. Table V shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WATCHDOG, the logic will pull the WDOUT (G1) pin low for an additional $16 t_c$ – $32 t_c$ cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CS will stop forcing the WDOUT output low.

The WATCHDOG service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WATCHDOG service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WATCHDOG will time out and WDOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following $16 t_c$ – $32 t_c$ clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10$ kHz—No clock rejection.

$1/t_c < 10$ Hz—Guaranteed clock rejection.

WATCHDOG AND CLOCK MONITOR SUMMARY

The following salient points regarding the COP888 WATCHDOG and CLOCK MONITOR should be noted:

- Both the WATCHDOG and Clock Monitor detector circuits are inhibited during RESET.
- Following RESET, the WATCHDOG and CLOCK MONITOR are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and Clock Monitor enable/disable option can only be changed once, during the initial WATCHDOG service following RESET.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.
- The Clock Monitor detector circuit is active during both the HALT and IDLE modes. Consequently, the COP888 inadvertently entering the HALT mode will be detected as a Clock Monitor error (provided that the Clock Monitor enable option has been selected by the program).
- With the single-pin R/C oscillator mask option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.
- With the crystal oscillator mask option selected, or with the single-pin R/C oscillator mask option selected and the CLKDLY bit set, the WATCHDOG service window will

WATCHDOG Operation (Continued)

be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.

- The IDLE timer T0 is not initialized with RESET.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the TOPND flag. The TOPND flag is set whenever the thirteenth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the TOPND flag.
- A hardware WATCHDOG service occurs just as the COP888 exits the IDLE mode. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following RESET, the initial WATCHDOG service (where the service window and the CLOCK MONITOR enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 2048 instruction cycles without causing a WATCHDOG error.

Detection of Illegal Conditions

The COP888CS can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F (Segment 0), 140 to 17F (Segment 1), and all other segments (i.e., Segments 3 . . . etc.) is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

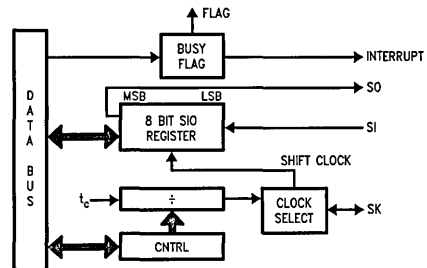
Thus, the chip can detect the following illegal conditions:

- Executing from undefined ROM
- Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before re-starting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CS to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E²PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 17 shows a block diagram of the MICROWIRE/PLUS logic.



TL/DD/10830-21

FIGURE 17. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table VI details the different clock rates that may be selected.

TABLE V. WATCHDOG Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

TABLE VI. MICROWIRE/PLUS
Master Mode Clock Select

SL1	SL0	SK
0	0	$2 \times t_c$
0	1	$4 \times t_c$
1	x	$8 \times t_c$

Where t_c is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The COP888CS may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 14* shows how two COP888CS microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock for the SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CS. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table VII summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table VII summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

The COP888CS allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE VII

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave

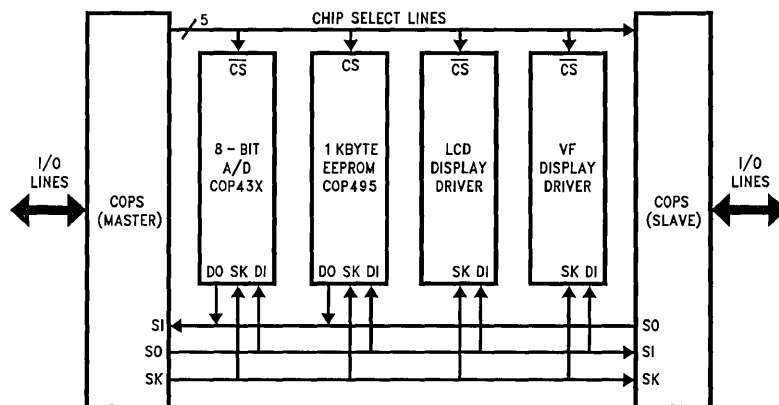


FIGURE 18. MICROWIRE/PLUS Application

TL/DD/10830--22

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address S/ADD REG	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads As All Ones)
xx80 to xxAF	Unused RAM Address Space (Reads Undefined Data)
xxB0 to xxB6	Reserved
xxB7	Comparator Select Register (CMPSL)
xxB8	UART Transmit Buffer (TBUF)
xxB9	UART Receive Buffer (RBUF)
xxBA	UART Control and Status Register (ENU)
xxBB	UART Receive Control and Status Register (ENUR)
xxBC	UART Interrupt and Clock Source Register (ENUI)
xxBD	UART Baud Register (BAUD)
xxBE	UART Prescale Select Register (PSR)
xxBF	Reserved for UART
xxC0 to xxC6	Reserved
xxC7	WATCHDOG Service Register (Reg:WDSVR)
xxC8	MIWU Edge Select Register (Reg:WKEDG)
xxC9	MIWU Enable Register (Reg:WKEN)
xxCA	MIWU Pending Register (Reg:WKPND)
xxCB	Reserved
xxCC	Reserved
xxCD to xxCF	Reserved

Address S/ADD REG	Contents
xxD0	Port L Data Register
xxD1	Port L Configuration Register
xxD2	Port L Input Pins (Read Only)
xxD3	Reserved for Port L
xxD4	Port G Data Register
xxD5	Port G Configuration Register
xxD6	Port G Input Pins (Read Only)
xxD7	Port I Input Pins (Read Only)
xxD8	Port C Data Register
xxD9	Port C Configuration Register
xxDA	Port C Input Pins (Read Only)
xxDB	Reserved for Port C
xxDC	Port D
xxDD to DF	Reserved for Port D
xxE0 to xxE5	Reserved for EE Control Registers
xxE6	Timer T1 Autoload Register T1RB Lower Byte
xxE7	Timer T1 Autoload Register T1RB Upper Byte
xxE8	ICNTRL Register
xxE9	MICROWIRE/PLUS Shift Register
xxEA	Timer T1 Lower Byte
xxEB	Timer T1 Upper Byte
xxEC	Timer T1 Autoload Register T1RA Lower Byte
xxED	Timer T1 Autoload Register T1RA Upper Byte
xxEE	CNTRL Control Register
xxEF	PSW Register
xxF0 to FB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	S Register
0100-013F	On-Chip RAM Bytes (64 bytes)

Reading memory locations 0070H-007FH (Segment 0) will return all ones. Reading unused memory locations 0080H-00AFH (Segment 0) will return undefined data. Reading unused memory locations 0140-017F (Segment 1) will return all ones. Reading memory locations from other Segments (i.e., Segment 2, Segment 3, ... etc.) will return all ones.

All reserved location reads undefined data.

Addressing Modes

The COP888CS has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CS. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the Load B Immediate instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from -31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
MemI	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	$A \leftarrow A + \text{Meml}$
ADC	A,Meml	ADD with Carry	$A \leftarrow A + \text{Meml} + C, C \leftarrow \text{Carry}$
SUBC	A,Meml	Subtract with Carry	$HC \leftarrow \text{Half Carry}$ $A \leftarrow A - \text{Meml} + C, C \leftarrow \text{Carry}$ $HC \leftarrow \text{Half Carry}$
AND	A,Meml	Logical AND	$A \leftarrow A \text{ and Meml}$
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if $(A \text{ and Imm}) = 0$
OR	A,Meml	Logical OR	$A \leftarrow A \text{ or Meml}$
XOR	A,Meml	Logical EXclusive OR	$A \leftarrow A \text{ xor Meml}$
IFEQ	MD,Imm	IF Equal	Compare MD and Imm, Do next if $MD = \text{Imm}$
IFEQ	A,Meml	IF Equal	Compare A and Meml, Do next if $A = \text{Meml}$
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if $A \neq \text{Meml}$
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if $A > \text{Meml}$
IFBNE	#	If B Not Equal	Do next if lower 4 bits of $B \neq \text{Imm}$
DRSZ	Reg	Decrement Reg., Skip if Zero	$\text{Reg} \leftarrow \text{Reg} - 1, \text{Skip if Reg} = 0$
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	$A \leftrightarrow \text{Mem}$
LD	A,Meml	LoaD A with Memory	$A \leftarrow \text{Meml}$
LD	B,Imm	LoaD B with Immed.	$B \leftarrow \text{Imm}$
LD	Mem,Imm	LoaD Memory Immed	$\text{Mem} \leftarrow \text{Imm}$
LD	Reg,Imm	LoaD Register Memory Immed.	$\text{Reg} \leftarrow \text{Imm}$
X	A, [B ±]	EXchange A with Memory [B]	$A \leftrightarrow [B], (B \leftarrow B \pm 1)$
X	A, [X ±]	EXchange A with Memory [X]	$A \leftrightarrow [X], (X \leftarrow X \pm 1)$
LD	A, [B ±]	LoaD A with Memory [B]	$A \leftarrow [B], (B \leftarrow B \pm 1)$
LD	A, [X ±]	LoaD A with Memory [X]	$A \leftarrow [X], (X \leftarrow X \pm 1)$
LD	[B ±],Imm	LoaD Memory [B] Immed.	$[B] \leftarrow \text{Imm}, (B \leftarrow B \pm 1)$
CLR	A	CLeaR A	$A \leftarrow 0$
INC	A	INCRement A	$A \leftarrow A + 1$
DEC	A	DECReamentA	$A \leftarrow A - 1$
LAI	A	LoaD A InDirect from ROM	$A \leftarrow \text{ROM (PU,A)}$
DCOR	A	Decimal CORrect A	$A \leftarrow \text{BCD correction of A (follows ADC, SUBC)}$
RRC	A	RotatE A Right thru C	$C \rightarrow A7 \rightarrow \dots \rightarrow A0 \rightarrow C$
RLC	A	RotatE A Left thru C	$C \leftarrow A7 \leftarrow \dots \leftarrow A0 \leftarrow C$
SWAP	A	SWAP nibbles of A	$A7 \dots A4 \leftrightarrow A3 \dots A0$
SC		Set C	$C \leftarrow 1, HC \leftarrow 1$
RC		Reset C	$C \leftarrow 0, HC \leftarrow 0$
IFC		IF C	If C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	$\text{SP} \leftarrow \text{SP} + 1, A \leftarrow [\text{SP}]$
PUSH	A	PUSH A onto the stack	$[\text{SP}] \leftarrow A, \text{SP} \leftarrow \text{SP} - 1$
VIS		Vector to Interrupt Service Routine	$\text{PU} \leftarrow [\text{VU}], \text{PL} \leftarrow [\text{VL}]$
JMPL	Addr.	Jump absolute Long	$\text{PC} \leftarrow \text{ii} (\text{ii} = 15 \text{ bits}, 0 \text{ to } 32\text{k})$
JMP	Addr.	Jump absolute	$\text{PC9} \dots 0 \leftarrow \text{i} (\text{i} = 12 \text{ bits})$
JP	Disp.	Jump relative short	$\text{PC} \leftarrow \text{PC} + \text{r} (\text{r} \text{ is } -31 \text{ to } +32, \text{ except } 1)$
JSRL	Addr.	Jump SubRoutine Long	$[\text{SP}] \leftarrow \text{PL}, [\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2, \text{PC} \leftarrow \text{ii}$
JSR	Addr	Jump SubRoutine	$[\text{SP}] \leftarrow \text{PL}, [\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2, \text{PC9} \dots 0 \leftarrow \text{i}$
JID		Jump InDirect	$\text{PL} \leftarrow \text{ROM (PU,A)}$
RET		RETurn from subroutine	$\text{SP} + 2, \text{PL} \leftarrow [\text{SP}], \text{PU} \leftarrow [\text{SP}-1]$
RETSK		RETurn and SKip	$\text{SP} + 2, \text{PL} \leftarrow [\text{SP}], \text{PU} \leftarrow [\text{SP}-1]$
RETI		RETurn from Interrupt	$\text{SP} + 2, \text{PL} \leftarrow [\text{SP}], \text{PU} \leftarrow [\text{SP}-1], \text{GIE} \leftarrow 1$
INTR		Generate an Interrupt	$[\text{SP}] \leftarrow \text{PL}, [\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2, \text{PC} \leftarrow \text{OFF}$
NOP		No OPeration	$\text{PC} \leftarrow \text{PC} + 1$

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

RPND	1/1
------	-----

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.	
	[B]	[X]			[B+, B-]	[X+, X-]
XA,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B, Imm				1/1		
LD B, Imm				2/2		
LD Mem, Imm	2/2		3/3		2/2	
LD Reg, Imm			2/3			
IFEQ MD, Imm			3/3			

(IF B < 16)

(IF B > 15)

* = > Memory location addressed by B or X or directly.

COP888CS Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP -15	JP -31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A,[B]	0
JP -14	JP -30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP -13	JP -29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A, #i	IFEQ A,[B]	2
JP -12	JP -28	LD 0F3, # i	DRSZ 0F3	X A, [X-]	X A,[B-]	IFGT A, #i	IFGT A,[B]	3
JP -11	JP -27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A, #i	ADD A,[B]	4
JP -10	JP -26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A, #i	AND A,[B]	5
JP -9	JP -25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A, #i	XOR A,[B]	6
JP -8	JP -24	LD 0F7, # i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	7
JP -7	JP -23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A, #i	IFC	8
JP -6	JP -22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md, #i	IFNE A, #i	IFNC	9
JP -5	JP -21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+], #i	INCA	A
JP -4	JP -20	LD 0FB, # i	DRSZ 0FB	LD A,[X-]	LD A,[B-]	LD [B-], #i	DECA	B
JP -3	JP -19	LD 0FC, # i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	POPA	C
JP -2	JP -18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	D
JP -1	JP -17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B], #i	RET	E
JP -0	JP -16	LD 0FF, # i	DRSZ 0FF	*	*	LD B, #i	RETI	F

COP888CS Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B, #0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP + 17	INTR	0
IFBIT 1,[B]	*	LD B, #0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP + 18	JP + 2	1
IFBIT 2,[B]	*	LD B, #0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP + 19	JP + 3	2
IFBIT 3,[B]	*	LD B, #0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP + 20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B, #0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP + 21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B, #0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP + 22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B, #09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP + 23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B, #08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP + 24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B, #07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP + 25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B, #06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP + 26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B, #05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP + 27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B, #04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP + 28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B, #03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP + 29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B, #02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP + 30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B, #01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP + 31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B, #00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP + 32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CS mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

- = 1 Crystal Oscillator (CKI/10)
 - G7 (CKO) is clock generator output to crystal/resonator
 - CKI is the clock input
- = 2 Single-pin RC controlled oscillator (CKI/10)
 - G7 is available as a HALT restart and/or general purpose input

OPTION 2: HALT

- = 1 Enable HALT mode
- = 2 Disable HALT mode

OPTION 3: OPTIONS BONDING

- = 1 44-Pin PLCC
- = 2 40-Pin DIP
- = 3 NA
- = 4 28-Pin DIP
- = 5 28-Pin SO

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefineable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC® via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS-232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884CG28D5PC	28 DIP	4.5V-5.5V	COP884CS
MHW-884CG28DWPC	28 DIP	2.5V-6.0V	COP884CS
MHW-888CG40D5PC	40 DIP	4.5V-5.5V	COP888CS
MHW-888CG40DWPC	40 DIP	2.5V-6.0V	COP888CS
MHW-888CG44D5PC	44 PLCC	4.5V-5.5V	COP888CS
MHW-888CG44DWPC	44 PLCC	2.5V-6.0V	COP888CS

Development Support (Continued)

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 FAX: (206) 882-1043

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM® PC/XT®, PC-AT® or compatible	424410527-001

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888CSMHXL-X	X = 1 : Crystal X = 3 : R/C	44 LDCC	Multi-Chip Module (MCM), UV Erasable	COP888CS
COP888CSMHD-X	X = 1 : Crystal X = 3 : R/C	40 DIP	MCM, UV Erasable	COP888CS
COP884CSMHD-X	X = 1 : Crystal X = 3 : R/C	28 DIP	MCM, UV Erasable	COP884CS
COP884CSMHEA-X	X = 1 : Crystal X = 3 : R/C	28 LCC	MCM (Same Footprint as 28 SO), UV Erasable	COP884CS

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for use with Scrambler Boards	COP884CSMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP Socket	COP888CSMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888CSMHXL
COP8-SCRM-SBX	MCM Scrambler Board for 28 LCC Socket	COP884CSMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP884CSMHD, COP888CSMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC Scrambler Board	COP888CSMEL, COP884CSMHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days

COP8780C/COP8781C/COP8782C

Single-Chip EPROM/OTP Microcontrollers

General Description

The COP8780C, COP8781C and COP8782C are members of the COP8™ 8-bit microcontroller family. They are fully static microcontrollers, fabricated using double-metal, double poly silicon gate microCMOS EPROM technology. These devices are available as UV erasable or One Time Programmable (OTP). These low cost microcontrollers are complete microcomputers containing all system timing, interrupt logic, EPROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with associated 16-bit autoreload/capture register, and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the device to the specific application. These devices operate over a voltage range of 4.5V to 6.0V. An efficient, regular instruction set operating at a 1 μs instruction cycle rate provides optimal throughput.

The COP8780C, COP8781C and COP8782C can be configured to EMULATE the COP880C, COP840C and COP820C microcontrollers.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS
- 4096 x 8 on-chip UV erasable or OTP EPROM
- EPROM security
- 128 or 64 bytes of on-chip RAM, user configurable
- Crystal, RC or External Oscillator, user configurable
- 1 μs instruction time (10 MHz clock)
- Low current drain
- Extra-low current static HALT mode

- Single supply operation: 4.5V to 6.0V
- 8-bit stack pointer (stack in RAM)
- 16-bit read/write timer operates in a variety of modes
 - PWM (Pulse Width Modulation) mode with 16-bit autoreload register
 - External Event Counter mode, with selectable edge
 - Input Capture mode (selectable edge) with 16-bit capture register
- Multi-source interrupt
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- Powerful instruction set, with most instructions single byte
- Many single byte, single cycle instructions
- BCD arithmetic instructions
- MICROWIRE/PLUS serial I/O
- Software selectable I/O options (TRI-STATE, push-pull, weak pull-up)
- Temperature ranges: -40°C to +85°C
- Schmitt trigger inputs on G port
- COP8780C EPROM Programming fully supported by National Semiconductor and Data I/O
- Packages:
 - 44 PLCC, OTP, Emulates COP880C, 36 I/O pins
 - 40 DIP, OTP, Emulates COP880C, 36 I/O pins
 - 28 DIP, OTP, Emulates COP820C/840C/881C, 24 I/O pins
 - 20 DIP, OTP, Emulates COP822C/842C, 16 I/O pins
 - 28 SO, 20 SO, OTP
 - 44 LDCC, UV Erasable
 - 40 CERDIP, 28 CERDIP, 20 CERDIP, UV Erasable
 - 20 Ceramic SO, 28 Ceramic SO, UV Erasable

Block Diagram

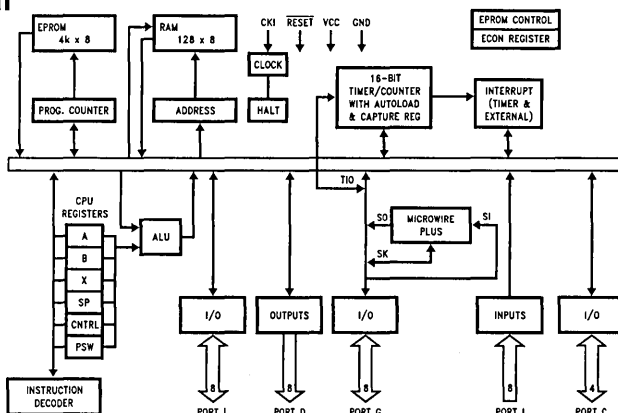


FIGURE 1

TL/DD/11299-1

COP8780C/COP8781C/COP8782C**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Programming Voltage V_{PP} (RESET pin) and ME (pin G6)	14V
Voltage at any Pin	-0.3V to $V_{CC} + 0.3V$

Total Current into V_{CC} Pin (Source)	50 mA
Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	-65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics COP87XXC; -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current				21	mA
CKI = 10 MHz (Note 2)	$V_{CC} = 6V, t_c = 1 \mu s$			10	μA
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$				
Input Levels					
RESET, CKI		0.9 V_{CC}			V
Logic High				0.1 V_{CC}	V
Logic Low					V
All Other Inputs		0.7 V_{CC}			V
Logic High				0.2 V_{CC}	V
Logic Low					V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V, V_{IN} = 0V$	40		250	μA
G Port Input Hysteresis	(Note 6)		0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Notes 4, 6) without Latchup (Room Temp)	Room Temp			±200	mA
RAM Retention Voltage, V_r (Note 5)		2.0			V
Input Capacitance	(Note 6)			7	pF
Load Capacitance on D2	(Note 6)			1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the crystal configurations. Halt test conditions: All Inputs tied to V_{CC} L, C, and G port I/O's configured as outputs and programmed low; D outputs programmed low; the window for UV erasable packages is completely covered with an opaque cover to prevent light from falling onto the die during HALT mode test. Parameter refers to HALT mode entered via setting bit 7 of the G Port data register.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typ). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Note 5: To maintain RAM integrity, the voltage must not be dropped or raised instantaneously.

Note 6: Parameter characterized but not tested.

COP8780C/COP8781C/COP8782C**AC Electrical Characteristics** $-40^{\circ}\text{C} < T_A < +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal/Resonator or External Clock R/C Oscillator Mode	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
CKI Clock Duty Cycle (Note 7) Rise Time (Note 7) Fall Time (Note 7)	$f_r = \text{Max}$	40		60	%
	$f_r = 10\text{ MHz Ext Clock}$			12	ns
	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs t_{SETUP} t_{HOLD}	$V_{CC} \geq 4.5\text{V}$	200			ns
	$V_{CC} \geq 4.5\text{V}$	60			ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$C_L = 100\text{ pF}, R_L = 2.2\text{ k}\Omega$				
	$V_{CC} \geq 4.5\text{V}$			0.7	μs
	$V_{CC} \geq 4.5\text{V}$			1	μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20			ns
		56			ns
				220	ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1			t_c
		1			t_c
		1			t_c
		1			t_c
		1			t_c
Reset Pulse Width		1.0			μs

Note 7: Parameter guaranteed by design, but not tested.

t_c = Instruction Cycle Time.

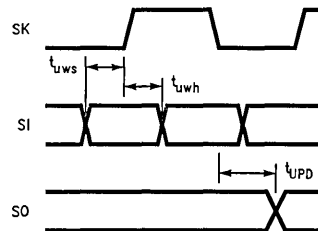
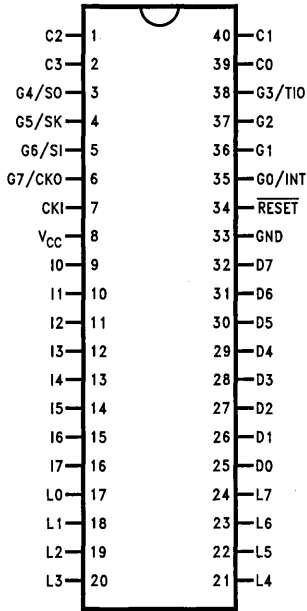
Timing Diagram

FIGURE 2. MICROWIRE/PLUS Timing

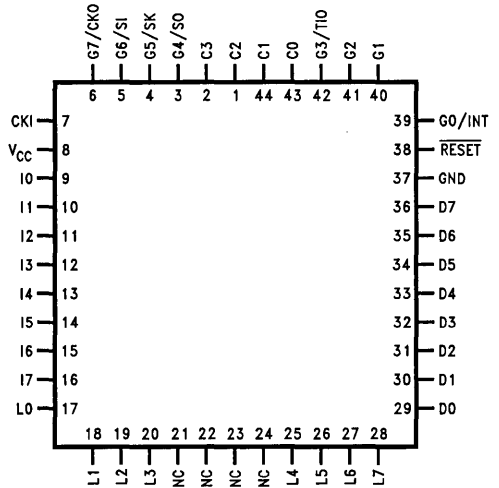
TL/DD/10802-2

Connection Diagrams



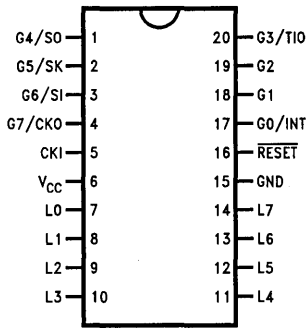
Top View
COP8780CN, COP8780CJ

TL/DD/11299-3



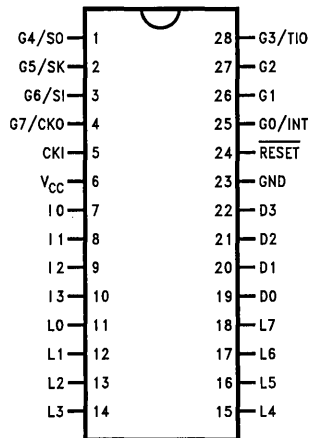
Top View
COP8780CV, COP8780CEL

TL/DD/11299-4



Top View
COP8782CN, COP8782CWM,
COP8782CJ, COP8782CMC

TL/DD/11299-5



Top View
COP8781CN, COP8781CWM
COP8781CJ, COP8781CMC

TL/DD/11299-6

FIGURE 3. COP8780C Connection Diagrams

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT I is an 8-bit Hi-Z input port.

PORT L is an 8-bit I/O port.

PORT C is a 4-bit I/O port.

Three memory locations are allocated for the L, G and C ports, one each for data register, configuration register and the input pins. Reading bits 4–7 of the C-Configuration register, data register, and input pins returns undefined data.

There are two registers associated with the L and C ports: a data register and a configuration register. Therefore, each L and C I/O bit can be individually configured under software control as shown below:

Config.	Data	Ports L and C Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

On the 20- and 28-pin parts, it is recommended that all bits of Port C be configured as outputs to minimize current.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated with the G port: a data register and a configuration register. Therefore, each G port bit can be individually configured under software control as shown below:

Config.	Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Since G6 and G7 are input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. The device will be placed in the HALT mode by writing a one to the G7 bit in the G-port data register.

Six pins of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE/PLUS serial data output)

G5 SK (MICROWIRE/PLUS clock I/O)

G6 SI (MICROWIRE/PLUS serial data input)

G7 CKO crystal oscillator output (selected by programming the ECON register) or HALT Restart/general purpose input

Pins G1 and G2 currently do not have any alternate functions.

PORT D is an 8-bit output port that is preset high when RESET goes low. Care must be exercised with the D2 pin operation. At RESET, the external loads on this pin must ensure that the output voltages stay above 0.7 V_{CC} to pre-

vent the chip from entering special modes. Also, keep the external loading on D2 to less than 1000 pF.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 8-bit Accumulator register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack in RAM. The SP must be initialized with software (usually to RAM address 06F Hex with 128 bytes of on-chip RAM selected, or to RAM address 02F Hex with 64 bytes of on-chip RAM selected). The SP is used with the subroutine call and return instructions, and with the interrupts.

B, X and SP registers are mapped into the on-chip RAM. The B and X registers are used to address the on-chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns.

PROGRAM MEMORY

The COP8780C contains 4096 bytes of UV erasable or OTP EPROM memory. This memory is mapped in the program memory address space from 0000 to 0FFF Hex. The program memory may contain either instructions or data constants, and is addressed by the 15-bit program counter (PC). The program memory can be indirectly read by the LAID (Load Accumulator Indirect) instruction for table lookup of constant data.

All locations in the EPROM program memory will contain 0FF Hex (all 1's) after the COP8780C is erased. OTP parts are shipped with all locations already erased to 0FF Hex. Unused EPROM locations should always be programmed to 00 Hex so that the software trap can be used to halt run-away program operation.

The COP8780C can be configured to inhibit external reads of the program memory. This is done by programming the security bit in the ECON (EPROM configuration) register to zero. See the ECON REGISTER section for more details.

DATA MEMORY

The data memory address space includes on-chip RAM, I/O, and registers. Data memory is addressed directly by instructions, or indirectly by means of the B, X, or SP pointers. The COP8780C can be configured to have either 64 or 128 bytes of RAM, depending on the value of the "RAM SIZE" bit in the ECON (EPROM CONFIGURATION) register. The sixteen bytes of RAM located at data memory address 0F0–0FF are designated as "registers". These sixteen registers can be decremented and tested with the DRSZ (Decrement Register and Skip if Zero) instruction.

Functional Description (Continued)

The three pointers X, B, and SP are memory mapped into this register address space at addresses 0FC, 0FE, and 0FD respectively. The remaining registers are available for general usage.

Any bit of data memory can be directly set, reset or tested. All of the I/O registers and control registers (except A and PC) are memory mapped. Consequently, any of the I/O bits or control register bits can be directly and individually set, reset, or tested.

ECON (EPROM CONFIGURATION) REGISTER

The ECON register is used to configure the user selectable clock, security, and RAM size options. The register can be programmed and read only in EPROM programming mode. Therefore, the register should be programmed at the same time as the program memory locations 0000 through 0FFF Hex. UV erasable parts are shipped with 0FF Hex in this register while the OTP parts are shipped with 07F Hex in this register. Erasing the EPROM program memory also erases the ECON register.

The COP8780C has a security feature which, when enabled, prevents reading of the EPROM program memory. The security bit in the ECON register determines whether security is enabled or disabled. If the security option is enabled, then any attempt to externally read the contents of the EPROM will result in the value E0 Hex being read from all program memory locations. If the security option is disabled, the contents of the internal EPROM may be read. The ECON register is readable regardless of the state of the security bit.

The format of the COP8780C ECON register is as follows:

TABLE I

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	X	SECURITY	CKI 2	CKI 1	X	RAM SIZE	X

Bit 7 = 0 Must be programmed to zero.

Bit 6 = X Don't care.

Bit 5 = 1 Security disabled. EPROM read and write are allowed.

= 0 Security enabled. EPROM read and write are not allowed.

Bits 4,3

= 1,1 External CKI option selected.

= 0,1 Not allowed.

= 1,0 RC oscillator option selected.

= 0,0 Crystal oscillator option selected.

Bit 2 = X Don't care.

Bit 1 = 1 Selects 128 byte RAM option. This emulates COP840 and COP880.

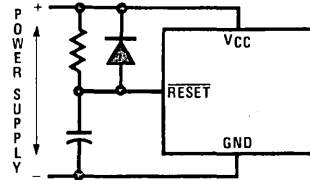
= 0 Selects 64 byte RAM option. This emulates COP820.

Bit 0 = X Don't care.

RESET

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the Ports L, G and C are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L, G and C are cleared.

The external RC network shown in *Figure 4* should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.



TL/DD/11299-7

RC \geq 5X Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

OSCILLATOR CIRCUITS

Figure 5 shows the three clock oscillator configurations available for the device. The CKI 1 and CKI 2 bits in the ECON register are used to select the clock option. See the ECON REGISTER section for more details.

A. Crystal Oscillator

The device can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table II shows the component values required for various standard crystal frequencies.

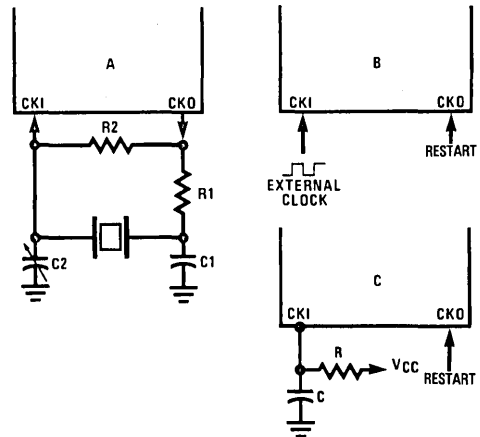
B. External Oscillator

CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels. In External oscillator mode, G7 is available as a general purpose input and/or HALT restart control.

C. R/C Oscillator

CKI can be configured as a single pin RC controlled oscillator. In RC oscillator mode, G7 is available as a general purpose input and/or HALT restart control.

Table III shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/11299-8

FIGURE 5. Crystal, External and R-C Connection Diagrams

Functional Description (Continued)

TABLE II. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	10	$V_{CC} = 5\text{V}$
0	1	30	30–36	4	$V_{CC} = 5\text{V}$

TABLE III. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5\text{V}$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5\text{V}$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5\text{V}$

Note: (R/C Oscillator Configuration): $3\text{k} \leq R \leq 200\text{k}$, $50\text{ pF} \leq C \leq 200\text{ pF}$.

HALT MODE

The device supports a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. (Stopping the clock input will draw more current than setting the G7 data bit.) In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to V_r (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the RESET or by the G7 pin. A low on the RESET line reinitializes the microcontroller and starts execution from address 0000H. In external and RC oscillator modes, a low to high transition on the G7 pin also causes the microcontroller to come out of the HALT mode. Execution resumes at the address following the HALT instruction. Except for the G7 data bit, which gets reset, all RAM locations retain the values they had prior to execution of the "HALT" instruction. It is required that the first instruction following the "HALT" instruction be a "NOP" in order to synchronize the clock.

INTERRUPTS

The device has a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer underflow or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to

select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupts respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPNB bits signal which interrupt is pending. After an interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

Functional Description (Continued)

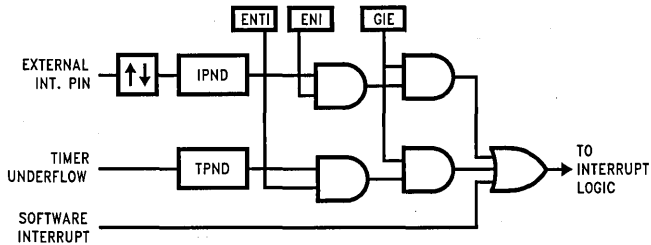


FIGURE 6. Interrupt Block Diagram

TL/DD/11299-9

DETECTION OF ILLEGAL CONDITIONS

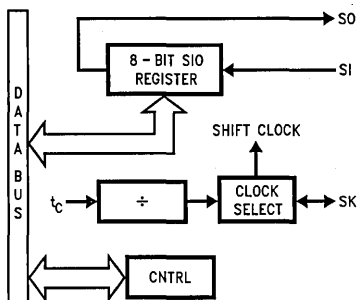
The device incorporates a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and "brown out" voltage drop situations. Specifically, it detects cases of executing out of undefined EPROM area and unbalanced stack situations.

Reading an undefined EPROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also "00". Thus a program accessing undefined EPROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the device grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined EPROM location and will trigger a software interrupt.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the device to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMS, etc.) and with other microcontrollers which support the MICROWIRE/PLUS interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). Figure 7 shows the block diagram of the MICROWIRE/PLUS interface.



TL/DD/11299-10

FIGURE 7. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS interface with the internal clock source is called the Master mode of operation. Operating the MICROWIRE/PLUS interface with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table IV details the different clock rates that may be selected.

TABLE IV

S1	S0	SK Cycle Time
0	0	2t _c
0	1	4t _c
1	x	8t _c

where,

t_c is the instruction cycle time.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The device may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. Figure 8 shows how two device microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the device. The MICROWIRE/PLUS Master always initiates all data exchanges (Figure 8). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions on the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table V summarizes the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL

Functional Description (Continued)

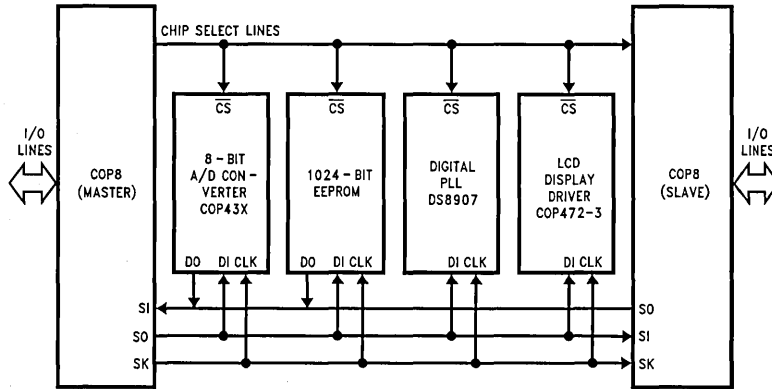


FIGURE 8. MICROWIRE/PLUS Application

TL/DD/11299-11

bit in the CNTRL register enables the SO and SK functions on the G Port. The SK pin must be selected as an input and the SO pin selected as an output pin by appropriately setting up the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation. The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated (Figure 8).

TABLE V

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The device has a powerful 16-bit timer with an associated 16-bit register enabling it to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table VI details various timer operating modes and their requisite control settings.

MODE 1. TIMER WITH AUTO-LOAD REGISTER

In this mode of operation, the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allows the generation of square-wave outputs or pulse width modulated outputs under software control (Figure 9).

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt (Figure 9).

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge (Figure 10).

TABLE VI. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter w/Auto-Load Reg.	Timer Underflow	TIO Pos. Edge
0 0 1	External Counter w/Auto-Load Reg.	Timer Underflow	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer w/Auto-Load Reg.	Timer Underflow	t _c
1 0 1	Timer w/Auto-Load Reg./Toggle TIO Out	Timer Underflow	t _c
1 1 0	Timer w/Capture Register	TIO Pos. Edge	t _c
1 1 1	Timer w/Capture Register	TIO Neg. Edge	t _c

Functional Description (Continued)

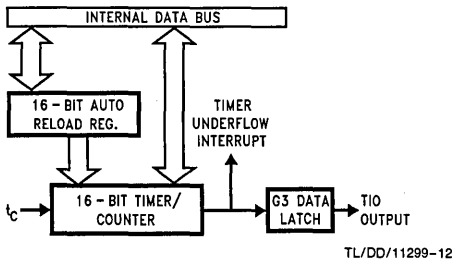


FIGURE 9. Timer/Counter Auto Reload Mode Block Diagram

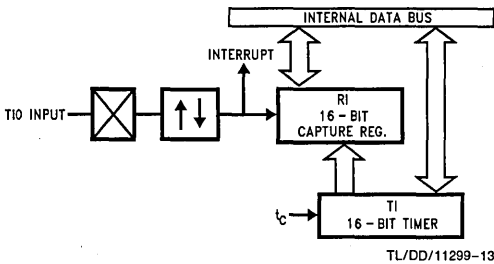


FIGURE 10. Timer Capture Mode Block Diagram

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.

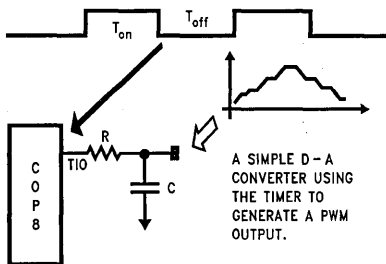


FIGURE 11. Timer Application

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE/PLUS control register contains the following bits:

- S1 & S0 Select the MICROWIRE/PLUS clock divide-by
- IEDG External interrupt edge polarity select (0 = rising edge, 1 = falling edge)
- MSEL Enable MICROWIRE/PLUS functions SO and SK
- TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)
- TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)
- TC2 Selects the capture mode
- TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0	
Bit 7							Bit 0	

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable
- ENI External interrupt enable
- BUSY MICROWIRE/PLUS busy shifting
- IPND External interrupt pending
- ENTI Timer interrupt enable
- TPND Timer interrupt pending
- C Carry Flag
- HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE
Bit 7							Bit 0

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for the device. The operand is the memory location addressed by the B register or X register.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory location for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of -31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, all 15 bits of PC are used.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

RAM Select	Address	Contents
64 On-Chip RAM Bytes Selected by ECON reg.	00–2F 30–7F	48 On-Chip RAM Bytes Unused RAM Address Space (Reads as all 1's)
128 On-Chip RAM Bytes Selected by ECON reg.	00–6F 70–7F	112 On-chip RAM Bytes Unused RAM Address Space (Reads as all 1's)
	80 to BF	Expansion Space for On-Chip EERAM
	C0 to CF	Expansion Space for I/O and Registers
	D0 to DF	On-Chip I/O and Registers
	D0	Port L Data Register
	D1	Port L Configuration Register
	D2	Port L Input Pins (Read Only)
	D3	Reserved for Port L
	D4	Port G Data Register
	D5	Port G Configuration Register
	D6	Port G Input Pins (Read Only)
	D7	Port I Input Pins (Read Only)
	D8	Port C Data Register
	D9	Port C Configuration Register
	DA	Port C Input Pins (Read Only)
	DB	Reserved for Port C
	DC	Port D Data Register
	DD–DF	Reserved for Port D
	E0 to EF	On-Chip Functions and Registers
	E0–E7	Reserved for Future Parts
	E8	Reserved
	E9	MICROWIRE/PLUS Shift Register
	EA	Timer Lower Byte
	EB	Timer Upper Byte
	EC	Timer Autoload Register Lower Byte
	ED	Timer Autoload Register Upper Byte
	EE	CNTRL Control Register
	EF	PSW Register
	F0 to FF	On-Chip RAM Mapped as Registers
	FC	X Register
	FD	SP Register
	FE	B Register

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
Meml	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set

ADD ADC SUBC AND OR XOR IFEQ IFGT IFBNE DRSZ SBIT RBIT IFBIT	add add with carry subtract with carry Logical AND Logical OR Logical Exclusive-OR IF equal IF greater than IF B not equal Decrement Reg. ,skip if zero Set bit Reset bit If bit	A ← A + Meml A ← A + Meml + C, C ← Carry HC ← Half Carry A ← A + Meml + C, C ← Carry HC ← Half Carry A ← A and Meml A ← A or Meml A ← A xor Meml Compare A and Meml, Do next if A = Meml Compare A and Meml, Do next if A > Meml Do next if lower 4 bits of B ≠ Imm Reg ← Reg - 1, skip if Reg goes to 0 1 to bit, Mem (bit= 0 to 7 immediate) 0 to bit, Mem If bit, Mem is true, do next instr.
X LD A LD mem LD Reg	Exchange A with memory Load A with memory Load Direct memory Immed. Load Register memory Immed.	A ↔ Mem A ← Meml Mem ← Imm Reg ← Imm
X X LD A LD A LD M	Exchange A with memory [B] Exchange A with memory [X] Load A with memory [B] Load A with memory [X] Load Memory Immediate	A ↔ [B] (B ← B ± 1) A ↔ [X] (X ← X ± 1) A ← [B] (B ← B ± 1) A ← [X] (X ← X ± 1) [B] ← Imm (B ← B ± 1)
CLRA INCA DECA LAID DCORA RRCA SWAPA SC RC IFC IFNC	Clear A Increment A Decrement A Load A indirect from ROM DECIMAL CORRECT A ROTATE A RIGHT THRU C Swap nibbles of A Set C Reset C If C If not C	A ← 0 A ← A + 1 A ← A - 1 A ← ROM(PU,A) A ← BCD correction (follows ADC, SUBC) C → A7 → ... → A0 → C A7 ... A4 ↔ A3 ... A0 C ← 1, HC ← 1 C ← 0, HC ← 0 If C is true, do next instruction If C is not true, do next instruction
JMPL JMP JP JSRL JSR JID RET RETSK RETI INTR NOP	Jump absolute long Jump absolute Jump relative short Jump subroutine long Jump subroutine Jump indirect Return from subroutine Return and Skip Return from Interrupt Generate an interrupt No operation	PC ← ii (ii = 15 bits, 0 to 32k) PC11..0 ← i (i = 12 bits) PC ← PC + r (r is -31 to +32, not 1) [SP] ← PL,[SP-1] ← PU,SP-2,PC ← ii [SP] ← PL,[SP-1] ← PU,SP-2,PC11..0 ← i PL ← ROM(PU,A) SP + 2,PL ← [SP],PU ← [SP-1] SP + 2,PL ← [SP],PU ← [SP-1],Skip next instruction SP + 2,PL ← [SP],PU ← [SP-1],GIE ← 1 [SP] ← PL,[SP-1] ← PU,SP-2,PC ← OFF PC ← PC + 1

Bits 7-4

OPCODE LIST

Bits 3-0

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0,[B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A,[B]	IFBIT 1,[B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A,[B]	IFBIT 2,[B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X-]	X A, [B-]	IFGT A, #i	IFGT A,[B]	IFBIT 3,[B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	AND A, #i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A,[B]	IFBIT 6,[B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	IFBIT 7,[B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2,[B]	RBIT 2,[B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X-]	LD A, [B-]	LD [B-], #i	DECA	SBIT 3,[B]	RBIT 3,[B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A, Md	*	SBIT 4,[B]	RBIT 4,[B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP + 13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5,[B]	RBIT 5,[B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP + 14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00-0EFF	JMP 0E00-0EFF	JP + 31	JP + 15	E
JP -0	JP -16	LD 0FF, #1	DRSZ 0FF	*	*	*	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

Arithmetic Instructions (Bytes/Cycles)

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions (Bytes/Cycles)

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr	
	[B]	[X]			[B+, B-]	[X+, X-]
X A,*	1/1	1/3	2/3		1/2	1/3
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3
LD B,Imm				1/1		
LD B,Imm				2/3		
LD Mem,Imm			3/3		2/2	
LD Reg,Imm				2/3		

(If B < 16)
(If B > 15)

* => Memory location addressed by B or X or directly.

Instructions Using A & C

Instructions	Bytes/Cycles
CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

Instructions	Bytes/Cycles
JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

BYTES and CYCLES per INSTRUCTION (Continued)

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Programming the COP8780C

Programming of the COP87XXC single-chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 FAX: (206) 882-1043

The National Semiconductor Duplicator Board is a stand alone programmer capable of supporting all COP8780C package types when combined with available adaptor boards (Scrambler Boards). The duplicator works in conjunction with a pre-programmed source EPROM containing the application program. (The source EPROM may be programmed via a standard programmer.) The duplicator board essentially copies the information from the source EPROM into the COP8780C program memory.

In addition to the application program stored in locations 0000 through 0FFF Hex, the source EPROM must contain a value for the ECON register at location 1FFF Hex. The following tables provide examples of some ECON register values. For more detailed information refer to the ECON REGISTER section.

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-87A	Duplicator Board with COP87XX Scrambler for 28 DIP, 28 SO, and 40 DIP	COP8781CN, COP8781CJ, COP8781CWM, COP8781CMC, COP8780CN, COP8780CJ
COP8-PRGM-87B	Duplicator Board with COP87XX Scrambler for 20 DIP, 20 SO, and 44 PLCC/LDCC	COP8780CV, COP8780CEL, COP8782CN, COP8782CWM, COP8782C5, COP8782CMC

EPROM Security Enabled

RAM Memory	External CKI	RC Oscillator	Crystal Oscillator
64 Bytes	18	10	00
128 Bytes	1A	12	02

EPROM Security Disabled

RAM Memory	External CKI	RC Oscillator	Crystal Oscillator
64 Bytes	38	30	20
128 Bytes	3A	32	22

ERASING THE COP8780C EPROM

The COP8780C EPROM program memory is erased by exposing the transparent window on the UV erasable packages to an ultraviolet light source. Erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000Å range.

After programming, opaque labels should be placed over the window of the device to prevent functional failure due to the generation of photo currents, erasure and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the COP8780C is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (UV intensity × exposure time) for erasure should be a minimum of 30W-sec/cm².

The COP8780C device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

Programming the COP8780C (Continued)

Minimum COP8780C Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time* (Minutes)
15,000	36
10,000	50
8,500	60

*Does not include light intensity ramp up time.

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

Common symptoms of insufficient erasure are:

- Inability to be programmed.
- Operational malfunctions associated with V_{CC}, temperature, or clock frequency.
- Loss of data in program memory.
- A change in configuration values in the ECON register.

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kbytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges, or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes, and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy-to-use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2 kBaud serial link keeps typical program download to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulator
MHW-880C28D5PC	28 DIP	4.5V-5.5V	COP820C, 840C, 881C, 8781C
MHW-880C28DWPC	28 DIP	2.5V-6.0V	COP820C, 840C, 881C, 8781C
MHW-880C40D5PC	40 DIP	4.5V-5.5V	COP880C, 8780C
MHW-880C28DWPC	40 DIP	2.5V-6.0V	COP880C, 8780C
MHW-880C44D5PC	44 PLCC	4.5V-5.5V	COP880C, 8780C
MHW-880C44DWPC	44 PLCC	2.5V-6.0V	COP880C, 8780C

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM® PC-XT®, PC-AT®, or compatible	424410527-001

CROSS REFERENCE TABLE

The following cross reference table lists the COP800 devices which can be emulated with the COP87XXC single-chip, form fit and function emulators.

Single-Chip Emulator Selection Table

Device Number	Package	Description	Emulates
COP8780CV	44 PLCC	One Time Programmable (OTP)	COP880C
COP8780CEL	44 LDCC	UV Erasable	COP880C
COP8780CN	40 DIP	OTP	COP880C
COP8780CJ	40 DIP	UV Erasable	COP880C
COP8781CN	28 DIP	OTP	COP881C, COP840C, COP820C
COP8781CJ	28 DIP	UV Erasable	COP881C, COP840C, COP820C
COP8781CWM	28 SO	OTP	COP881C, COP840C, COP820C
COP8781CMC	28 SO	UV Erasable	COP881C, COP840C, COP820C
COP8782CN	20 DIP	OTP	COP842C, COP822C
COP8782CJ	20 DIP	UV Erasable	COP842C, COP822C
COP8782CWM	20 SO	OTP	COP842C, COP822C
COP8782CMC	20 SO	UV Erasable	COP842C, COP822C

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user has questions, he can leave messages on our electronic bulletin board.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 baud

Setup: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs. 7 Days



COP842CMH Microcontroller Emulator

General Description

The COP842CMH hybrid emulator is a member of the COPSTM microcontroller family. The device is a two chip system in a dual cavity 20-pin DIP package. Within the package is the COP842C and a UV-erasable 8k EPROM with port recreation logic. Code executes out of the EPROM. The part contains a transparent window which allows the EPROM to be erased and re-programmed. The COP842CMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSTM serial I/O, and a 16-bit timer/counter supporting three modes (PWM generation, External Event counter, and Input Capture). Each I/O pin has software selectable configurations. The COP842CMH operates over a voltage range of 4.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

The COP842CMH is primarily intended as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

Features

- Form fit and function emulation device for the COP842C/COP822C
- Fully static CMOS
- 1 μ s instruction time (10 MHz clock)
- 8191 bytes EPROM/128 bytes RAM
- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - External interrupt with selectable edge
 - Timer/capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions are single byte
- BCD arithmetic instructions
- MICROWIRE/PLUS serial I/O
- 20-pin package
- 16 input/output pins
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Real time emulation and full program debug offered by National's Development Systems

Ordering Information

Hybrid Emulator	Package Type	Part Emulated
COP842CMHD-x	20-Pin DIP	COP822C-XXX/N COP842C-XXX/N

x = 1, 2, or 3. See "Oscillator Circuits".

1 = Crystal \div 10

2 = External \div 10

3 = R/C \div 10

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Reset (V_{PP}) and G6 (ME)	-0.3V to 14V
Voltage at any other Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	50 mA

Total Current Out of GND Pin (Sink)	60 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics -0°C < T_A < +70°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current				19	mA
CKI = 10 MHz	$V_{CC} = 6.0V, t_c = 1 \mu s$			14	mA
CKI = 5 MHz (Note 2)	$V_{CC} = 6.0V, t_c = 2 \mu s$				
HALT Current (Note 3)	$V_{CC} = 6.0V, CKI = 0 MHz$		500		μA
INPUT LEVELS					
Reset, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
All Others					
Source (Weak Pull-up Mode)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
Source (Push-pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin				3	mA
Maximum Input Current without Latchup (Note 4)	Room Temp			± 100	mA
RAM Retention Voltage, V_R	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

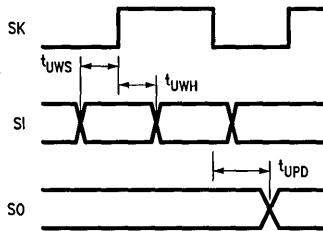
Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. HALT test conditions: L and G ports are at TRI-STATE and tied to ground, EPROM window covered.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

AC Electrical Characteristics $-0^{\circ}\text{C} < T_A < +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal/Resonator (Div-by 10)	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
R/C Oscillator Mode (div-by 10)	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
CKI Clock Duty Cycle (Note 5)		40		60	%
Rise Time (Note 5)	$f_r = 10\text{ MHz ext clock}$			12	ns
Fall Time (Note 5)	$f_r = 10\text{ MHz ext clock}$			8	ns
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			tC
Interrupt Input Low Time		1			tC
Timer Input High Time		1			tC
Timer Input Low Time		1			tC
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).



TL/DD/10717-4

FIGURE 1. MICROWIRE Timing Diagram

Pin Descriptions

ON CHIP I/O

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO).

RESET is the master reset input.

Port L is an 8-bit I/O port.

There are two registers associated with the L port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown in the following table:

Configure	Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Three RAM data memory locations are allocated for the L port, one for the data register, one for the configuration register and one for the input pins.

Port G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated with the G port: a data register and a configuration register. Therefore, each G port bit can be individually configured under software control as shown below:

Configure	Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Three RAM data memory locations are allocated for the G port, one for the data register, one for the configuration register, and one for the input pins.

Since G6 and G7 are input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the COP842CMH will be placed in the HALT mode by setting the G7 data bit.

Six Port G pins have alternate functions:

- G0 INT (External Interrupt)
- G3 TIO (Timer/Counter I/O)
- G4 SO (MICROWIRE Serial Data Output)

G5 SK (MICROWIRE Serial Clock)

G6 SI (MICROWIRE Serial Data Input)

G7 CKO Crystal Oscillator Output (Selected by Mask Option) or HALT Restart Input (General Purpose Input)

TABLE I. COP842CMH Pinouts for 20-Pin Package

Port	Type	Alternate Function	20-Pin DIP
L0	I/O		7
L1	I/O		8
L2	I/O		9
L3	I/O		10
L4	I/O		11
L5	I/O		12
L6	I/O		13
L7	I/O		14
G0	I/O	INT	17
G1	I/O		18
G2	I/O		19
G3	I/O	TIO	20
G4	I/O	SO	1
G5	I/O	SK	2
G6	I	SI	3
G7	I/CKO	HALT RESTART	4
VCC			6
GND			15
CKI			5
RESET			16

Connection Diagram

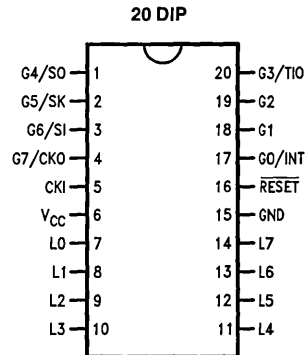


FIGURE 2. COP842CMH Connection Diagram

TL/DD/10717-2

Note: See COP842C datasheet for complete details.

Connection Diagram (Continued)

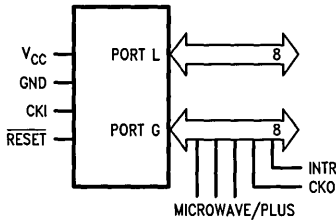


FIGURE 3. COP842CMH Function Diagram
TL/DD/10717-3

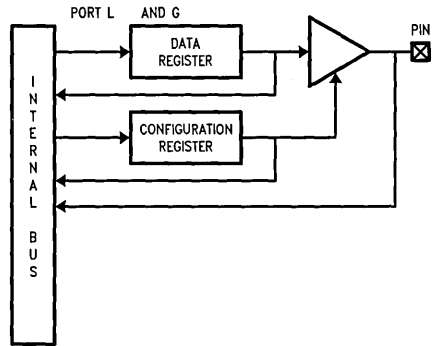


FIGURE 4. I/O Port Configurations
TL/DD/10717-5

Oscillator Circuits

A. CRYSTAL OSCILLATOR—COP842CMHD-1

By selecting CKO as a clock output, CKI and CKO can be connected to make a crystal controlled oscillator. See Table II for value of R & C.

B. EXTERNAL OSCILLATOR—COP842CMHD-2

CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels. CKO (G7) is available as a general purpose input and/or Halt Control.

C. R/C OSCILLATOR—COP842CMHD-3

CKI is configured as a single pin R/C controlled Schmitt trigger oscillator. CKO (G7) is available as a general purpose input and/or HALT restart control.

Table III shows the variation in the oscillator frequencies as functions of the component (R and C) values.

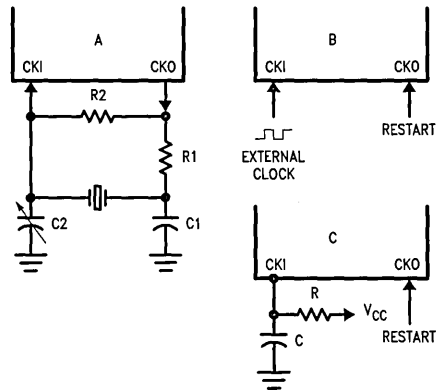


FIGURE 6. Crystal, External, and R-C Oscillator Diagrams
TL/DD/10717-6

TABLE II. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq. (MHz)
0	1	30	30-36	10
0	1	30	30-36	4
0	1	200	100-150	0.455

TABLE III. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP842CMH

Programming the COP842CMH hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the value 0E7 Hex. The device will not function properly if any other value resides in this last byte's location.

The following product codes are used by the customer to order the duplicator board.

NSID	Description	Documentation
COP8-RRGM-DIP	Duplicator Board for 20-Pin DIP	User Instruction Manual

The device will also program on a Data I/O Programmer. The following table provides the programming information on a Data I/O Programmer.

COPs Part Number	Package Type	Family Code	Pin	Software Rev	Adapter
COP842CMHD	20 DIP	16F	174	V3.2	SITE48

ERASING THE COP842CMH

Erasure of program memory is achieved by removing the COP842CMH from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the COP842CMH are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000Å (Angstroms). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range.

After programming, opaque labels should be placed over the COP842CMH's window to prevent temporary functional failure due to the generation of photo currents or high HALT mode current.

The recommended erasure procedure for the COP842CMH is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 30W-sec/cm².

The COP842CMH should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. Table IV shows the minimum COP842CMH erasure time for various light intensities.

An erasure system should be calibrated periodically. The distance from lamp to unit should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that full erasure is occurring.

TABLE IV. Minimum COP842CMH Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	40
10,000	50
5,000	100

Development Support

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP822C/842C	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1A	Personality Board	COP880 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk	424410527-001
			PC-DOS Communications Software Users Manual	420040416-001
420411060-001	Programmer's Manual		420411060-001	

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be downloaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating the development system, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice: (408) 721-5582

Modem: (408) 739-1162

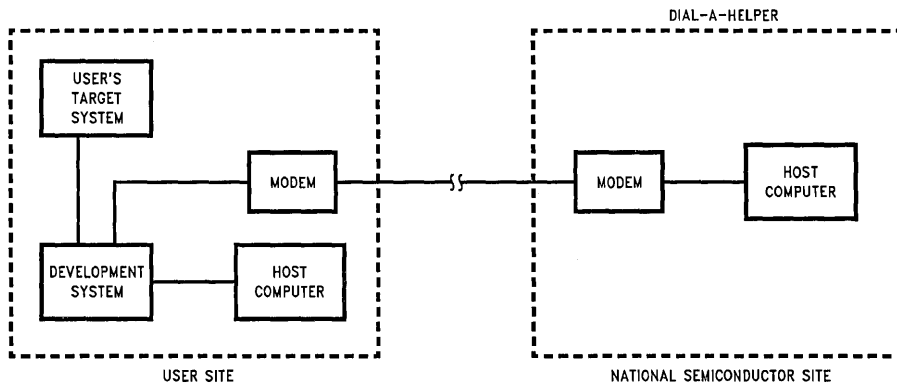
Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days



TL/DD/10717-7



COP880CMH/COP881CMH Microcontroller Emulators

General Description

The COP880CMH/COP881CMH hybrid emulators are members of the COP^{STM} microcontroller family. The device is a two chip system in a dual cavity package. Within the package is the COP880C and a UV-erasable 8k EPROM with port recreation logic code executes of the EPROM. The devices (offered in 44-pin LDCC, 40-pin DIP and 28-pin DIP packages) contain transparent windows which allow the EPROM to be erased and re-programmed. The devices are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSTM serial I/O, and a 16-bit timer/counter supporting three modes (PWM generation, External Event counter, and Input Capture). Each I/O pin has software selectable configurations. The devices operate over a voltage range of 4.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

The COP881CMH (28-pin package) can be used to emulate the COP820C/COP840C.

COP880CMH and COP881CMH are intended primarily as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

- 1 μ s instruction time
- 8191 bytes EPROM/128 bytes RAM
- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - External interrupt with selectable edge
 - Timer/capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set; most instructions are single byte
- BCD arithmetic instructions
- MICROWIRE/PLUS serial I/O
- Packages: 44-pin LDCC with 36 I/O pins
40-pin DIP with 36 I/O pins
28-pin DIP with 24 I/O pins
- Software selectable I/O options (TRI-STATE[®], push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Real time emulation and full program debug offered by National's Development Systems

Features

- Form fit and function emulation device for the COP880C/COP881C/COP840C/COP820C
- Fully static CMOS

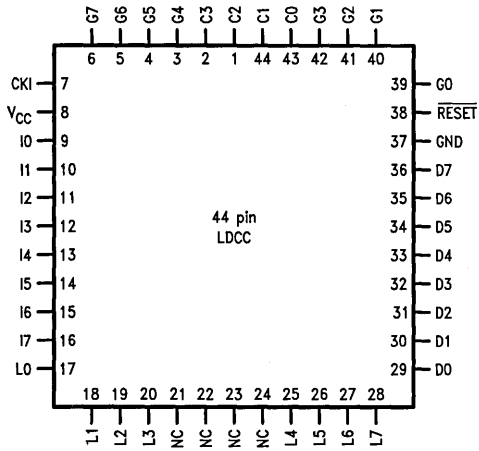
Ordering Information

Hybrid Emulator	Package Type	Part Emulated
COP880CMHD-x	40-Pin DIP	COP880C-XXX/N
COP880CMHEL-x	44-Pin LDCC	COP880C-XXX/V
COP880CMHD-x	28-Pin DIP	COP881C-XXX/N COP840C-XXX/N COP820C-XXX/N

x = 1, 2, 3. See Table III.

Connection Diagrams

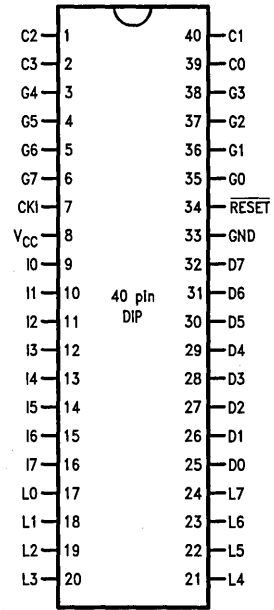
Plastic Chip Carrier



TL/DD/11022-2

Top View

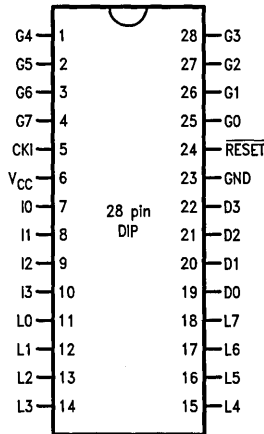
Dual-In-Line Package



TL/DD/11022-3

Top View

Dual-In-Line Package



TL/DD/11022-4

Top View

Note: X is the number which corresponds to the clock option (X = 1, for Crystal, 2 for External, 3 for RC).

FIGURE 1. COP880CMH/COP881CMH Connection Diagrams

COP880CMH/COP881CMH Pinouts

Port	Type	Alternate Function	28-Pin DIP	40-Pin DIP	44-Pin LDCC
L0	I/O		11	17	17
L1	I/O		12	18	18
L2	I/O		13	19	19
L3	I/O		14	20	20
L4	I/O		15	21	25
L5	I/O		16	22	26
L6	I/O		17	23	27
L7	I/O		18	24	28
G0	I/O	Interrupt	25	35	39
G1	I/O		26	36	40
G2	I/O		27	37	41
G3	I/O	TIO	28	38	42
G4	I/O	SO	1	3	3
G5	I/O	SK	2	4	4
G6	I	SI	3	5	5
G7	I/CKO	Halt Restart	4	6	6
I0	I		7	9	9
I1	I		8	10	10
I2	I		9	11	11
I3	I		10	12	12
I4	I			13	13
I5	I			14	14
I6	I			15	15
I7	I			16	16
D0	O		19	25	29
D1	O		20	26	30
D2	O		21	27	31
D3	O		22	28	32
D4	O			29	33
D5	O			30	34
D6	O			31	35
D7	O			32	36
C0	I/O			39	43
C1	I/O			40	44
C2	I/O			1	1
C3	I/O			2	2
V _{CC}			6	8	8
GND			23	33	37
CKI			5	7	7
RESET			24	34	38

Note: Unused pins 21–24 on 44-pin device are not connected.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Reset (V_{PP}) and G6 (ME)	-0.3V to 14V
Voltage at any other Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	50 mA

Total Current Out of GND Pin (Sink) 60 mA

Storage Temperature Range -65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics -40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current					
High Speed Mode, CKI = 10 MHz	$V_{CC} = 6.0V, t_c = 1 \mu s$			21	mA
Normal Mode, CKI = 5 MHz (Note 2)	$V_{CC} = 6.0V, t_c = 2 \mu s$			15	mA
HALT Current (Note 3)	$V_{CC} = 6.0V, CKI = 0 MHz$		500		μA
INPUT LEVELS					
Reset, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
All Others					
Source (Weak Pull-up Mode)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
Source (Push-pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs				15	mA
All Others				3	mA
Maximum Input Current without Latchup (Note 4)	Room Temp			±100	mA
RAM Retention Voltage, V_R	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

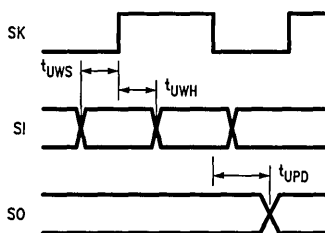
Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. HALT test conditions: L and G ports are at TRI-STATE and tied to ground, EPROM window covered.

Note 4: Pins G6 and \overline{RESET} are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

AC Electrical Characteristics — $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal/Resonator	$V_{CC} \geq 4.5\text{V}$	1		DC	μs
R/C Oscillator Mode (div-by 10)	$V_{CC} \geq 4.5\text{V}$	3		DC	μs
CKI Clock Duty Cycle (Note 5)	$f_r = \text{Max}$	40		60	%
Rise Time (Note 5)	$f_r = 10\text{ MHz ext clock}$			12	ns
Fall Time (Note 5)	$f_r = 10\text{ MHz ext clock}$			8	ns
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_c			
Interrupt Input Low Time		t_c			
Timer Input High Time		t_c			
Timer Input Low Time		t_c			
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled (not 100% tested).



TL/DD/11022-5

FIGURE 2. MICROWIRE Timing Diagram

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, an R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT I is an 8-bit Hi-Z input port.

PORT L is an 8-bit I/O port.

PORT C is a 4-bit I/O port.

Three memory locations are allocated for the L, G and C ports, one each for data register, configuration register and the input pins. Reading bits 4–7 of the C-Configuration register, data register, and input pins returns undefined data.

There are two registers associated with the L and C ports: a data register and a configuration register. Therefore, each L and C I/O bit can be individually configured under software control as shown below:

Config.	Data	Ports L and C Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

On the 28-pin part, it is recommended that all bits of Port C be configured as outputs.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated with the G port: a data register and a configuration register. Therefore, each G port bit can be individually configured under software control as shown below:

Config.	Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Pull-Up (Weak One Output)
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Since G6 and G7 are input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. The device will be placed in the HALT mode by writing to the G7 bit in the G-port data register.

Six pins of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is an 8-bit output port that is preset high when RESET goes low.

Oscillator Circuits

A. CRYSTAL OSCILLATOR

By selecting CKO as a clock output, CKI and CKO can be connected to make a crystal controlled oscillator. See Table II for value of R & C.

B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels. CKO (G7) is available as a general purpose input and/or Halt Control.

C. R/C OSCILLATOR

CKI is configured as a single pin R/C controlled Schmitt trigger oscillator. CKO (G7) is available as a general purpose input and/or HALT restart control.

Table III shows the variation in the oscillator frequencies as functions of the component (R and C) values.

OSCILLATOR MASK OPTIONS

The devices can be driven by crystal or external clock inputs between DC and 10 MHz. Table IV shows the clock option per package.

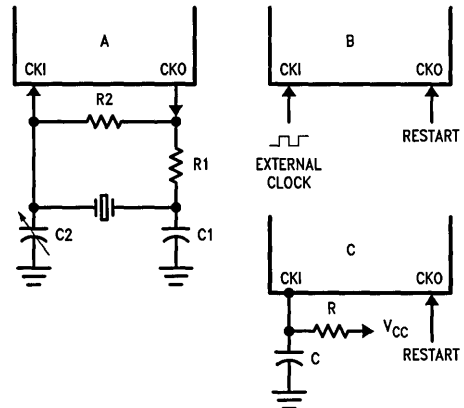


FIGURE 3. Crystal, External, and R/C Oscillator Diagrams

TL/DD/11022-6

Oscillator Circuits (Continued)

TABLE I. Crystal Oscillator Configuration, $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq. (MHz)
0	1	30	30-36	10
0	1	30	30-36	4
0	1	200	100-150	0.455

TABLE II. RC Oscillator Configuration, $T_A = 25^\circ\text{C}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

TABLE III. Clock Option Per Package

Order Part Number	Package	Clock Option
COP880CMHEL-1 COP880CMHD-1 COP881CMHD-1	44 LDCC 40 DIP 28 DIP	Crystal Oscillator \div 10
COP880CMHEL-2 COP880CMHD-2 COP881CMHD-2	44 LDCC 40 DIP 28 DIP	External Oscillator \div 10
COP880CMHEL-3 COP880CMHD-3 COP881CMHD-3	44 LDCC 40 DIP 28 DIP	R/C Oscillator \div 10

Programming the COP880CMH/COP881CMH

Programming the hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the NSC development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the proper value specified in the following table.

TABLE V

Device	Package Type	RAM Size Emulated	Contents of Last Byte (Address 01FFF)
COP880CMH	44 LDCC 40 DIP	128	7F
COP881CMH	28 DIP	128	6F
COP881CMH	28 DIP	64	EF

ORDERING INFORMATION

P/N	Description	Documentation
COP8-PRGM-PCC	Duplicator Board for 44-Pin LDCC	User Instruction Manual
COP8-PRGM-DIP	Duplicator Board for 40-Pin DIP	User Instruction Manual
COP8-PRG-28D	Duplicator Board for 28-Pin DIP	User Instruction Manual

The device will also program on a Data I/O programmer. The following table provides the programming information on a Data I/O Programmer.

COPS Part Number	Package Type	Family Code	Pin	Software Rev	Adapter
COP881CMHD	28 DIP	16F	19E	V3.3	SITE 48
COP880CMHD	40 DIP	16F	19F	V3.3	SITE 48
COP880CMHEL	44 LDCC	16F	175	V3.2	PINSITE

ERASING THE PROGRAM MEMORY

Erasement of the EPROM program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

Programming the COP880CMH/COP881CMH (Continued)

The erasure characteristics of the device are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000Å range.

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the devices is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (UV intensity \times exposure time) for erasure should be a minimum of 15 W-sec/cm².

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

Minimum Erasure Time	
Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	20
10,000	25
5,000	50

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

Development Support

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP880C/881C	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1A	Personality Board	COP880 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

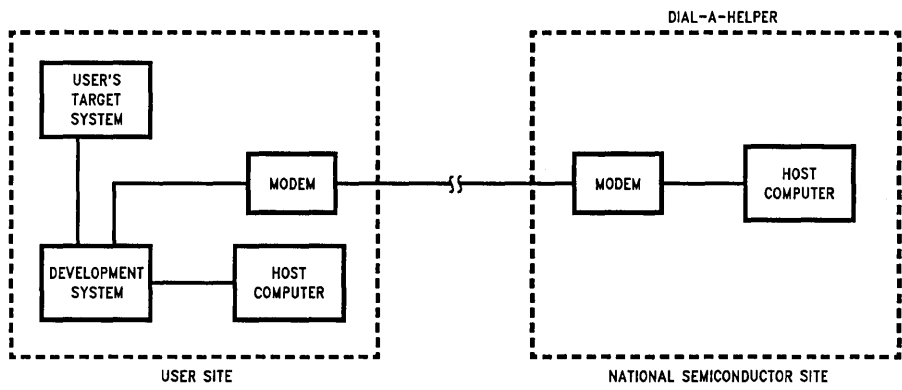
Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating the development system, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Development Support (Continued)

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 Baud
Set-up: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs., 7 Days



TL/DD/11022-7



COP8640CMH/COP8642CMH Microcontroller Emulator

General Description

The COP8640CMH/COP8642CMH hybrid emulators are members of the COPST[™] microcontroller family. The devices (offered in 28-pin DIP LCC and 20-pin DIP) contain transparent windows which allow the EPROM to be erased and reprogrammed. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. These microcontrollers are complete microcomputers containing all system timing, interrupt logic, EPROM, RAM, EEPROM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUST[™] serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP8640CMH/COP8642CMH to the specific application. The part operates over a voltage range of 4.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate.

COP8640CMH and COP8642CMH are intended primarily as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

Features

- Form fit and function emulation devices for COP8640C/COP8642C/COP8620C/COP8622C
- Fully static CMOS
- 1 μ s instruction time
- Single supply operation: 4.5V to 6.0V
- 8k bytes EPROM/64 bytes RAM/64 bytes EEPROM
- 16-Bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - Reset master clear
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE/PLUS serial I/O
- 28-pin and 20-pin DIP packages
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE[®], push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Fully supported by National's Development Systems

Ordering Information

Hybrid Emulator	Package Type	Part Emulated
COP8640CMHD-x	28-DIP	COP8640C-XXX/N COP8620C-XXX/N
COP8642CMHD-x	20-DIP	COP8642C-XXX/N COP8622C-XXX/N

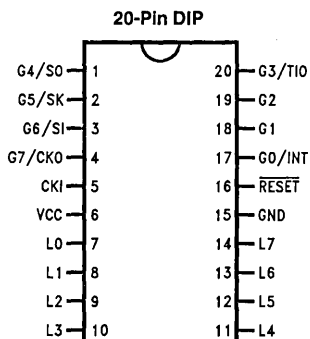
x = 1, 2, 3 corresponds to oscillator option.

Connection Diagrams

DUAL-IN-LINE PACKAGES

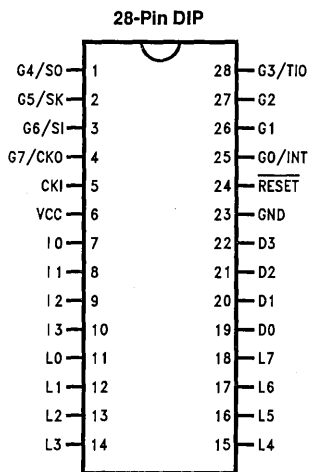
COP8640CMH/COP8642CMH

Pinouts



Top View

TL/DD/11207-1



Top View

TL/DD/11207-2

Port	Type	Alternate Function	20-Pin DIP	28-Pin DIP/LCC
L0	I/O		7	11
L1	I/O		8	12
L2	I/O		9	13
L3	I/O		10	14
L4	I/O		11	15
L5	I/O		12	16
L6	I/O		13	17
L7	I/O		14	18
G0	I/O	Interrupt	17	25
G1	I/O		18	26
G2	I/O		19	27
G3	I/O	TIO	20	28
G4	I/O	SO	1	1
G5	I/O	SK	2	2
G6	I	SI	3	3
G7	I/CKO	Halt Restart	4	4
I0	I			7
I1	I			8
I2	I			9
I3	I			10
D0	O			19
D1	O			20
D2	O			21
D3	O			22
VCC			6	6
GND			15	23
CKI			5	5
RESET			16	24

FIGURE 1. COP8640CMH/COP8642CMH Connection Diagrams

COP8640CMH/COP8642CMH**Absolute Maximum Ratings** (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	50 mA
Total Current out of GND Pin (Sink)	60 mA

Storage Temperature Range -65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current				19	mA
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$				
Supply Current during Write Operation (Note 2)				25	mA
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$				
HALT Current (Note 3)	$V_{CC} = 6V, CKI = 0 MHz$		500		μA
Input Levels					
RESET, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 4) without Latchup (Room Temp)	Room Temp			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF

COP8640CMH/COP8642CMH (Continued)**DC Electrical Characteristics** $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Condition	Min	Typ	Max	Units
EEPROM Characteristics					
EEPROM Write Cycle Time				10	ms
EEPROM Number of Write Cycles				10,000	Cycle
EEPROM Data Retention				10	Years

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports at TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Ext, Crystal/Resonator (Div-by 10)		1		DC	μs
R/C Oscillator Mode (Div-by 10)		3		DC	μs
CKI Clock Duty Cycle (Note 4)		40		60	%
Rise Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			12	ns
Fall Time (Note 4)	$f_r = 10\text{ MHz Ext Clock}$			8	ns
Inputs					
t_{SETUP}		200			ns
t_{HOLD}		60			ns
Output Propagation Delay	$C_L = 100\text{ pF}, R_L = 2.2\text{ k}\Omega$				
$t_{\text{PD1}}, t_{\text{PD0}}$				0.7	μs
SO, SK				1	μs
All Others					
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (T_{UWH})		56			ns
MICROWIRE Output Propagation Delay Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1.0			μs

Note 4: Parameter sampled but not 100% tested.

Timing Diagram

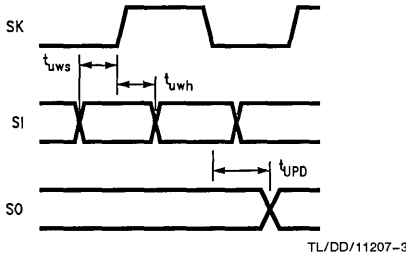


FIGURE 2. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

\overline{RESET} is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Inupt (TRI-STATE)
0	1	Input with Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below:

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input with Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when \overline{RESET} goes low.

Functional Description

OSCILLATOR CIRCUITS

Figure 3 shows the three clock oscillator configurations. Table III shows the clock options per package.

A. CRYSTAL OSCILLATOR

The COP8640CMH/COP8642CMH can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKI is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies (due to the part) as functions of the R/C component values (R/C tolerances not included).

TABLE I. Crystal Oscillator Configuration
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5.0\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
5.5	1	100	100	0.455

TABLE II. RC Oscillator Configuration
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5.0\text{V}$

R (k Ω)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)
3.3	82	2.2 to 2.7	3.7 to 4.6
5.6	100	1.1 to 1.3	7.4 to 9.0
6.8	100	0.9 to 1.1	8.8 to 10.8

Note: $3\text{k} \leq R \leq 200\text{k}$

$50\text{ pF} \leq C \leq 200\text{ pF}$

Functional Description (Continued)

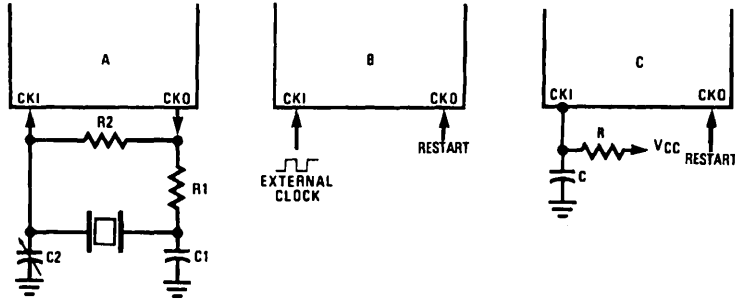


FIGURE 3. Crystal and R-C Connection Diagrams

TL/DD/11207-4

TABLE III. Clock Option per Package

Order Part Number	Package	Clock Option
COP8640CMHD-1 COP8642CMHD-1	28 DIP 20 DIP	Crystal Oscillator ÷ 10
COP8640CMHD-2 COP8642CMHD-2	28 DIP 20 DIP	External Oscillator ÷ 10
COP8640CMHD-3 COP8642CMHD-3	28 DIP 20 DIP	R/C Oscillator ÷ 10

Programming the COP8640CMH/COP8642CMH

Programming the hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the NSC development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the proper value specified in the following table:

TABLE IV

Device	Package Type	Contents of Last Byte (Address 01FFF)
COP8640CMHD	28 DIP	6F
COP8642CMHD	20 DIP	E7

ERASING THE PROGRAM MEMORY

Erasure of the EPROM program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the device are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000Å range.

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the devices is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (UV intensity × exposure time) for erasure should be a minimum of 15 W-sec/cm².

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities:

TABLE V. Minimum Erasure Time

Package Type	Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
28 DIP	15,000	20
	10,000	25
	5,000	50
20 DIP	15,000	40
	10,000	50
	5,000	100

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kbytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefineable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC® via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-8640C20D5PC	20 DIP	4.5V-5.5V	COP8642C, 8622C
MHW-8640C20DWPC	20 DIP	2.5V-6.0V	COP8642C, 8622C
MHW-8640CG28D5PC	28 DIP	4.5V-5.5V	COP8640C, 8620C
MHW-8640CG28DWPC	28 DIP	2.5V-6.0V	COP8640C, 8620C

Development Support (Continued)

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

SIMULATOR

The COP8 Designers' Toolkit is available for evaluating National Semiconductor's COP8 microcontroller family. The kit contains programmer's manuals, device datasheets, pocket reference guides, assembler and simulator which allow the user to write, test, debug and run code on an industry standard compatible PC. The simulator has a windowed user interface and can handle script files that simulate hardware inputs, interrupts and automatic command processing. The capture file feature enables the user to record to a file current cycle count and output port changes which are caused by the program under test.

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 FAX: (206) 882-1043

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 Macro Cross Assembler for IBM® PC-XT®, PC-AT® or Compatible	424410527-001

Simulator Ordering Information

Part Number	Description	Manual
COP8-TOOL-KIT	COP8 Designer's Tool Kit Assembler and Simulator	420420270-001 424420269-001

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP8640CMHD-X	X = 1 : Crystal X = 2 : External X = 3 : R/C	28 DIP	Multi-Chip Module (MCM), UV Erasable	COP8640C, 8620C
COP8640CMHEA-X	X = 1 : Crystal X = 2 : External X = 3 : R/C	28 LCC	MCM (Same Footprint as 28 SO), UV Erasable	COP8640C, 8620C
COP8642CMHD-X	X = 1 : Crystal X = 2 : External X = 3 : R/C	20 DIP	MCM, UV Erasable	COP8642C, 8622C

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP and for use with Scrambler Boards	COP8640CMHD
COP8-SCRM-DIP	Scrambler Board for 20 DIP Socket	COP8642CMHD
COP8-SCRM-SBX	Scrambler Board for 28 LCC Socket	COP8640CMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP8642CMHD, COP8640CMHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user has questions, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days

COP888CLMH

Single-Chip microCMOS Microcontroller Emulator

General Description

The COP888CLMH hybrid emulator is a member of the COPSTM microcontroller family. The device is a two chip system in a dual cavity package. Within the package is the COP888CL and a UV-erasable 8k EPROM with port recreation logic. Code executes out of EPROM. The device is offered in three packages: 44-pin LDCC, 40-pin DIP, and 28-pin DIP. All packages contain transparent windows which allows the EPROM to be erased and re-programmed.

The COP888CLMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSTM serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CLMH operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

The COP888CLMH is primarily intended as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μ s instruction cycle time
- 8192 bytes on-board EPROM
- 128 bytes on-board RAM

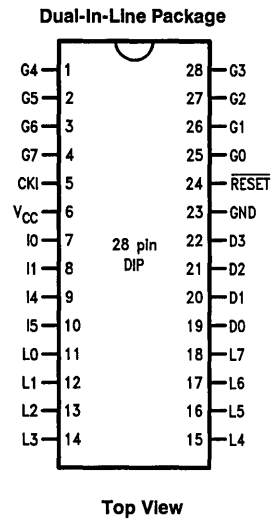
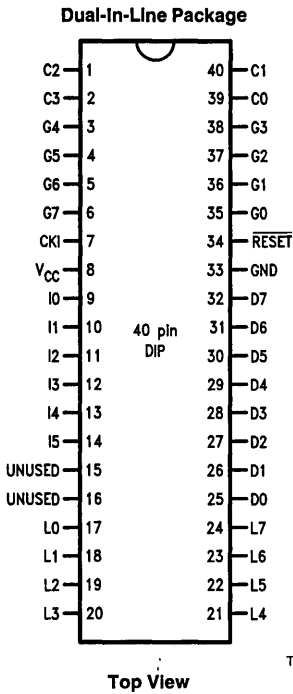
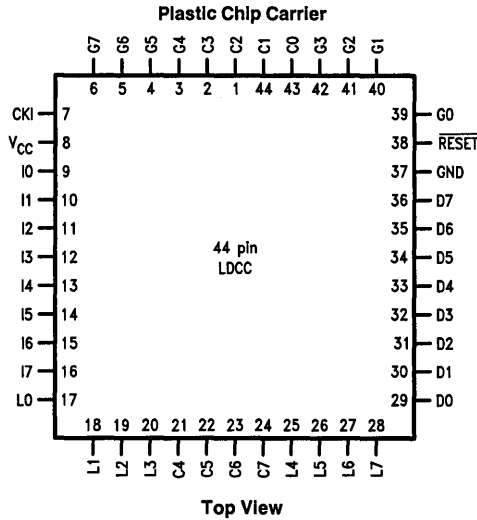
- Single supply operation: 4.5V–5.5V
- MICROWIRE/PLUSTM serial I/O
- WATCHDOG™ and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Two 16-bit timers, each with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- Ten multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Two Timers each with 2 interrupts
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - Default VIS
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set with true bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 LDCC with 39 I/O pins
40 DIP with 33 I/O pins
28 DIP with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CL
- Real time emulation and full program debug offered by National's Development Systems

Ordering Information

Hybrid Emulator	Package Type	Part Emulated with Crystal Oscillator Option
COP888CLMHD-x	40-Pin DIP	COP888CL-XXX/N
COP888CLMHXL-x	44-Pin LDCC	COP888CL-XXX/V
COP884CLMHD-x	28-Pin DIP	COP884CL-XXX/N

x indicates crystal oscillator option; for applications requiring R/C oscillator option check with your local National Sales Representative.

Connection Diagrams



Note: The pins labeled unused must be connected to GND.

FIGURE 1. COP888CLMH Connection Diagrams

COP888CLMH Pinouts

Port	Type	Alternate Fun	Alternate Fun	28-Pin DIP	40-Pin DIP	44-Pin LDCC
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU		12	18	18
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT RESTART		4	6	6
I0	I			7	9	9
I1	I			8	10	10
I2	I				11	11
I3	I				12	12
I4	I			9	13	13
I5	I			10	14	14
I6	I					15
I7	I					16
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
D4	O				29	33
D5	O				36	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
Unused*					16	
Unused*					15	
Vcc				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

*On the 40-pin package pins 15 and 16 must be connected to GND.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	$-65^{\circ}C$ to $+140^{\circ}C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1 V_{CC}$	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			25	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0$ MHz		250		μA
IDLE Current CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			15	mA
Input Levels \overline{RESET} Logic High Logic Low CKI (External and Crystal Osc. Modes) Logic High Logic Low All Other Inputs Logic High Logic Low		$0.8 V_{CC}$ $0.7 V_{CC}$ $0.7 V_{CC}$		$0.2 V_{CC}$ $0.2 V_{CC}$ $0.2 V_{CC}$	V V V V V V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 5.5V$	40		250	μA
G and L Port Input Hysteresis			$0.05 V_{CC}$		V
Output Current Levels D Outputs Source Sink All Others Source (Weak Pull-Up Mode) Source (Push-Pull Mode) Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.3V$ $V_{CC} = 4.5V, V_{OL} = 1V$ $V_{CC} = 4.5V, V_{OH} = 2.7V$ $V_{CC} = 4.5V, V_{OH} = 3.3V$ $V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4 10 10 0.4 1.6		100	mA mA μA mA mA
TRI-STATE Leakage	$V_{CC} = 4.5V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

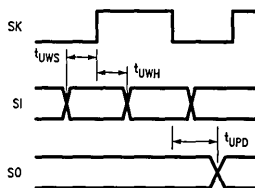
Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All Others				15 3	mA mA
Maximum Input Current without Latchup	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator		1 3		DC DC	μs μs
CKI Clock Duty Cycle (Note 5) Rise Time (Note 5) Fall Time (Note 5)	$f_r = \text{Max}$ $f_r = 10 \text{ MHz Ext Clock}$ $f_r = 10 \text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}		200 60			ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100 \text{ pF}$			0.7 1	μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Note 5: Parameter sampled (not 100% tested).



TL/DD/10467-3

FIGURE 2. MICROWIRE/PLUS Timing

Pin Descriptions

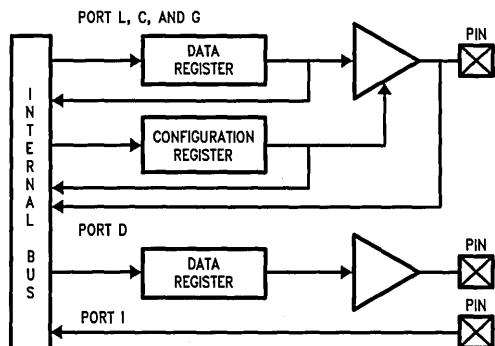
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CLMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. Figure 3 shows the I/O port configurations for the COP888CLMH. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/10467-4

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B.

The Port L has the following alternate features:

- L0 MIWU
- L1 MIWU
- L2 MIWU
- L3 MIWU
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT (WATCHDOG and/or Clock Monitor dedicated output)
- G7 CKO (Oscillator dedicated output or general purpose input)

Port I is an eight-bit input port. The 28- and 40-pin devices do not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Pin Descriptions (Continued)

Port I1-13 are used for Comparator 1. Port I4-16 are used for Comparator 2.

The Port I has the following alternate features.

- 11 COMP1-IN (Comparator 1 Negative Input)
- 12 COMP1+IN (Comparator 1 Positive Input)
- 13 COMP1OUT (Comparator 1 Output)
- 14 COMP2-IN (Comparator 2 Negative Input)
- 15 COMP2+IN (Comparator 2 Positive Input)
- 16 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 4 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (Special Order from Factory)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.

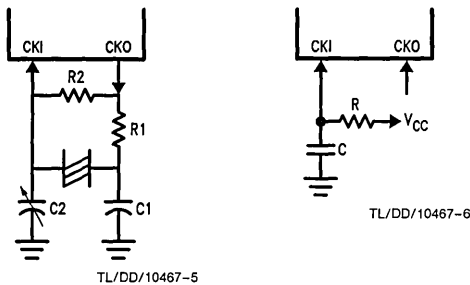


FIGURE 4. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration,
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30-36	10
0	1	30	30-36	4
0	1	200	100-150	0.455

TABLE II. RC Oscillator Configuration,
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP888CLMH

Programming the COP888CLMH hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the value specified in the following table.

Package	HALT MODE	Contents of Last Byte (Address 01FFF)
28	Enabled	6F
28	Disabled	EF
40/44	Enabled	7F
40/44	Disabled	FF

The following product codes are used by the customer order to order the duplicator board.

NSID	Description	Documentation
COP8-PRMG-PCC	Duplicator Board for 44-Pin LDCC	User Instruction Manual
COP8-PRGM-DIP	40-Pin DIP	User Instruction Manual
COP8-PRGM-28D	28-Pin DIP	User Instruction Manual

The device will also program on a Data I/O Programmer. The following table provides the programming information on a data I/O programmer.

COPs Part Number	Package Type	Family Code	Pin	Software Rev	Adapter
COP884CLMHD	28 DIP	16F	19E	V3.3	SITE 48
COP888CLMHD	40 DIP	16F	19F	V3.3	SITE 48
COP888CLMHEL	44 LDCC	16F	175	V3.2	PINSITE

ERASING THE PROGRAM MEMORY

Erasure of the program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the device are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (\AA). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000 \AA to 4000 \AA range.

Programming the COP888CLMH (Continued)

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the device is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (UV intensity \times exposure time) for erasure should be a minimum of 15 W-sec/cm².

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

TABLE III. Minimum COP888CLMH Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	20
10,000	25
5,000	50

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the dis-

tance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

Development Support

DEVELOPMENT SYSTEM

The NSC Development System is a low cost development system and emulator for all microcontroller products. These include COPs microcontrollers and the HPC family of products. The development system consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the development system is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other development systems in a multi-development system environment.

Development systems can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be downloaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating the development system he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice: (408) 721-5582

Modem: (408) 739-1162

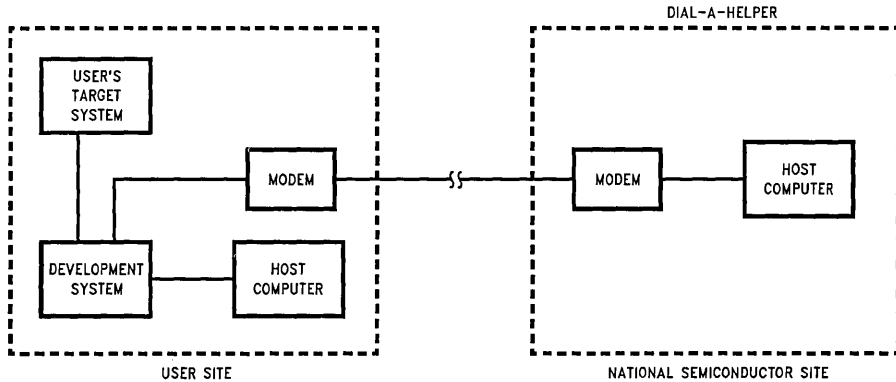
Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days



TL/DD/10467-9



COP888CFMH

Single-Chip microCMOS Microcontroller Emulator

General Description

The COP888CFMH hybrid emulator is a member of the COPS™ microcontroller family. The device is a two chip system in a dual cavity package. Within the package is the COP888CF and a UV-erasable 8k EPROM with port recreation logic code executes of the EPROM. This device is offered in three packages: 44-pin LDCC, 40-pin DIP and 28-pin DIP. All packages contain transparent windows which allows the EPROM to be erased and re-programmed.

The COP888CFMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), an 8-channel, 8-bit A/D converter with both differential and single ended modes, and two power savings modes (HALT and IDLE), both with a multi-sourced wake-up/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CFMH operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

The COP888CFMH is primarily intended as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

Features

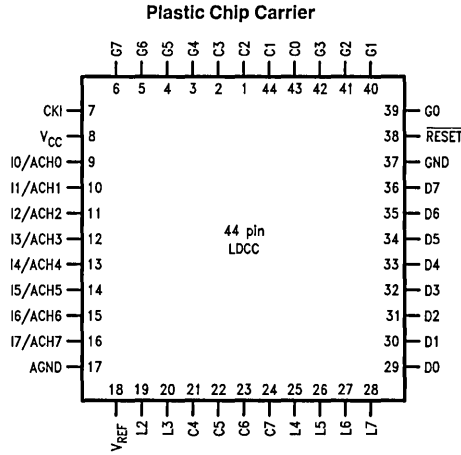
- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μ s instruction cycle time
- 8192 bytes on-board EPROM
- 128 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- 8-Channel A/D converter with prescaler and both differential and single ended modes
- MICROWIRE/PLUS serial I/O
- WATCHDOG™ and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer TO
 - Two Timers each with 2 interrupts
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - Default VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set with True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 LDCC with 37 I/O pins
40 DIP with 33 I/O pins
28 DIP with 21 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CF
- Real time emulation and full program debug offered by National's Development Systems

Ordering Information

Hybrid Emulator	Package Type	Part Emulated with Crystal Oscillator Option
COP888CFMHD-x	40-Pin DIP	COP888CF-XXX/N
COP888CFMHEL-x	44-Pin LDCC	COP888CF-XXX/V
COP884CFMHD-x	28-Pin DIP	COP884CF-XXX/N

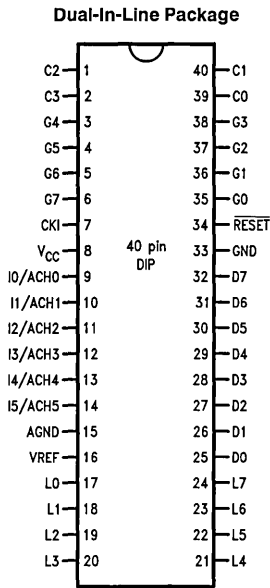
x indicates crystal option; for applications requiring R/C option check with local sales representative.

Connection Diagrams



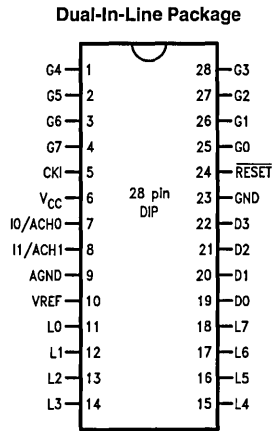
TL/DD/10464-2

Top View



TL/DD/10464-1

Top View



TL/DD/10464-10

Top View

FIGURE 1. COP888CFMH Connection Diagrams

Connection Diagrams (Continued)

COP888CFMH Pinouts

Port	Type	Alternate Fun	Alternate Fun	28-Pin DIP	40-Pin DIP	44-Pin LDCC
L0	I/O	MIWU		11	17	
L1	I/O	MIWU		12	18	
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT RESTART		4	6	6
I0	I	ACH0		7	9	9
I1	I	ACH1		8	10	10
I2	I	ACH2			11	11
I3	I	ACH3			12	12
I4	I	ACH4			13	13
I5	I	ACH5			14	14
I6	I	ACH6				15
I7	I	ACH7				16
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V _{REF}	+V _{REF}			10	16	18
AGND/GND	AGND			9	15	17
V _{CC}				6	8	8
GND				23	33	37
CK1				5	7	7
RESET				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			25	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0$ MHz		250		μA
IDLE Current CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			15	mA
Input Levels					
RESET					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 5.5V$	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.3V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1V$	10			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V, V_{OH} = 2.7V$	10		100	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.3V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage	$V_{CC} = 4.5V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. If the A/D is not being used and minimum standby current is desired, V_{REF} should be tied to AGND (effectively shorting the reference resistor). The clock monitor is disabled.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All Others				15 3	mA mA
Maximum Input Current without Latchup	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications $V_{CC} = 5\text{V} \pm 10\%$; $(V_{SS} - 0.050\text{V}) \leq \text{Any Input} \leq (V_{CC} + 0.050\text{V})$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			± 1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the best straight line			$\pm \frac{1}{2}$	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			$\pm \frac{1}{2}$	LSB
Input Reference Resistance		1.6		4.8	k Ω
Common Mode Input Range (Note 7)		AGND		V_{REF}	V
DC Common Mode Error				$\pm \frac{1}{4}$	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
A/D Clock Frequency (Note 5)		0.1		1.67	MHz
Conversion Time (Note 4)			12		A/D clock Cycles

Note 4: Conversion Time includes sample and hold time.

Note 5: See Prescaler description.

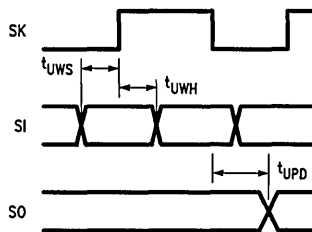
Note 6: Pins G6 and RESET are designed with a high voltage input network for factory testing. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

Note 7: For $V_{IN(-)} \geq V_{IN(+)}$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator		1 3		DC DC	μs μs
CKI Clock Duty Cycle (Note 7) Rise Time (Note 7) Fall Time (Note 7)	$f_r = \text{Max}$ $f_r = 10 \text{ MHz Ext Clock}$ $f_r = 10 \text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}		200 60			ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100 \text{ pF}$			0.7 1	μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 7: Parameter sampled (not 100% tested).



TL/DD/10464-3

FIGURE 2. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

V_{REF} and AGND are the reference pins for the onboard A/D converter.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CFMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. Figure 3 shows the I/O port configurations. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

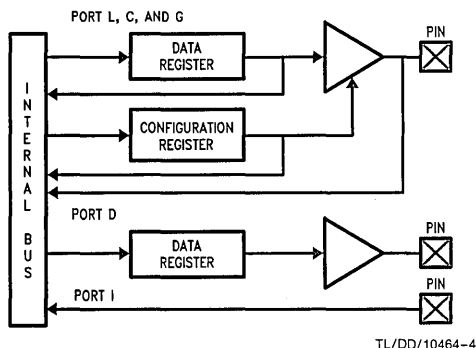


FIGURE 3. I/O Port Configurations

Port L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up. L4 and L5 are used for the timer input functions T2A and T2B.

The Port L has the following alternate features:

- L0 MIWU (28- and 40-pin only)
- L1 MIWU (28- and 40-pin only)
- L2 MIWU
- L3 MIWU
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT (WATCHDOG and/or Clock Monitor dedicated output)
- G7 CKO (Oscillator dedicated output or general purpose input)

Port I is an eight-bit Hi-Z input port and also provides the analog inputs to the A/D Converter.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Port C is an 8-bit I/O port.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Oscillator Circuits (Continued)

Figure 4 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

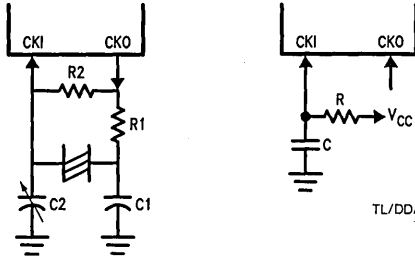
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (Special Order from Factory)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10464-6

TL/DD/10464-5

FIGURE 4. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration,
 $T_A = 25^\circ\text{C}, V_{CC} = 5\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30-36	10
0	1	30	30-36	4
0	1	200	100-150	0.455

TABLE II. RC Oscillator Configuration,
 $T_A = 25^\circ\text{C}, V_{CC} = 5\text{V}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP888CFMH

Programming the COP888CFMH hybrid emulators is accomplished with the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the value specified in the following table.

TABLE III

Package	HALT Mode	Contents of Last Byte (Address 01FFF)
28	Enabled	6F
28	Disabled	EF
40/44	Enabled	7F
40/44	Disabled	FF

The following product codes are used by the customer order to order the duplicator board.

NSID	Description	Documentation
COP8-PRGM-PCC	Duplicator Board for 44-Pin LDCC	User Instruction Manual
COP8-PRGM-DIP	40-Pin DIP	User Instruction Manual
COP8-PRGM-28D	28-Pin DIP	User Instruction Manual

The device will also program on a Data I/O Programmer.

The following table provides the programming information on a Data I/O Programmer.

COPs Part Number	Package Type	Family Code	Pin	Software Rev	Adapter
COP884CFMHD	28 DIP	16F	19E	V3.3	SITE 48
COP888CFMHD	40 DIP	16F	19F	V3.3	SITE 48
COP888CFMHEL	44 LDCC	16F	175	V3.2	PINSITE

ERASING THE PROGRAM MEMORY

Erasure of the program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the device are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000Å range.

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the device is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (UV intensity X exposure time) for erasure should be a minimum of 15 W-sec/cm².

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

TABLE IV. Minimum COP888CFMH Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	20
10,000	25
5,000	50

Development Support

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

DEVELOPMENT SYSTEM

The NSC Development System is a low cost development system and emulator for all microcontroller products. These include COPs microcontrollers and the HPC family of products. The development system consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the development system is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other development systems in a multi-development system environment.

Development system can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

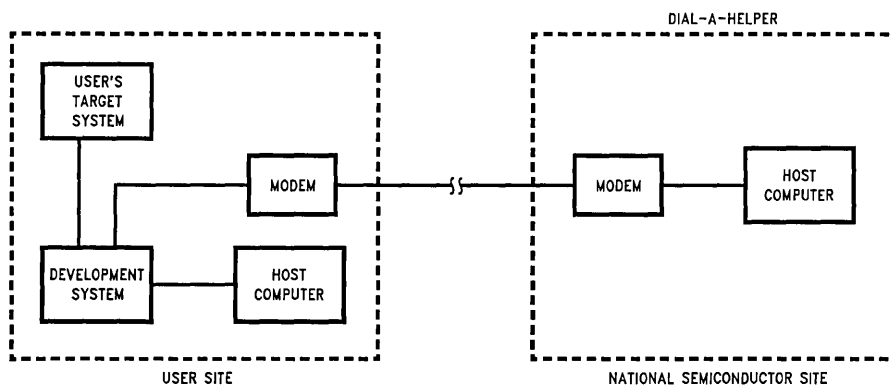
Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating the development system, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Development Support (Continued)

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 Baud
Set-up: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs., 7 Days



TL/DD/10464-09



PRELIMINARY

COP888CGMH/COP884CGMH and COP888EGMH microCMOS Microcontroller Emulator

General Description

The COP888CGMH and COP888EGMH hybrid emulators are members of the COP8™ microcontroller family. The device is a two chip system in a dual cavity package. Within the package is the COP888CG or COP888EG and a UV-erasable 8k EPROM with port recreation logic. Code executes out of EPROM. The device is offered in the following packages: COP888CG and COP888EG are in 44-pin LDCC, 40-pin DIP, and COP884CGMH in 28-pin DIP. All packages contain transparent windows which allow the EPROM to be erased and re-programmed.

The COP888CGMH/COP888EGMH are fully static, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSTM serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The device operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μ s per instruction rate.

These devices are primarily intended as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μ s instruction cycle time
- 8192 bytes on-board EPROM
- 192 bytes on-board RAM for COP888CGMH; 256 bytes for COP888EGMH
- Single supply operation: 4.5V–5.5V
- Full duplex UART
- Two analog comparators
- MICROWIRE/PLUS serial I/O
- WATCHDOG and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Three 16-bit timers, each with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- Fourteen multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Two Timers each with 2 interrupts
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - UART (2)
 - Default VIS
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set with true bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 LDCC with 39 I/O pins
40 DIP with 35 I/O pins
28 DIP with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CG and COP888EG
- Real time emulation and full program debug offered by National's Development Systems

Ordering Information

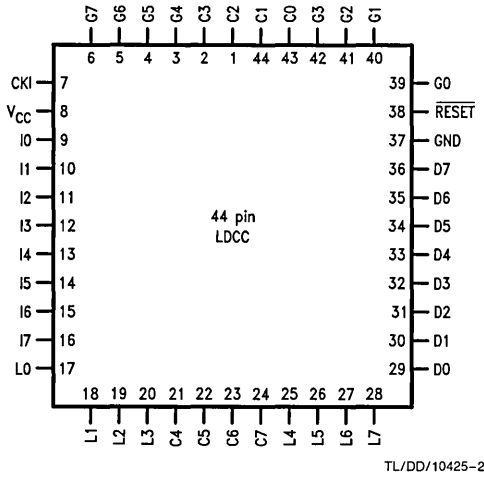
Hybrid Emulator	Package Type	Part Emulated with Crystal Oscillator Option
COP888CGMHD-X	40-Pin DIP	COP888CG-XXX/N
COP888CGMHXL-X	44-Pin LDCC	COP888CG-XXX/V
COP884CGMHD-X	28-Pin DIP	COP884CG-XXX/N

Hybrid Emulator	Package Type	Part Emulated with Crystal Oscillator Option
COP888EGMHD-X	40-Pin DIP	COP888EG-XXX/N
COP888CGMHXL-X	44-Pin LDCC	COP888EG-XXX/V

X Indicates Crystal Option: for applications requiring R/C oscillator option check with your local sales representative.

Connection Diagrams

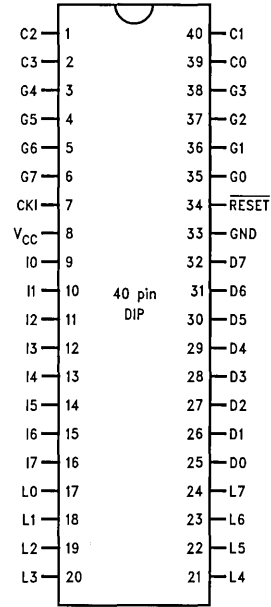
Plastic Chip Carrier



Top View

Order Number
COP888CGMHXL-X or COP888EGMHXL-X
 See NS Package EL44B

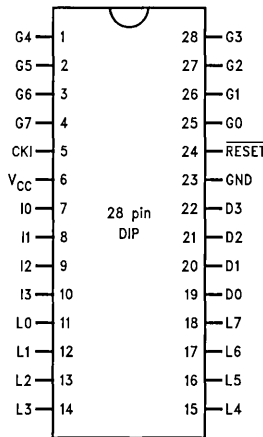
Dual-In-Line Package



Top View

Order Number
COP888CGMHD-X or COP888EGMHD-X
 See NS Package D40J

Dual-In-Line Package



Top View

Order Number **COP884CGMHD**
 See NS Package D28G

FIGURE 1. COP888CGMHD/COP888EGMH Connection Diagrams

Connection Diagrams (Continued)

COP888CGMH/COP888EGMH Pinouts

Port	Type	Alternate Fun	Alternate Fun	28-Pin DIP	40-Pin DIP	44-Pin LDCC
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU	T3A	17	23	27
L7	I/O	MIWU	T3B	18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT RESTART		4	6	6
I0	I			7	9	9
I1	I	COMP1IN-		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I	COMP2IN-			13	13
I5	I	COMP2IN+			14	14
I6	I	COMP2OUT			15	15
I7	I				16	16
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V _{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
<u>RESET</u>				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			25	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0$ MHz		250		μA
IDLE Current CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			15	mA
Input Levels RESET Logic High Logic Low CKI (External and Crystal Osc. Modes) Logic High Logic Low All Other Inputs Logic High Logic Low		0.8 V_{CC} 0.7 V_{CC} 0.7 V_{CC}		0.2 V_{CC} 0.2 V_{CC} 0.2 V_{CC}	V V V V V V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 5.5V$	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels D Outputs Source Sink All Others Source (Weak Pull-Up Mode) Source (Push-Pull Mode) Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.3V$ $V_{CC} = 4.5V, V_{OL} = 1V$ $V_{CC} = 4.5V, V_{OH} = 2.7V$ $V_{CC} = 4.5V, V_{OH} = 3.3V$ $V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4 10 10 0.4 1.6		100	mA mA μA mA mA
TRI-STATE Leakage	$V_{CC} = 4.5V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The comparators and Clock Monitor are disabled.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

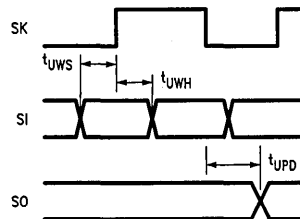
Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All Others				15 3	mA mA
Maximum Input Current without Latchup (Note 4)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator		1 3		DC DC	μs μs
CKI Clock Duty Cycle (Note 5) Rise Time (Note 5) Fall Time (Note 5)	$f_r = \text{Max}$ $f_r = 10 \text{ MHz Ext Clock}$ $f_r = 10 \text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}		200 60			ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100 \text{ pF}$			0.7 1	μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Except pin G7: -60 mA to $+100 \text{ mA}$ (sampled but not 100% tested).

Note 5: Parameter sampled (not 100% tested).



TL/DD/10425-5

FIGURE 2. MICROWIRE/PLUS Timing

Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25^\circ C$

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		± 10	± 25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs

Pin Descriptions

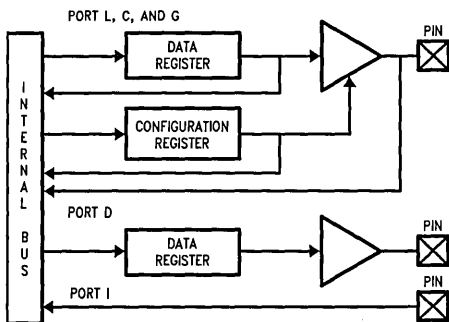
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CGMH/COP888EGMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. Figure 3 shows the I/O port configurations. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/10425-6

FIGURE 3. I/O Port Configurations

Port L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

- L0 MIWU
- L1 MIWU or CKX
- L2 MIWU or TDX
- L3 MIWU or RDX
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU or T3A
- L7 MIWU or T3B

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUR WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Pin Descriptions (Continued)

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOU (WATCHDOG and/or Clock Monitor dedicated output)
- G7 CKO (Oscillator dedicated output or general purpose input)

Port I is an 8-bit port. Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

- I1 COMP1–IN (Comparator 1 Negative Input)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I3 COMP1OUT (Comparator 1 Output)
- I4 COMP2–IN (Comparator 2 Negative Input)
- I5 COMP2+IN (Comparator 2 Positive Input)
- I6 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Port C is an 8-bit I/O port.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 4 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (SPECIAL ORDER FROM FACTORY)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.

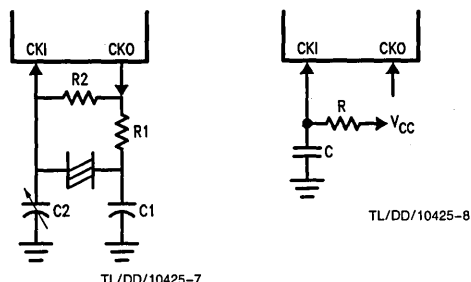


FIGURE 4. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration,
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. RC Oscillator Configuration,
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP888CGMH/ COP888EGMH

Programming the COP888CGMH/COP888EGMH hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the value specified in the following table.

Programming the COP888CGMH/ COP888EGMH (Continued)

TABLE III

Package	HALT Mode	Contents of Last Byte (Address 01FFF)
28	Enabled	6F
28	Disabled	EF
40/44	Enabled	7F
40/44	Disabled	FF

ERASING THE PROGRAM MEMORY

Erasure of the program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the device are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000Å range.

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the device is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The intergrated dose (UV intensity × exposure time) for erasure should be a minimum of 15 W-sec/cm².

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

TABLE IV. Minimum COP888CGMH/COP888EGMH Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	20
10,000	25
5,000	50

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features

high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32k Bytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read & write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER'S performance analyzer offers a resolution of better than 6 μs. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2k Baud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	Metalink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Development Support

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884CG28D5PC	28 DIP	4.5-5.5V	COP884CG
MHW-884CG28DWPC	28 DIP	2.5-6.0V	COP884CG
MHW-888CG40D5PC/ MHW-888EG40D5PC	40 DIP	4.5-5.5V	COP888CG/ COP888EG
MHW-888CG40DWPC/ MHW-888EG40DWPC	40 DIP	2.5-6.0V	COP888CG/ COP888EG
MHW-888CG44D5PC/ MHW-888EG44D5PC	44 PLCC	4.5-5.5V	COP888CG/ COP888EG
MHW-888CG44DWPC/ MHW-888EG44DWPC	44 PLCC	2.5-6.0V	COP888CG/ COP888EG

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 Macro Cross Assembler for IBM PC/XT, AT or Compatible	424410527-001

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888CGMHXL-X/ COP888EGMHXL-X	X=1: Crystal X=3: R/C	44 LDCC	Multi-Chip Module (MCM), UV Erasable	COP888CG/ COP888EG
COP888CGMHD-X/ COP888EGMHD-X	X=1: Crystal X=3: R/C	40 DIP	MCM, UV Erasable	COP888CG/ COP888EG
COP884CGMHD-X	X=1: Crystal X=3: R/C	28 DIP	MCM, UV Erasable	COP884CG
COP884CGMHEA-X	X=1: Crystal X=3: R/C	28 LCC	MCM (Same Footprint as 28SO), UV Erasable	COP884CG

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for use with Scrambler Boards	COP884CGMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP Socket	COP888CGMHD/COP888EGMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888CGMHXL/COP888EGMHXL
COP8-SCRM-SBX	MCM Scrambler Board for 28 LCC Socket	COP884CGMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP884CGMHD, COP888CGMHD/ COP888EGMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC Scrambler Board	COP888CGMHXL/COP888EGMHD, COP884CGMHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

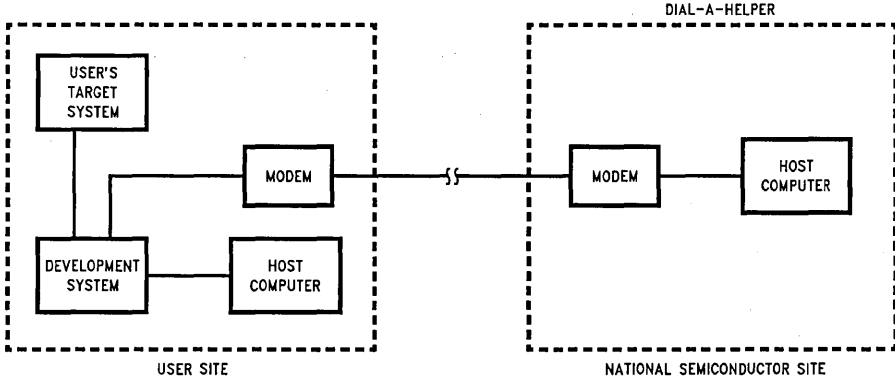
ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
 Dial-A-Helper Users Manual
 Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating the development system, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice: (408) 721-5582
 Modem: (408) 739-1162
 Baud: 300 or 1200 Baud
 Set-up: Length: 8-Bit
 Parity: None
 Stop Bit: 1
 Operation: 24 Hrs., 7 Days



TL/DD/10425-11



COP820CJM/COP822CJM Single-Chip microCMOS Microcontroller

General Description

The COP820CJM and COP822CJM hybrid emulators are members of the COPSM microcontroller family. Each device is a two chip system in a dual cavity package. Within the package is the COP820CJ and a UV-erasable 8k EPROM with port recreation logic. The code executes out of the EPROM. The devices (offered in 28-pin DIP and 20-pin DIP packages) contain transparent windows which allow the EPROM to be erased and reprogrammed. The devices are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRETM serial I/O, a 16-bit timer/counter with capture register, a multi-sourced interrupt, Comparator, WATCHDOGTM Timer, Modulator/Timer, and Multi-Input Wakeup. Each I/O pin has software selectable options to adapt the device to the specific application. The device operates over a voltage range of 4.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 μ s per instruction rate.

COP820CJM and COP822CJM are intended primarily as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only. These devices do not emulate the Brown Out feature.

Features

- Form, fit and function emulation device for the COP820CJ/COP822CJ
- Fully static CMOS
- 1 μ s instruction time
- Single supply operation: 4.5V to 6.0V
- 8191 x 8 on-chip ROM
- 64 bytes on-chip RAM
- WATCHDOG Timer
- Comparator
- Modulator/Timer (High speed PWM Timer for IR Transmission)
- Multi-Input Wakeup (on the 8-bit Port L)
- 4 high current I/O pins with 15 mA sink capability
- MICROWIRE/PLUSTSM serial I/O
- 16-bit read/write timer operates in a variety of modes
 - Timer with 16-bit auto reload register
 - 16-bit external event counter
 - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
 - External interrupt with selectable edge
 - Timer interrupt or capture interrupt
 - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- 28- and 20-pin DIP
- Software selectable I/O options (TRI-STATE[®], push-pull, weak pull-up)
- Schmitt trigger inputs on Port G and Port L

Ordering Information

Hybrid Emulator	Package Type	Part Emulated
COP820CJMHD-X	28-Pin DIP	COP820CJ-XXX/N
COP822CJMHD-X	20-Pin DIP	COP822CJ-XXX/N

Note: X corresponds to clock options.

- X = 1,2 or 3.
- 1 = crystal \div 10,
- 2 = External \div 10,
- 3 = R/C \div 10

COP820CJMH/COP822CJMH**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7.0V
Reset (V_{PP}) and G6 (ME)	-0.3V to 14V
Voltage at any Pin	-0.3V to V_{CC} + 0.3V
Total Current into V_{CC} pin (Source)	80 mA
Total Current out of GND pin (sink)	80 mA
Storage Temperature Range	-65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur.

DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage	Brown Out Disabled	4.5		6.0	V
Power Supply Ripple 1 (Note 1)	Peak to Peak			$0.1 V_{CC}$	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1 \mu s$			18.0	mA
CKI = 4 MHz	$V_{CC} = 6V, t_c = 2.5 \mu s$			16.0	mA
HALT Current	$V_{CC} = 6V, CKI = 0 \text{ MHz}$		500		μA
INPUT LEVELS (V_{IH}, V_{IL})					
Reset, CKI:					
Logic High		$0.8 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 6.0V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 6.0V$	40		250	μA
L- and G-Port Hysteresis			$0.05 V_{CC}$	$0.35 V_{CC}$	V
Output Current Levels					
D Outputs:					
Source	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1.0V$	10			mA
L4-L7 Output Sink	$V_{CC} = 4.5V, V_{OL} = 2.5V$	15			mA
All Others					
Source (Weak Pull-up Mode)	$V_{CC} = 4.5V, V_{OH} = 3.2V$	10		110	μA
Source (Push-pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.8V$	0.4			mA
Sink (Push-pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		-2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs				15	mA
L4-L7 (Sink)				20	mA
All Others				3	mA

DC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Maximum Input Current without Latchup (Note 4)	Room Temperature			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 10 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. HALT test conditions: L and G0..G5 ports configured as outputs and set high. The D port set to zero. All inputs tied to V_{CC} . The comparator and the Brown Out circuits are disabled.

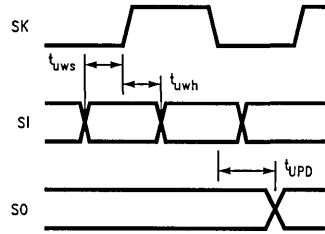
Note 4: Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750 Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal/Resonator	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	1		DC	μs
R/C Oscillator	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	3		DC	μs
V_{CC} Rise Time when Using Brown Out Frequency at Brown Out Reset CKI Frequency For Modular Output		50		4 4	μs MHz MHz
CKI Clock Duty Cycle (Note 5)	fr = Max	40		60	%
Rise Time (Note 5)	fr = 10 MHz ext. Clock			12	ns
Fall Time (Note 5)	fr = 10 MHz ext. Clock			8	ns
Inputs					
t_{Setup}	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	200			ns
t_{Hold}	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$	60			ns
Output Propagation Delay	$R_L = 2.2\text{k}, C_L = 100\text{pF}$				
$t_{\text{PD1}}, t_{\text{PD0}}$				0.7	μs
SO, SK	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$			1	μs
All Others	$4.5\text{V} \leq V_{CC} \leq 6.0\text{V}$				
Input Pulse Width					
Interrupt Input High Time		1			tc
Interrupt Input Low Time		1			tc
Timer Input High Time		1			tc
Timer Input Low Time		1			tc
MICROWIRE Setup Time ($t_{\mu\text{WS}}$)		20			ns
MICROWIRE Hold Time ($t_{\mu\text{WH}}$)		56			ns
MICROWIRE Output Propagation Delay ($t_{\mu\text{PD}}$)				220	ns
Reset Pulse Width		1.0			μs

Note 5: Parameter sampled but not 100% tested.

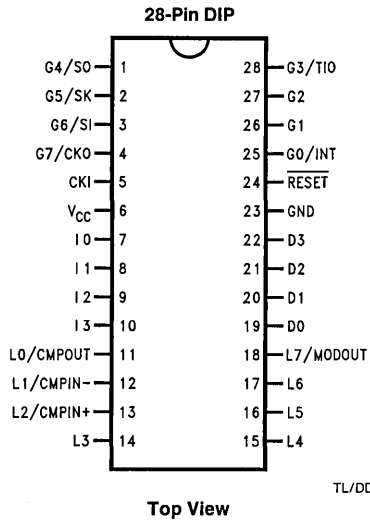
AC Electrical Characteristics (Continued)



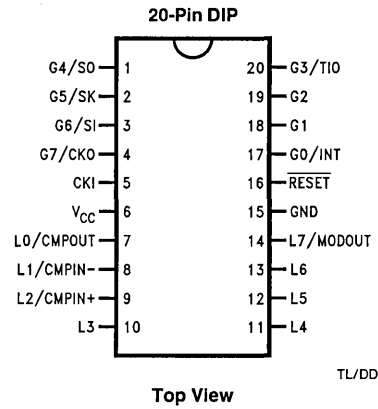
TL/DD/11373-1

FIGURE 1. MICROWIRE/PLUS Timing

COP820CJMHD Connection Diagrams



TL/DD/11373-2



TL/DD/11373-3

FIGURE 2. COP820CJMHD/COP822CJMHD Pinout

Pin Assignment

Port Pin	Typ	ALT Funct.	20 Pin	28 Pin
L0	I/O	MIWU/CMPOUT	7	11
L1	I/O	MIWU/CMPIN-	8	12
L2	I/O	MIWU/CMPIN+	9	13
L3	I/O	MIWU	10	14
L4	I/O	MIWU	11	15
L5	I/O	MIWU	12	16
L6	I/O	MIWU	13	17
L7	I/O	MIWU/MODOUT	14	18
G0	I/O	INTR	17	25
G1	I/O		18	26
G2	I/O		19	27
G3	I/O	TIO	20	28
G4	I/O	SO	1	1
G5	I/O	SK	2	2
G6	I	SI	3	3
G7	I	CKO	4	4
I0	I			7
I1	I			8
I2	I			9
I3	I			10
D0	O			19
D1	O			20
D2	O			21
D3	O			22
V _{CC}			6	6
GND			15	23
CKI			5	5
RESET			16	24

Pin Description

V_{CC} and **GND** are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT I is a 4-bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with the L port: a data register and a configuration register. Therefore, each L

I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input with Weak Pull-up
1	0	Push-pull Zero Output
1	1	Push-pull One Output

Three data memory address locations are allocated for this port, one each for data register [00D0], configuration register [00D1] and the input pins [00D2].

Port L has the following alternate features:

L0 MIWU or CMPOUT

L1 MIWU or CMPIN-

L2 MIWU or CMPIN+

L3 MIWU

L4 MIWU (high sink current capability)

L5 MIWU (high sink current capability)

L6 MIWU (high sink current capability)

L7 MIWU or MODOUT (high sink current capability)

The selection of alternate Port L function is done through registers WKEN [00C9] to enable MIWU and CNTRL2 [00CC] to enable comparator and modulator.

All eight L-pins have Schmitt Triggers on their inputs.

PORT G is an 8-bit port with 6 I/O pins (G0-G5) and 2 input pins (G6, G7).

All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated with the G port: a data register and a configuration register. Therefore each G port bit can be individually configured under software control as shown below:

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input with Weak Pull-up
1	0	Push-pull Zero Output
1	1	Push-pull One Output

Three data memory address locations are allocated for this port, one for data register [00D3], one for configuration register [00D5] and one for the input pins [00D6]. Since G6 and G7 are Hi-Z input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the device will be placed in the Halt mode by writing a "1" to the G7 data bit.

Six pins of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input/general purpose input (if clock option is R/C or external clock)

Pin Description (Continued)

Pins G1 and G2 currently do not have any alternate functions.

The selection of alternate Port G functions are done through registers PSW [00EF] to enable external interrupt and CNTRL1 [00EE] to select TIO and MICROWIRE operations.

PORT D is a four bit output port that is preset when RESET goes low. One data memory address location is allocated for the data register [00DC].

Oscillator Circuits

EXTERNAL OSCILLATOR

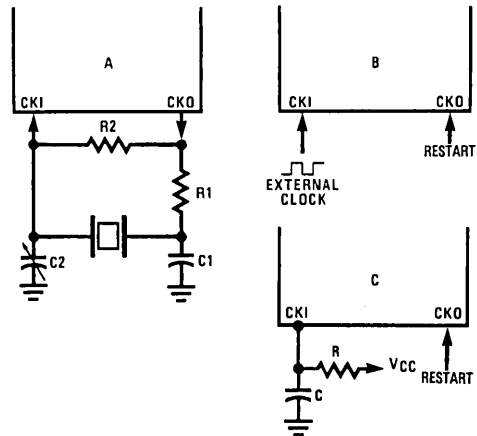
CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels. CKO is available as a general purpose input G7 and/or Halt control.

CRYSTAL OSCILLATOR

By selecting CKO as a clock output, CKI and CKO can be connected to create a crystal controlled oscillator. Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator, CKI can make a R/C oscillator. CKO is available as a general purpose input and/or HALT control. Table II shows variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/11373-4

FIGURE 3. Clock Oscillator Configurations

TABLE I. Crystal Oscillator Configuration

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq. (MHz)	Conditions
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4	$V_{CC} = 5V$
5.6	1	100	100-156	0.455	$V_{CC} = 5V$

TABLE II. RC Oscillator Configuration (Part-To-Part Variation)

R (k Ω)	C (pF)	CK1 Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	$V_{CC} = 5V$
5.6	100	1.1 to 1.3	7.4 to 9.0	$V_{CC} = 5V$
6.8	100	0.9 to 1.1	8.8 to 10.8	$V_{CC} = 5V$

Programming the COP820CJMH/ COP822CJMH

Programming the hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the NSC development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the proper value specified in the following table

Device	Package Type	Contents of Last Byte (Address 01FFF)
COP820CJMH	28 DIP	6F
COP822CJMH	20 DIP	E7

ERASING THE PROGRAM MEMORY

Erasure of the EPROM program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the device are such that the erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000 range.

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the devices is exposure to short wave ultraviolet light which as a wavelength of 2537Å. The integrated dose (UV intensity \times exposure time) for erasure should be a minimum of 15 w-sec/cm².

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

Minimum Erasure Time		
Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)	
	28-Pin Package	20-Pin Package
15,000	20	40
10,000	25	50
5,000	50	100

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kBytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER's performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down-menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2 kBAud serial link keeps typical program download time to under 3 seconds.

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS 232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Development Support (Continued)**Probe Card Ordering Information**

Part Number	Package	Voltage Range	Emulates
MH-820CJ20D5PC	20 DIP	4.5V-5.5V	COP822CJ
MHW-820CJ20DWPC	20 DIP	2.3V-6.0V	COP822CJ
MHW-820CJ28D5PC	28 DIP	4.5V-5.5V	COP820CJ
MHW-820CJ28DWPC	28 DIP	2.3V-6.0V	COP820CJ

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE-COP8-IBM	COP8 macro cross assembler for IBM® PC-XT®, PC-AT® or compatible	424410527-001

SINGLE CHIP EMULATOR

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP820CJMHD-X	X = 1: crystal X = 2: external X = 3: R/C	28 DIP	Multi-Chip Module (MCM), UV erasable	COP820CJ
COP820CJMHEA-X	X = 1: crystal X = 2: external X = 3: R/C	28 LCC	MCM (same footprint as 28 SO), UV erasable	COP820CJ
COP822CJMHD-X	X = 1: crystal X = 2: external X = 3: R/C	20 DIP	MCM, UV erasable	COP822CJ

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator board for 28 DIP and for use with Scrambler Boards	COP820CJMHD
COP8-SCRM-DIP	Scrambler board for 20 DIP socket	COP822CJMHD
COP8-SCRM-SBX	Scrambler board for 28 LCC sockets	COP820CJMHEA
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler board	COP822CJMHD COP820CJMHD

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user has questions, he can leave messages on our electronic bulletin board.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 baud

Setup: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs. 7 Days



COP888CSMH microCMOS Microcontroller Emulator

General Description

The COP888CSMH hybrid emulators is a members of the COPS™ microcontroller family. The device is a two chip system in a dual cavity package. Within the package is the COP888CS and a UV-erasable 8k EPROM with port recreation logic. Code executes out of EPROM. The device is offered in the following packages: 44-pin LDCC, 40-pin DIP, and 28-pin DIP. All packages contain transparent windows which allow the EPROM to be erased and re-programmed.

The COP888CSMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUSTM serial I/O, one 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, one comparator, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CSMH operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate.

These COP888CSMH is primarily intended as a prototyping design tool. The Electrical Performance Characteristics are not tested but are included for reference only.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 8192 bytes on-board EPROM
- 192 bytes on-board RAM
- Single supply operation: 4.5V–5.5V

- Full duplex UART
- One analog comparator
- MICROWIRE/PLUS serial I/O
- WATCHDOG and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- One 16-bit timer, with two 16-bit registers supporting:
 - Processor Independent PWM mode
 - External Event counter mode
 - Input Capture mode
- Ten multi-source vectored interrupts servicing
 - External Interrupt
 - Idle Timer T0
 - Timer (2)
 - MICROWIRE/PLUS
 - Multi-Input Wake Up
 - Software Trap
 - UART (2)
 - Default VJS
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set with true bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 LDCC with 39 I/O pins
40 DIP with 35 I/O pins
28 DIP with 23 I/O pins
- Software selectable I/O options
 - TRI-STATE® Output
 - Push-Pull Output
 - Weak Pull Up Input
 - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CS
- Real time emulation and full program debug offered by National's Development Systems

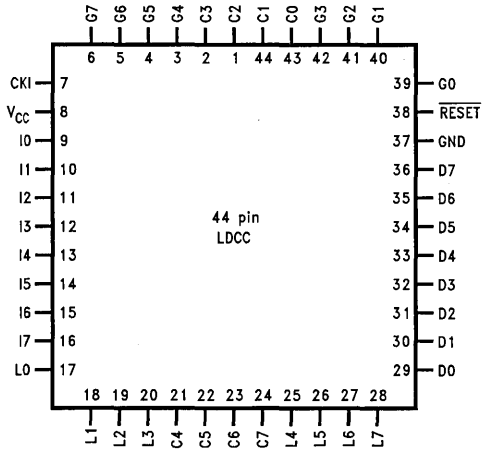
Ordering Information

Hybrid Emulator	Package Type	Part Emulated with Crystal Oscillator Option
COP888CSMHD-X	40-Pin DIP	COP888CS-XXX/N
COP888CSMHEL-X	44-Pin LDCC	COP888CS-XXX/V
COP884CSMHD-X	28-Pin DIP	COP884CS-XXX/N

X indicates Crystal Option: for applications requiring R/C oscillator option check with your local sales representative.

Connection Diagrams

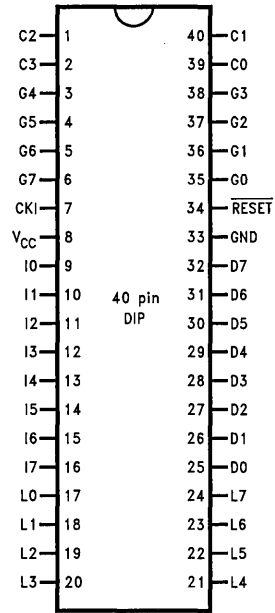
Plastic Chip Carrier



Top View

TL/DD/11387-1

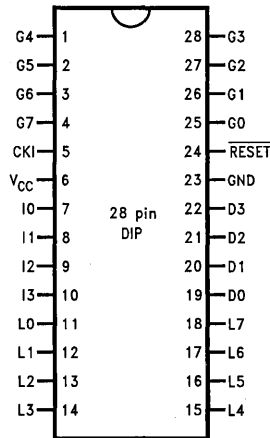
Dual-In-Line Package



Top View

TL/DD/11387-2

Dual-In-Line Package



Top View

TL/DD/11387-3

FIGURE 1. COP888CSMH Connection Diagrams

Connection Diagrams (Continued)

COP888CSMH Pinouts

Port	Type	Alternate Fun	Alternate Fun	28-Pin DIP	40-Pin DIP	44-Pin LDCC
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU		15	21	25
L5	I/O	MIWU		16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT RESTART		4	6	6
I0	I			7	9	9
I1	I	COMP1IN-		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I				13	13
I5	I				14	14
I6	I				15	15
I7	I				16	16
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V _{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
RESET				24	34	38

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	-0.3V to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	-65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

The following AC and DC Electrical Characteristics are not tested but are for reference only.

DC Electrical Characteristics $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1 V_{CC}$	V
Supply Current (Note 2) CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			25	mA
HALT Current (Note 3)	$V_{CC} = 5.5V, CKI = 0 \text{ MHz}$		250		μA
IDLE Current CKI = 10 MHz	$V_{CC} = 5.5V, t_c = 1 \mu s$			15	mA
Input Levels RESET					
Logic High		$0.8 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
CKI (External and Crystal Osc. Modes)					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-2		+2	μA
Input Pullup Current	$V_{CC} = 5.5V$	40		250	μA
G and L Port Input Hysteresis				$0.35 V_{CC}$	V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V, V_{OH} = 3.3V$	0.4			mA
Sink	$V_{CC} = 4.5V, V_{OL} = 1V$	10			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V, V_{OH} = 2.7V$	10		100	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OH} = 3.3V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V, V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage	$V_{CC} = 4.5V$	-2		+2	μA

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC} , L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The comparators and Clock Monitor are disabled.

DC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All Others				15 3	mA mA
Maximum Input Current without Latchup (Note 4)	$T_A = 25^{\circ}\text{C}$			± 100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator		1 3		DC DC	μs μs
CKI Clock Duty Cycle (Note 5) Rise Time (Note 5) Fall Time (Note 5)	$f_r = \text{Max}$ $f_r = 10 \text{ MHz Ext Clock}$ $f_r = 10 \text{ MHz Ext Clock}$	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}		200 60			ns ns
Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others	$R_L = 2.2\text{k}$, $C_L = 100 \text{ pF}$			0.7 1	μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		1 1 1 1			t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 4: Except pin G7: -60 mA to $+100 \text{ mA}$ (sampled but not 100% tested).

Note 5: Parameter sampled (not 100% tested).

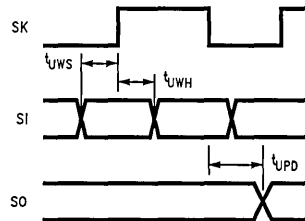


FIGURE 2. MICROWIRE/PLUS Timing

TL/DD/11387-4

Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25^\circ C$

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		± 10	± 25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs

Pin Descriptions

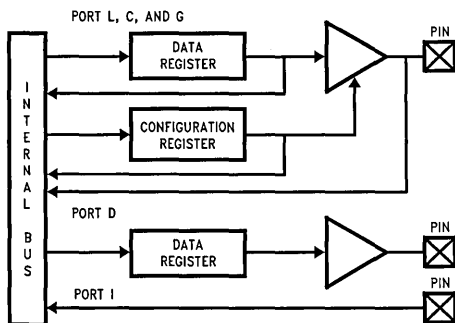
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CSMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. Figure 3 shows the I/O port configurations for the COP888CSMH. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/11387-5

FIGURE 3. I/O Port Configurations

Port L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive.

The Port L has the following alternate features:

- L0 MIWU
- L1 MIWU or CKX
- L2 MIWU or TDX
- L3 MIWU or RDX
- L4 MIWU
- L5 MIWU
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Pin Descriptions (Continued)

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT (WATCHDOG and/or Clock Monitor dedicated output)
- G7 CKO (Oscillator dedicated output or general purpose input)

Port I is an 8-bit port. Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

- I1 COMP1–IN (Comparator 1 Negative Input)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I3 COMP1OUT (Comparator 1 Output)
- I4 COMP2–IN (Comparator 2 Negative Input)
- I5 COMP2+IN (Comparator 2 Positive Input)
- I6 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Port C is an 8-bit I/O port.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 4 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (SPECIAL ORDER FROM FACTORY)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.

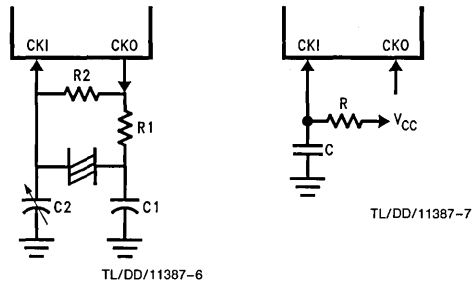


FIGURE 4. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration,
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R1 (k Ω)	R2 (M Ω)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. RC Oscillator Configuration,
 $T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$

R (k Ω)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.2 to 2.7	3.7 to 4.6
5.6	100	1.1 to 1.3	7.4 to 9
6.8	100	0.9 to 1.1	8.8 to 10.8

Programming the COP888CSMH

Programming the COP888CSMH hybrid emulators is accomplished through the duplicator board which is a stand alone programmer capable of supporting different package types. It works in conjunction with a pre-programmed EPROM (either via the development system or a standard programmer) holding the application program. The duplicator board essentially copies the information in the EPROM into the hybrid emulator.

The last byte of program memory (EPROM location 01FFF Hex) must contain the value specified in the following table.

Programming the COP888CSMH

(Continued)

TABLE III

Package	HALT Mode	Contents of Last Byte (Address 01FFF)
28	Enabled	6F
28	Disabled	EF
40/44	Enabled	7F
40/44	Disabled	FF

ERASING THE PROGRAM MEMORY

Erasure of the program memory is achieved by removing the device from its socket and exposing the transparent window to an ultra-violet light source.

The erasure characteristics of the device are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å to 4000Å range.

After programming, opaque labels should be placed over the window of the device to prevent temporary functional failure due to the generation of photo currents, erasure, and excessive HALT current. Note that the device will also draw more current than normal (especially in HALT mode) when the window of the device is not covered with an opaque label.

The recommended erasure procedure for the device is exposure to short wave ultraviolet light which has a wavelength of 2537Å. The integrated dose (UV intensity \times exposure time) for erasure should be a minimum of 15 W-sec/cm².

The device should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The following table shows the minimum erasure time for various light intensities.

TABLE IV. Minimum COP888CSMH Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	20
10,000	25
5,000	50

An erasure system should be calibrated periodically. The distance from lamp to device should be maintained at one inch. The erasure time increases as the square of the distance. Lamps lose intensity as they age. When a lamp has aged, the system should be checked to make certain that adequate UV dosages are being applied for full erasure.

Development Support

IN-CIRCUIT EMULATOR

The MetaLink iceMASTER™-COP8 Model 400 In-Circuit Emulator for the COP8 family of microcontrollers features high-performance operation, ease of use, and an extremely flexible user-interface for maximum productivity. Interchangeable probe cards, which connect to the standard common base, support the various configurations and packages of the COP8 family.

The iceMASTER provides real time, full speed emulation up to 10 MHz, 32 kbytes of emulation memory and 4k frames of trace buffer memory. The user may define as many as 32k trace and break triggers which can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together. Trace information consists of address bus values, opcodes and user selectable probe clips status (external event lines). The trace buffer can be viewed as raw hex or as disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.

During single-step operation the dynamically annotated code feature displays the contents of all accessed (read & write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed.

The iceMASTER'S performance analyzer offers a resolution of better than 6 μ s. The user can easily monitor the time spent executing specific portions of code and find "hot spots" or "dead code". Up to 15 independent memory areas based on code address or label ranges can be defined. Analysis results can be viewed in bargraph format or as actual frequency count.

Emulator memory operations for program memory include single line assembler, disassembler, view, change and write to file. Data memory operations include fill, move, compare, dump to file, examine and modify. The contents of any memory space can be directly viewed and modified from the corresponding window.

The iceMASTER comes with an easy to use windowed interface. Each window can be sized, highlighted, color-controlled, added, or removed completely. Commands can be accessed via pull-down menus and/or redefinable hot keys. A context sensitive hypertext/hyperlinked on-line help system explains clearly the options the user has from within any window.

The iceMASTER connects easily to a PC via the standard COMM port and its 115.2 kBaud serial link keeps typical program download time to under 3 seconds.

Development Support (Continued)

The following tables list the emulator and probe cards ordering information.

Emulator Ordering Information

Part Number	Description
IM-COP8/400	Metalink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS/232 serial interface cable
MHW-PS3	Power Supply 110V/60 Hz
MHW-PS4	Power Supply 220V/50 Hz

Probe Card Ordering Information

Part Number	Package	Voltage Range	Emulates
MHW-884CS28D5PC	28 DIP	4.5V-5.5V	COP884CS
MHW-884CS28DWPC	28 DIP	2.5V-6.0V	COP884CS
MHW-888CS40D5PC	40 DIP	4.5V-5.5V	COP888CS
MHW-888CS40DWPC	40 DIP	2.5V-6.0V	COP888CS
MHW-888CS44D5PC	44 PLCC	4.5V-5.5V	COP888CS
MHW-888CS44DWPC	44 PLCC	2.5V-6.0V	COP888CS

MACRO CROSS ASSEMBLER

National Semiconductor offers a COP8 macro cross assembler. It runs on industry standard compatible PCs and supports all of the full-symbolic debugging features of the MetaLink iceMASTER emulators.

Assembler Ordering Information

Part Number	Description	Manual
MOLE™-COP8-IBM	COP8 Macro Cross Assembler for IBM® PC/XT®, PC/AT® or Compatible	424410527-001

SINGLE CHIP EMULATOR DEVICE

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific data sheets and the form, fit, function emulator selection table below.

PROGRAMMING SUPPORT

Programming of the single chip emulator devices is supported by different sources. National Semiconductor offers a duplicator board which allows the transfer of program code from a standard programmed EPROM to the single chip emulator and vice versa. Data I/O supports COP8 emulator device programming with its uniSite 48 and System 2900 programmers. Further information on Data I/O programmers can be obtained from any Data I/O sales office or the following USA numbers:

Telephone: (206) 881-6444 Fax: (206) 882-1043

Single Chip Emulator Selection Table

Device Number	Clock Option	Package	Description	Emulates
COP888CSMHXL-X	X=1 : Crystal X=3 : R/C	44 LDCC	Multi-Chip Module (MCM), UV Erasable	COP888CS
COP888CSMHDX-X	X=1 : Crystal X=3 : R/C	40 DIP	MCM, UV Erasable	COP888CS
COP884CSMHDX-X	X=1 : Crystal X=3 : R/C	28 DIP	MCM, UV Erasable	COP884CS
COP884CSMHDX-X	X=1 : Crystal X=3 : R/C	28 LCC	MCM (Same Footprint as 28 SO), UV Erasable	COP884CS

Duplicator Board Ordering Information

Part Number	Description	Devices Supported
COP8-PRGM-28D	Duplicator Board for 28 DIP Multi-Chip Module (MCM) and for use with Scrambler Boards	COP884CSMHD
COP8-SCRM-DIP	MCM Scrambler Board for 40 DIP Socket	COP888CSMHD
COP8-SCRM-PCC	MCM Scrambler Board for 44 PLCC/LDCC	COP888CSMHXL
COP8-SCRM-SBX	MCM Scrambler Board for 28 LCC Socket	COP884CSMHDX
COP8-PRGM-DIP	Duplicator Board with COP8-SCRM-DIP Scrambler Board	COP884CSMHD, COP888CSMHD
COP8-PRGM-PCC	Duplicator Board with COP8-SCRM-PCC Scrambler Board	COP888CSMHXL, COP884CSMHD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit

Parity: None

Stop Bit: 1

Operation: 24 Hrs., 7 Days

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user has questions he can leave messages on our electronic bulletin board, which we will respond to.



Section 3
COPS Applications



Section 3 Contents

COP Brief 2 Easy Logarithms for COP400	3-3
COP Brief 6 RAM Keep-Alive	3-14
COP Note 1 Analog to Digital Conversion Techniques with COPS Family Microcontrollers	3-15
COP Note 4 The COP444L Evaluation	3-47
COP Note 5 Oscillator Characteristics of COPS Microcontrollers	3-52
COP Note 6 Triac Control Using the COP400 Microcontroller Family	3-69
COP Note 7 Testing of COP400 Family Devices	3-77
AB-3 Current Consumption in NMOS COPS Microcontrollers	3-86
AB-4 Further Information on Testing of COPS Microcontrollers	3-88
AB-6 COPS Interrupts	3-90
AB-15 Protecting Data in Serial EEPROMs	3-91
AN-326 A User's Guide to COPS Oscillator Operation	3-93
AN-329 Implementing an 8-Bit Buffer in COPS	3-97
AN-338 Designing with the NMC9306/COP494 a Versatile Simple to Use EEPROM	3-101
AN-400 A Study of the Crystal Oscillator for CMOS-COPS	3-107
AN-401 Selecting Input/Output Options on COPS Microcontrollers	3-111
AN-440 New CMOS Vacuum Fluorescent Drivers Enable Three Chip System to Provide Intelligent Control of Dot Matrix V.F. Display	3-121
AN-452 MICROWIRE Serial Interface	3-131
AN-454 Automotive Multiplex Wiring	3-142
AN-521 Dual Tone Multiple Frequency (DTMF)	3-146
AN-579 MICROWIRE/PLUS Serial Interface for COP800 Family	3-155
AN-596 COP800 MathPak	3-167
AN-607 Pulse Width Modulation A/D Conversion Techniques with COP800 Family Microcontrollers	3-203
AN-662 COP800 Based Automated Security/Monitoring System	3-210
AN-663 Sound Effects for the COP800 Family	3-218
AN-666 DTMF Generation with a 3.58 MHz Crystal	3-241
AN-673 2-Way Multiplexed LCD Drive and Low Cost A/D Converter Using V/F Techniques with COP8 Microcontrollers	3-269
AN-681 PC MOUSE Implementation Using COP800	3-288
AN-714 Using COP800 Devices to Control DC Stepper Motors	3-313
AN-734 MF2 Compatible Keyboard with COP8 Microcontrollers	3-323
AN-739 RS-232C Interface with COP800	3-343
AN-749 Quadrature Signal Interface to a COP400 Microcontroller	3-355

Easy Logarithms for COP400

National Semiconductor
COP Brief 2



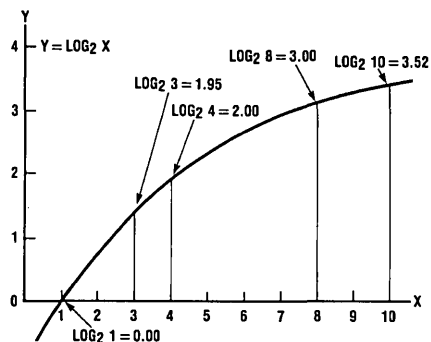
Logarithms have long been a convenient tool for the simplification of multiplication, division, and root extraction. Many assembly language programmers avoid the use of logarithms because of supposed complexity in their application to binary computers. Logarithms conjure up visions of time consuming iterations during the solution of a long series. The problem is far simpler than imagined and its solution yields, for the applications programmer, the classical benefits of logarithms:

- 1) Multiplication can be performed by a single addition.
- 2) Division can be performed by a single subtraction.
- 3) Raising a number to a power involves a single multiply.
- 4) Extracting a root involves a single divide.

When applied to binary computer operation logarithms yield two further important advantages. First, a broad range of values can be handled without resorting to floating point techniques (other than implied by the characteristic). Second, it is possible to establish the significance of an answer during the body of a calculation, again, without resorting to floating point techniques.

Implementation of base₁₀ logarithms in a binary system is cumbersome and unnecessary since logarithmic functions can be implemented in a number system of any base. The techniques presented here deal only with logarithms to the base₂.

A logarithm consists of two parts: an integer characteristic and a fractional mantissa.



TL/DD/6942-1

	CHARACTERISTIC	MANTISSA
LOG ₂ 3 =	1	0.95
LOG ₂ 4 =	2	0.00
LOG ₂ 8 =	3	0.00
LOG ₂ 10 =	3	0.52

FIGURE 1. The Logarithmic Function and Some Example Values

In *Figure 1* some points on the logarithmic curve are identified and evaluated to the base₂. Notice that the characteristic in each case represents the highest even power of 2 contained in the value of X. This is readily seen when binary notation is used.

X ₁₀	X ₂	Log ₂ X	Log ₂ X Where X =			
2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	Characteristic	Even Power of 2
3	0	0	0	1	1	
4	0	0	1	0	2	010.0000
8	0	1	0	0	3	011.0000
10	0	1	0	1	3	

FIGURE 2. Identification of the Characteristic

In *Figure 2* each point evaluated in *Figure 1* has been repeated using binary notation. An arrow subscript indicates the highest even power of 2 appearing in each value of X. Notice that in X = 3 the highest even power of 2 is 2¹. Thus the characteristic of the log₂ 3 is 1. Where X = 10 the characteristic of the log₂ 10 is 3.

To find the log₂ X is very easy where X is an even power of 2. We simply shift the value of X left until a carry bit emerges from the high order position of the register. This procedure is illustrated in *Figure 3*. This characteristic is found by counting the number of shifts required and subtracting the result from the number of bits in the register. In practice it is easier to be going with the number of bits and count down once prior to each shift.

Counter for Characteristic	Value of X in Binary		
1	0 0 0 0	1 0 0 0	Initial
0	1 1 1	0 0 0 0	First Shift
0	1 1 0	0 0 1 0	Second Shift
0	1 0 1	0 1 0 0	Third Shift
0	1 0 0	1 0 0 0	Fourth Shift
0	0 1 1	0 0 0 0	Fifth Shift
Characteristic	Mantissa		Final
0 1 1 . 0 0 0 0	0 0 0 0		Log ₂ X = 3.00

FIGURE 3. Conversion to Base₂ Logarithm by Base Shift

Examination of the final value obtained in *Figure 3* reveals no bits in the mantissa. The value 3 in the characteristic, however, indicates that a bit did exist in the 2³ position of the original number and would have to be restored in order to reconstruct the original value (antilog).

The log of any even power of 2 can be found in this way:

Decimal	Binary	Log ₂
128	10000000	0111.00000000
64	01000000	0110.00000000
32	00100000	0101.00000000
4	00000100	0010.00000000
2	00000010	0001.00000000
1	00000001	0000.00000000

A simple flow chart, and program, can be devised for generating the values found in the table and, as will be apparent, a straight line approximation for values that are not even powers of 2. The method, as already illustrated in *Figure 3*, involves only shifting a binary number left until the most significant bit moves into the carry position. The characteristic is formed by counting. Since a carry on each successive shift will yield a decreasing power of 2, we must start the characteristic count with the number of bits in the binary value (x) and count down one each shift.

FIGURE 4. Base₂ Logarithms of Even Powers of 2

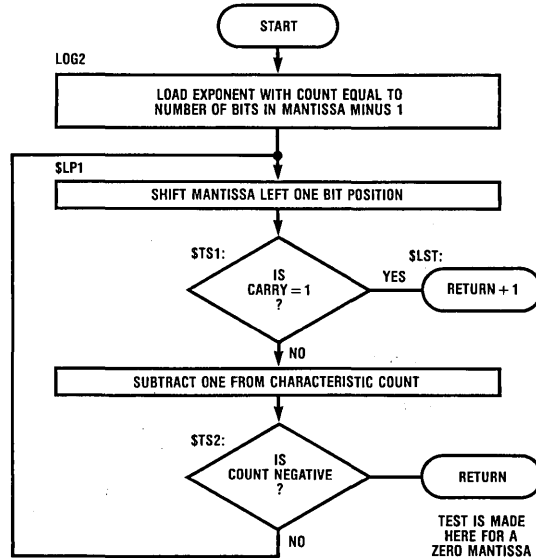


FIGURE 5. Log Flowchart

TL/DD/6942-2

```

1 ; TITLE LOGS ; BINARY LOGARITHMS
2
3 01A4 . CHIP 420
4
5 ; ----- CONVERT TO LOGARITHM -----;
6
7 ; RAM ASSIGNMENT
8
9 ; DIGIT: 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
10
11 ; REG 0
12
13 ; REG 1
14
15 ; REG 2
16
17 ; REG 3
18
19
20 . LOCAL
21
22 ; CH, HM, LM REPRESENT ANY THREE SEQUENTIAL MEMORY DIGITS. THEY
23 ; MAY BE DEFINED IN ANY REGISTER. THE SYMBOLIC NOTATION CH, HM,
24 ; AND LM ARE USED FOR ADDRESSING TO ALLOW USER FLEXIBILITY.
25 ; UPON ENTRY TO THE ROUTINE HM AND LM CONTAIN THE HI AND LO
26 ; OF SOME VALUE X. THE MEMORY POINTER MUST CONTAIN THE ADDRESS
27 ; OF THE CHARACTERISTIC (CH). THE CONTENTS OF THIS LOCATION ARE
28 ; IGNORED AND ARE LOST DURING EXECUTION.
29
30 ; UPON EXIT CH, HM, LM CONTAIN A STRAIGHT LINE APPROXIMATION OF
31 ; THE LOG BASE 2 OF X. CH = CHARACTERISTIC HM = HI ORDER MANTISSA
32 ; LM = LO ORDER MANTISSA. AN 8 BIT MEMORY AREA (TEMP) IS USED IN
33 ; THE REGISTER OPPOSITE DURING THE CORRECTION OF A STRAIGHT
34 ; LINE APPROXIMATION OF A LOG OR AN ANTILOG.
35 ; A TEST IS MADE FOR X=0. IF THE VALUE OF X
36 ; IS NOT ZERO AN INSTRUCTION IS SKIPPED UPON RETURN
37 ; TO THE CALLING ROUTINE.
38
39 ; ----- EXAMPLE -----
40
41 ; SUBROUTINE CALL JSR LOG2
42 ; RETURN HERE IF X = 0 ----> JP ZERO
43 ; RETURN HERE IF X > 0 ----> CONTINUE
44
45
46
47
48
49
50 000 00 LOG2: CLRA ; SET CHARACTERISTIC.
51 001 57 AISC 07 ; TO REG LENGTH - 1.
52 002 06 X ; STORE IN MEMORY.

```

									CH	HM	LM				TEMP
			CH	HM	LM										TEMP
					TEMP								CH	HM	LM
										TEMP			CH	HM	LM

```

53 003 A4 $LP1: JSRP SDB2 ; SET ADDRESS POINTER
54 ; BACK 2 DIGITS.
55 004 A9 JSRP SHLR ; RESET CARRY AND SHIFT
56 ; REG LEFT ONE BIT.
57 005 20 $TS1: SKC ; IS CARRY = 1 YET?
58 006 C8 JP $NO ; NO - KEEP GOING.
59 007 49 $LST: RETSK ; YES - FINISHED!!
60 008 05 $NO: LD ; NO - LOAD COUNT IN ACC.
61 009 5F AISC -1 ; SUBTRACT ONE.
62 00A 48 $TS2: RET ; MANTISSA IS A 0! RETURN
63 00B 06 X ; STORE CHARACTERISTIC.
64 00C C3 JP $LP1 ; DO IT AGAIN!
65
66
67
68
69
70 ; 2 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
71 ; PROGRAM: SDB2, SHLR.
72

```

FIGURE 6

The program shown develops the \log_2 of any even power of 2 by shifting and testing as previously described. Examine what happens to a value of X that is not an even power of 2. In Figure 7, the number 25 is converted to a base 2 log.

$$25_{10} = 00011002_2$$

Shift left until carry = 1

Characteristic	Carry	Mantissa	Log ₂
0100	1	100100000100	.10010000

Figure 7. Straight Line Approximation of Base₂ Log

The resulting number when viewed as an integer characteristic and a fractional mantissa is 4.5625_{10} . The fraction 0.5625 is a straight line approximation of the logarithmic curve between the correct values for the base₂ logs of 2⁴ and 2⁵. The accuracy of this approximation is sufficient for many applications. The error can be corrected, as will be seen later in this discussion, but for now let's look at the problem of exponents or the conversion to an antilog.

To reconstruct the original value of X, find the antilog, requires only restoration of the most significant bit and then its alignment with the power of 2 position indicated by the characteristic. In the example, approximation ($\log_2 25 = 0100.1001$) restoration of MSB can be accomplished by shifting the mantissa (only) one position to the right. In the process a one is shifted into the MSB position.

Approximation of Log ₂ X	Restoration of MSB
Char. Mantissa	Char. Mantissa
0100.10010000	0100.11001000

The value of the characteristic is 4 so the mantissa must be shifted to the right until MSB is aligned with the 2⁴ position.

2⁷ 2⁶ 2⁵ 2⁴ 2³ 2² 2¹ 2⁰

The completion of this operation restores the value of X ($X = 25$) and is the procedure used to find an antilog. Figure 8 is a flow chart for finding an antilog using this procedure. This implementation in source code is shown in Figure 9.

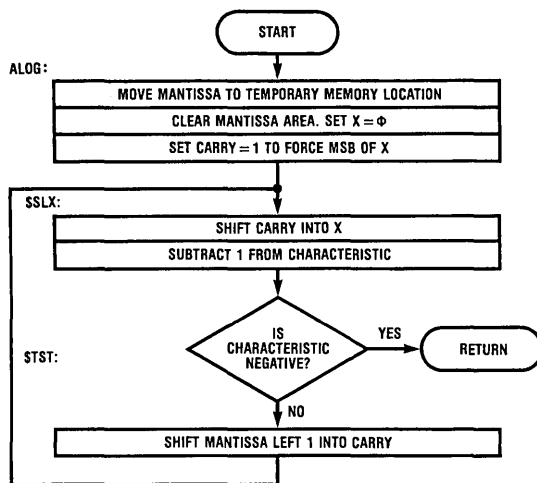


FIGURE 8. Flow Chart for Conversion to Antilog

TL/DD/6942-3

COP CROSS ASSEMBLER PAGE 3
LOGS

```

73          . FORM          ; ***** CONVERT TO ANTILOG ***** ;
74
75
76          ; THE FOLLOWING SUBROUTINE CONVERTS THE STRAIGHT LINE
77          ; THE APPROXIMATION OF A BASE 2 LOGARITHM TO ITS CORRESPONDING
78          ; ANTILOG. UPON EXIT FROM THE ROUTINE THE CONTENTS OF CH
79          ; WILL BE EQUAL TO THE HEXADECIMAL VALUE OF '0F'.
80
81          . LOCAL
82
83
84          00D  A4          ALOG:   JSRP          SDB2          ; SET ACC TO 0.
85          00E  00          CLRA          ; CLEAR MANTISSA AREA.
86          00F  36          X          03          ; AND MOVE MANTISSA TO
87          010  34          XIS         03          ; TEMPORARY STORAGE.
88          011  00          CLRA          ; LEAVE POINTER AT LO
89          012  36          X          03          ; ORDER OF MANTISSA.
90          013  37          XDS         03
91          014  22          SC          ; RESTORE MSB OF X.
92          015  D8          JP          $SLX
93          01  A9          $SLM:   JSRP          SHLR          ; SHIFT REMAINDER
94          ; LEFT INTO CARRY.
95          017  A3          JSRP          SDR2          ; MOVE BACK 2 DIGITS.
96          018  AA          $SLX:   JSRP          SHLC          ; SHIFT X LEFT 1.
97          019  05          LD          ; LOAD CHARACTERISTIC.
98          01A  5F          $TST:   AISC         -1          ; CHARACTERISTIC - 1.
99          01B  48          $LST:   RET          ; IF NO CARRY — FINIS.
100         01C  36          X          03          ; STORE REMAINDER AND MOVE
101         ; DOWN ONE REGISTER.
102         01D  A4          JSRP          SDB2          ; MOVE BACK 2 DIGITS.
103         01E  D6          JP          $SLM          ; DO IT AGAIN.
104
105
106          ; 4 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
107          ; PROGRAM: SDB2, SDR2, SHLR, SHLC.
108
109

```

TL/DD/6942-6

FIGURE 9

Using the linear approximation technique just described, some error will result when converting any value of X that is not an even power of 2.

Figure 10 contains a table of correct base 2 logarithms for values of X from 1 through 32 along with the error incurred for each when using linear approximation. Notice that no error results for values of X that are even powers of 2. Also notice that the error incurred for multiples of even powers of 2 of any given value of X is always the same.

Value of X	Error
5	0.12
$2 \times 5 = 10$	0.12
$4 \times 5 = 20$	0.12
3	0.15
$2 \times 3 = 6$	0.15
$4 \times 3 = 12$	0.15
$8 \times 3 = 24$	0.15

X	Hexadecimal Log Base	Linear Approximation of Log Base 2	Error Hexadecimal	$E_M - 1 + \frac{EM - EM - 1}{2}$
1	0.00	0.00	0.00	
2	1.00	1.00	0.00	
3	1.95	1.80	0.15	
4	2.00	2.00	0.00	
5	2.52	2.40	0.12	
6	2.95	2.80	0.15	
7	2.CE	2.C0	0.0E	
8	3.00	3.00	0.00	
9	3.2B	3.20	0.0B	
10	3.52	3.40	0.12	
11	3.75	3.60	0.15	
12	3.95	3.80	0.15	
13	3.B3	3.A0	0.13	
14	3.CE	3.C0	0.0E	
15	3.E8	3.E0	0.08	
16	4.00	4.00	0.00	
17	4.16	4.10	0.06	0.03
18	4.2B	4.20	0.0B	0.09
19	4.3F	4.30	0.0F	0.0D
20	4.52	4.40	0.12	0.11
21	4.67	4.50	0.17	0.15
22	4.75	4.60	0.15	0.16
23	4.87	4.70	0.17	0.16
24	4.95	4.80	0.15	0.16
25	4.A4	4.90	0.14	0.15
26	4.B3	4.IA0	0.13	0.14
27	4.C1	4.B0	0.11	0.12
28	4.CE;	4.C0	0.0E	0.10
29	4.DB	4.D0	0.0B	0.0D
30	4.E8	4.E0	0.08	0.0A
31	4.F4	4.F0	0.04	0.06
32	5.00	5.00	0.00	0.02
33		5.1-		

FIGURE 10. Error Incurred by Linear Approximation of Base 2 Logs

An error that repeats in this way is easily corrected using a look-up table. The greatest absolute error will occur for the least value of X not an even power of 2, X = 3, is about 8%. A 4 point correction table will eliminate this error but will move the greatest uncompensated error to X = 9 where it

will be about 4%. This process continues until at 16 correction points the maximum error for the absolute value of the logarithm is less than 1 percent. This can be reduced to 0.3 percent by distributing the error. Interpolated error values are listed in *Figure 10* and are repeated in *Figure 11* as a binary table.

High Order 4 Mantissa Bits	Binary Correction Value	Hexadecimal Correction Value
0000	0000 0000	0 0
0001	0000 1001	0 9
0010	0000 1101	0 3
0011	0001 0001	1 1
0100	0001 0101	1 5
0101	0001 0110	1 6
0110	0001 0110	1 6
0111	0001 0110	1 6
1000	0001 0101	1 5
1001	0001 0100	1 4
1010	0001 0010	1 2
1011	0001 0000	1 0
1100	0000 1101	0 D
1101	0000 1010	0 A
1110	0000 0110	0 6
1111	0000 0010	0 2

FIGURE 11. Correction Table for
 L_2 X Linear Approximations

Notice in *Figure 10* that left justification of the mantissa causes its high order four bits to form a binary sequence that always corresponds to the proper correction value. This works to advantage when combined with the COP400 LQID instruction. LQID implements a table look-up function using the contents of a memory location as the address pointer. Thus we can perform the required table look-up without disturbing the mantissa.

Figure 12 is the flow chart for correction of a logarithm found by linear approximation. *Figure 13* is its implementation in COP400 assembly language. Notice that there are two entry points into the program. One is for correction of logs (LADJ:), the other is for correction of a value prior to its conversion to an antilog (AADJ:).

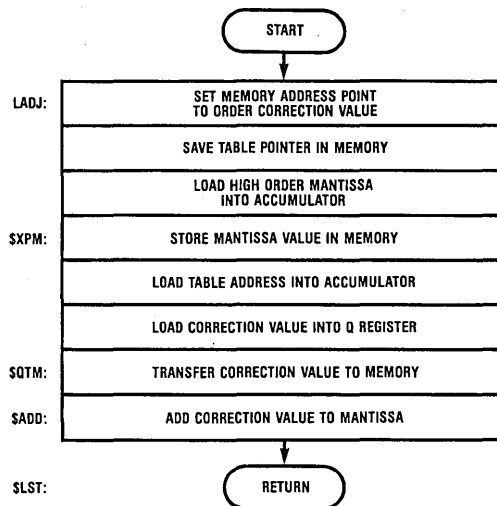


FIGURE 12. Flow Chart for Correction of a Value Found by Straight Line Approximation

TL/DD/6942-4

COP CROSS ASSEMBLER PAGE: 4
LOGS

```

110 . FORM ; ---- ADJUST VALUE OF LOGARITHM ---- ;
111
112 . LOCAL
113
114
115 ; THE FOLLOWING TABLE IS USED DURING THE CORRECTION OF VALUES
116 ; FOUND BY STRAIGHT LINE APPROXIMATION. IT IS PLACED HERE IN
117 ; ORDER TO ALIGN ITS BEGINNING ELEMENT WITH A ZERO ADDRESS AS
118 ; REQUIRED BY THE LQID INSTRUCTION.
119
120 01F 44 NOP ; REGISTER WITH ZERO ADDRESS.
121 020 03 TPLS: . WORD 03,09,0D,011
122 021 09
123 022 0D
124 023 11
125 024 15 . WORD 015,016,016,016
126 025 16
127 026 16
128 027 16
129 028 15 . WORD 015,014,012,010
130 029 14
131 02A 12
132 02B 10
133 02C 0D . WORD 0D,0A,06,02
134 02D 0A
135 02E 06
136 02F 02

```

```

125
126 ; THE FOLLOWING SUBROUTINE ADJUSTS THE VALUE OF A BASE 2
127 ; LOGARITHM FOUND BY STRAIGHT LINE APPROXIMATION. THE
128 ; CORRECTION TERMS ARE TAKEN FROM THE TABLE ABOVE. THE
129 ; SUBROUTINE HAS 2 ENTRY POINTS:
130

```

```

131 :
132 : LADJ: — ADJUSTS A VALUE DURING CONVERSION TO A LOG
133 :
134 : AADJ: — ADJUSTS A VALUE DURING CONVERSION TO ANTILOG
135 :

```

```

136 ; THE CARRY FLAG IS SET UPON ENTRY TO DISTINGUISH BETWEEN LOG
137 ; (C = 1) AND ANTILOG (C = 0) CONVERSIONS. DURING A LOGARITHM
138 ; CONVERSION THE VALUE FOUND IN THE ABOVE TABLE IS ADDED TO
139 ; THE MANTISSA. DURING AN ANTILOG CONVERSION THE VALUE FOUND
140 ; IN THE ABOVE TABLE IS SUBTRACTED FROM THE MANTISSA.
141

```

```

142 030 32 AADJ: RC ; C = 0 FOR ANTILOG
143 031 F3 JP $LD ; CONVERSION.
144 032 22 LADJ: SC ; C = FOR LOG2 ADJ.
145 033 05 $LD LD ; MOVE ADDRESS POINTER BACK
146 034 07 XDS ; ONE LOCATION.
147 035 05 LD ; LOAD CONTENTS OF HI MANTISSA
148 036 37 XDS 03 ; AND STORE IT IN THE LO ORDER
149 037 06 X ; OF THE TEMP MEMORY LOCATION.
150 038 00 CLRA ; SET TABLE POINTER
151 039 52 AISC TBL ; (ACC) TO TABLE ADDRESS.

```

COP CROSS ASSEMBLER PAGE: 5
LOGS

```

152 03A BF LQID ; LOAD CORRECTION VALUE TO Q.
153 03B 332C $GTM: COMA ; TRANSFER Q REGISTER
154 03D 04 XIS ; CONTENTS TO MEMORY.
155 03F 07 XDS
156 03F 20 SKC ; ANTILOG?
157 040 80 JSRP COMP ; YES — COMPLIMENT.
158 041 98 $ADD: JSRP ADRO ; ADD CORRECTION VALUE
159 ; TO MANTISSA.
160 042 35 LD 03 ; SET POINTER TO
161 043 48 $LST: RET ; CHARACTERISTIC AND
162 ; RETURN.
163

```

```

164 ; 2 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
165 ; PROGRAM: COMP, ADRO
166

```

```

167 0020 V1 = TPLS&OFF
168 0002 TBL = V1/16
169
170
171

```

FIGURE 13

Subroutines Used by the Log and Antilog Programs

COP CROSS ASSEMBLER PAGE: 6
LOGS

```

172                                     .FORM
173      0080      . PAGE 02                ;----- SUBROUTINES ----- ;
174
175      ; THE FOLLOWING ROUTINES RESIDE ON THE SUBROUTINE PAGE. THEY
176      ; ARE CALLED BY THE LOGS PROGRAM BUT ARE GENERAL PURPOSE IN
177      ; NATURE AND FUNCTION AS UTILITY ROUTINES.
178
179
180
181      ;----- COMPLEMENT 8 BITS ----- ;
182
183      . LOCAL
184
185      ; THIS ROUTINE FORMS IN MEMORY THE 2'S COMPLEMENT OF THE TWO
186      ; ADJACENT DIGITS IDENTIFIED BY THE ADDRESS POINTER. THE
187      ; CONTENTS OF THE ADDRESS POINTER ARE NOT ALTERED.
188
189      ; THERE ARE TWO ENTRY POINTS:
190      ;
191      ; COP: COMPLEMENT 8 BITS.
192      ;
193      ; CMPE: EXTEND THE COMPLEMENT TO AN ADDITIONAL 8 BITS
194      ;
195
196      080      22      COMP:      SC
197      081      00      CMPE:      CLRA                ; SET MINUEND=0
198      082      06                X                ; AND STORE IN MEMORY.
199      083      10                CASC
200      084      44                NOP
201      085      04                XIS
202      086      00                CLRA                ; SET MINUEND=0
203      087      06                X                ; AND STORE IN MEMORY.
204      083      10                CASC
205      089      44                NOP
206      08A      04                XIS
207      08B      44                NOP                ; AVOID SKIP IF DIGIT 15.
208      08C      A4                JP          SDB2      ; RETURN THRU SDB2
209      ; TO RESTORE POINTER.
210
211
212
213      ;----- ADD 8 BITS IN ADJACENT REGISTERS ----- ;
214
215      . LOCAL
216
217
218
219      ; THIS ROUTINE ADDS TWO BINARY DIGITS (8 BITS) FROM ANY REGISTER
220      ; TO THE CORRESPONDING TWO BINARY DIGITS IN EITHER REGISTER
221      ; IMMEDIATELY ADJACENT. THERE ARE THREE ENTRY POINTS:
222      ;
223      ; LADR: — RESET CARRY AND ADD 2 DIGIT PAIRS

```

TL/DD/6942-8

COP CROSS ASSEMBLER PAGE: 7
LOGS

```

224          ;          LADD: — ADD 2 DIGIT PAIRS WITH UNMODIFIED CARRY
225          ;          ADD1: — ADD 2 SINGLE DIGITS WITH UNMODIFIED CARRY
226
227
228
229
230 08D 32      LADR:      RC          ; RESET CARRY PRIOR TO ADD.
231 08E 15      LADD:      :D          01      ; LD ADDEND AND MOVE TO ADJ REG
232 08F 30          ASC          ; ADD AUGEND.
233 090 44          NOP          ; AVOID CARRY!
234 091 14          XIS          01      ; STORE SUM AND MOVE TO ADDEND
235 092 15      ADD1:      LD          01      ; REPEAT PROCESS
236 093 30          ASC          ; FOR
237 094 44          NOP          ; HIGH ORDER
238 095 14          XIS          01      ; DIGIT.
239 096 44          NOP          ; AVOID SKIP IF DIGIT 15.
240 097 48      $LST:      RET          ; FINISHED — RETURN!!!!
241
242
243
244
245          ; ----- ADD 8 BITS IN OPPOSITE REGISTERS ----- ;
246
247          . LOCAL
248
249
250
251          ; THIS ROUTINE ADDS TWO BINARY DIGITS (8BITS) FROM ANY REGISTER
252          ; TO THE CORRESPONDING TWO BINARY DIGITS IN EITHER REGISTER
253          ; DIRECTLY OPPOSITE. THERE ARE THREE ENTRY POINTS:
254          ;
255          ;          ADR0: — RESET CARRY AND ADD 2 DIGIT PAIRS
256          ;          ADD0: — ADD 2 DIGIT PAIRS WITH UNMODIFIED CARRY
257          ;          AD01: — ADD 2 SINGLE DIGITS WITH UNMODIFIED CARRY
258
259
260
261
262 098 32      ADR0:      RC          ; RESET CARRY PRIOR TO ADD.
263 099 35      ADD0:      LD          03      ; LD ADDEND AND MOVE TO OPP REG
264 09A 30          ASC          ; ADD AUGEND.
265 09B 44          NOP          ; AVOID CARRY!
266 09C 34          XIS          03      ; STORE SUM AND MOVE TO ADDEND.
267 09D 15      AD01:      LD          01      ; REPEAT PROCESS
268 09E 30          ASC          ; FOR
269 09F 44          NOP          ; HIGH ORDER
270 0A0 34          XIS          03      ; DIGIT.
271 0A1 44          NOP          ; AVOID SKIP IF DIGIT 15.
272 0A2 48      $LST:      RET          ; FINISHED — RETURN!!!!
273
274
275          ; ----- SET DIGIT ADDRESS BACK TWO ----- ;
276
277

```

TL/DD/6942-9

```

278 . LOCAL
279
280 ; THIS ROUTINE SUBTRACTS 2 FROM THE CONTENTS OF THE
281 ; DIGIT POINTER (B REGISTER). THE CONTENTS OF THE
282 ; ACCUMULATOR ARE LOST IN THE PROCESS. THE USE OF
283 ; SDB2 ALLOWS ADDRESSING WITHIN THE LOGS SUB
284 ; ROUTINE TO BE RELATIVE TO THE CONTENTS OF THE
285 ; ADDRESS POINTER (B REGISTER) UPON ENTRY.
286 ; SDB2 IS COMMONLY USED IN BYTE OPERATIONS TO RESTORE THE
287 ; DIGIT POINTER TO THE LOW ORDER POSITION.
288 ; THERE ARE TWO ENTRY POINTS:
289 ;
290 ; SDR2: SET DIGIT ADDRESS BACK 2 AND MOVE TO OPPOSITE REGISTER.
291 ;
292 ; SDB2: SET DIGIT ADDRESS BACK 2 RETAINING PRESENT REGISTER.
293
294
295
296 0A3 35 SDR2: LD 03 ; MOVE TO OPPOSITE REGISTER.
297 0A4 4E SDB2: CBA ; PLACE DIGIT COUNT IN ACC.
298 0A5 5E ; SUBTRACT 2.
299 0A6 44 NOP ; SHOULD ALWAYS SKIP.
300 0A7 50 CAB ; PUT DIGIT COUNT BACK.
301 0A8 48 RET ; FINISHED — RETURN!!
302
303
304 ; ----- SHIFT LEFT ----- ;
305
306 . LOCAL
307
308 ; THIS ROUTINE SHIFTS LEFT THE CONTENTS OF TWO MEMORY
309 ; LOCATIONS ONE BIT. THERE ARE THREE ENTRY POINTS:
310
311 ; SHLR: RESETS THE CARRY BEFORE SHIFTING
312 ; IN ORDER TO FILL THE LOW ORDER
313 ; BIT POSITION WITH A 0.
314 ;
315 ; SHLC: SHIFTS THE STATE OF THE CARRY INTO
316 ; THE LOW ORDER BIT POSITION.
317 ;
318 ; SHL1: SHIFTS LEFT THE CONTENTS OF ONLY
319 ; ONE MEMORY LOCATION. THE STATE
320 ; OF THE CARRY IS SHIFTED INTO THE
321 ; LOW ORDER POSITION OF MEMORY.
322
323
324
325 0A9 32 SHLR: RC ; CLEAR CARRY PRIOR TO SHIFT.
326 0AA 05 SHLC: LD ; LOAD FIRST MEM DIGIT.
327 0AB 30 ASC ; DOUBLE IT.
328 0AC 44 NOP ; AVOID SKIP.
329 0AD 04 XIS ; STORE SHIFTED DIGIT.
330 0AE 05 SHL1: LD ; LOAD NEXT MEM DIGIT.
331 0AF 30 ASC ; DOUBLE IT TOO.

```

```

332 0B0 44 NOP ; AVOID SKIP, IF ANY
333 0B1 04 XIS ; STORE SHIFTED DIGIT.
334 0B2 48 $LST: RET ; FINISHED — RETURN!!
335
336
337 . END

```


RAM Keep-Alive

National Semiconductor
COP Brief 6



A COPSM application is a small scale computer system and the design of a power shut-down is not trivial. During the time that power is available, but out of the designed operating range, the system must be prevented from doing anything to harm protected data. This will typically involve some type of external protection of timing circuit.

There is an option on the COP420, 420L, and 410L parts called "RAM Keep-Alive" that provides a separate power supply to the RAM area of the chip via the CKO pin. The application of power to the RAM while the remainder of the chip has been powered down via V_{CC} will keep the RAM "alive".

However, the integrity of data in the RAM is not only a function of power but is also influenced by transient conditions as power is removed and reapplied. During power-on, the Power On Reset (POR) circuit will keep transients from causing changes in the RAM states. The condition of power loss will have some probability of data change if external control is not used.

At some point below the minimum operating voltage certain gates will no longer respond properly while others may still be functional until a much lower voltage. During this transition time any false signal could cause a false write to one or more cells. Another effect could be to turn on multiple address select lines causing data destruction.

Testing the rate of data change is very difficult because it must be done on a statistical basis with many turn/on-turn/off cycles. Two factors have a major bearing on the numbers derived by testing. One is to call any change in a related data block a failure, even though more than one bit in that block may have changed (this latter case may well be due to the "address select mode"). The second factor is that without massive instrumentation it is impossible to examine the data after each power cycle. Indeed, to do so might have caused errors!

By running the power cycle for a period of time and then looking for changes, one could overlook multiple changes thus reducing the error rate. This has been minimized by more frequent checking which indicates that the errors are spread out randomly over time.

With a power supply that drops from 4.5 to 2V in approximately 100 ms, the drop-out rate is 1 in 5k to 6k power cycles. Reducing the voltage fall time will cause an improvement in the number of cycles per drop-out. This will reach a limit condition of a very high number (1 per 1 million?) when the power falls within one instruction cycle (4–10 μs for the 420, 15–40 μs for the "L" parts). Attaining very rapid fall time may cause problems due to the lack of decoupling/bypass capacitance. By inserting an electronic switch between the regulator and V_{CC} of the COP chip one might be able to meet this type of fall time. By implication some type of sensing is required to cause the switching.

The desirable approach is to force the COP reset input to zero before the voltage falls below 4.5V. This provides a drop out rate of approximately 1 in 50k for the "L" parts and 1 in 100k for the 420. By also stopping the clock of the "L" parts they can achieve a drop-out rate similar to the 420. While not perfect, the number of cycles between data error should be considered with respect to the needs of the application.

The external circuitry to control the chip during the power transition has several implementations each one being a function of the application. The simplest hardware is found in a battery powered (automotive) application. The circuit must sense that the switched 12V is falling (e.g., at some value much below 1.7V and still greater than 5V). This can be done by using the unswitched 12V as a reference for a divider to a nominal voltage of 8V. As the switched 12V drops below the reference a detector will turn on a clamp transistor to a series switch, the POR, and/or the clock circuit (Figure 1). It should be noted that this draws current during the absence of the switched 12V circuit.

In non-automotive usage a similar circuit can be used where there is a stable reference voltage available to use with the comparator/clamp. Thus a 3.6V rechargeable Ni-Cad battery could be used as the reference voltage and V_{RAM} if the appropriate divider is used to level shift to this operating range.

In AC line-powered applications, a similar method could be used with the raw DC being sensed for drop. Another method would be to sense that the line had missed 2–3 cycles either by means of a charge pump or peak detection technique. This will provide the signal to turn on the clamp. One must make this faster than the time to discharge the output capacitance of the power supply, thus assuring that the clamp has performed its function before the supply falls below spec value.

In conclusion, to protect the data stored in RAM during power-off cycle, the POR should go low before the V_{CC} power drops below spec and come up after V_{CC} is within spec. The first item must be handled with an external circuit like Figure 1 and the latter by an RC per the data sheet.

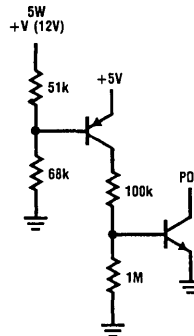


FIGURE 1

TL/DD/6946-1

Analog to Digital Conversion Techniques With COPS™ Family Microcontrollers

National Semiconductor
COP Note 1
Leonard A. Distaso



TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 SIMPLE CAPACITOR CHARGE TIME MEASUREMENT

- 2.1 Basic Approach
- 2.2 Accuracy Improvements
- 2.3 Conclusions

3.0 PULSE WIDTH MODULATION (DUTY CYCLE) TECHNIQUE

- 3.1 Mathematical Analysis
- 3.2 Basic Implementation
- 3.3 Accuracy Improvements

4.0 DUAL SLOPE INTEGRATION TECHNIQUES

- 4.1 Mathematical Background
- 4.2 Basic Dual Slope Technique
- 4.3 Modified Dual Slope Technique

5.0 VOLTAGE TO FREQUENCY CONVERTER, VCO'S

- 5.1 Basic Approach
- 5.2 The LM131/LM231/LM331
- 5.3 Voltage Controlled Oscillators
- 5.4 A Combined Approach

6.0 Successive Approximation

- 6.1 Basic Approach
- 6.2 Some Comments on Resistor Ladders

7.0 "OFFBOARD" TECHNIQUES

- 7.1 General Comments
- 7.2 ADC0800 Interface
- 7.3 ADC0801/2/3/4 Interface (COP431/32/33/34)

8.0 CONCLUSION

9.0 REFERENCES

1.0 Introduction

A variety of techniques for performing analog to digital conversion are presented. The COP420 microcontroller is used as the control element in all cases. However, any of the COPS family of microcontrollers could be used with only minor changes in some component values to allow for different instruction cycle times.

Indirect analog to digital converters are composed of three basic building blocks:

- D/A Converter
- Comparator
- Control logic

In a software driven system the D/A converter and comparator are present but the control logic is replaced by instruction sequences. There are a variety of software/hardware techniques for implementing A/D converters. They differ primarily in their approach to the included D/A. There are two primary approaches to the digital to analog conversion which can in turn be divided into a number of sub-categories:

- D/A as a function of weight closures
 - R/2R ladder
 - Binary weighted ladder
- D/A as function of time
 - RC exponential charge
 - Linear charge/discharge (dual slope)
 - Pulse width modulation

These techniques should be generally familiar to persons skilled in the electronic art. The objective here is to illustrate the application of these established methods to a low cost system with a COPS microcontroller as the intelligent control element. Circuit configurations are provided as well as the appropriate flow charts and code to implement the function.

Some mathematical and theoretical analysis is presented as an aid to understanding the various techniques and their limits. However, it is not the purpose here to provide a definitive theoretical analysis of the analog to digital conversion process or of the various techniques described.

2.0 Simple Capacitor Charge Time Measurement

2.1 BASIC APPROACH

General

Perhaps the simplest means to perform an analog to digital conversion is to charge a capacitor until the capacitor voltage is equal to the unknown voltage. The capacitor voltage and the unknown are compared by means of a standard analog comparator. The unknown is determined simply by counting, in the microcontroller, the amount of time it takes for the charge on the capacitor to reach a value equal to the unknown voltage. The capacitor voltage is given by the standard capacitor charge equation:

$$V_C = V_0 + [V_1 - V_0][1 - e^{-(t/RC)}]$$

where: V_C = capacitor voltage

V_0 = "discharge voltage" — low level voltage

V_1 = high level voltage

The most obvious problem with this method, from the standpoint of software implementation, is the nonlinearity of the

relationship. This can be circumvented in several ways. First of all, a routine to calculate the exponential can be implemented. This, however, usually requires too much code if the exponential routine is not otherwise required in the program. Alternatively, the range of input voltages can be restricted so that only a portion of the capacitor charge curve — which can be approximated with a linear relationship or with some minor straight time curve fitting — is used. Finally, a look up table can be used which will effectively convert the measured time to the appropriate voltage. The look up table has the advantage that all the math can be built into the table, thereby simplifying matters significantly. If arithmetic routines are going to be used, it is clear that the relationship is simplified if V_0 is 0V because it then drops out the equation.

BASIC CIRCUIT IMPLEMENTATION

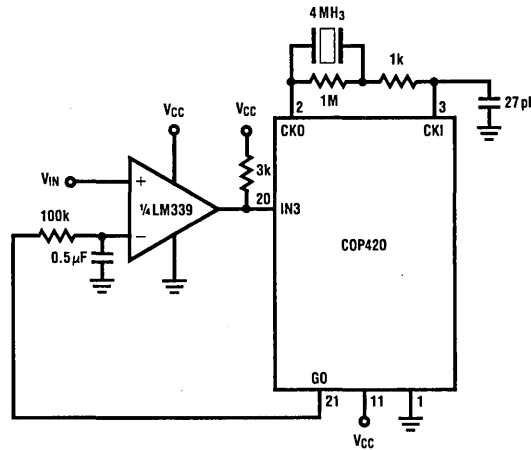
The circuit in *Figure 1* is the basic implementation of the capacitor charge method of A/D conversion. The selection of input and output used is arbitrary and is dictated by general system considerations. V_0 is the "0" level of the G output and V_1 is the "1" level of the output. The technique is basically to discharge the capacitor to V_0 (which is ideally ground) and then to apply V_1 and increment an internal counter until the comparator changes state. The flow chart and code for this implementation are shown in *Figure 2*.

ACCURACY CONSIDERATIONS

The levels reached by the microcontroller output constitute one of the more significant problems with this basic imple-

mentation. The levels of V_1 and V_0 are not V_{CC} and ground as would be desired. The level is defined by the load on the output, the value of V_{CC} , and the device itself. Furthermore, these levels are likely to change from device to device and over temperature. To be sure, the output values will be at least those given in the data sheet, but it must be remembered that those values are minimum high voltages and maximum low voltages. Typically, the high value will be greater than the spec minimum and the low value will be lower than the spec maximum. In fact, with a light load the values will be close to V_{CC} and ground. Therefore, in order to obtain any accurate result for a voltage measurement the exact values of V_1 and V_0 need to be measured and somehow stored in the microcontroller. Typical values of these voltages can be measured experimentally and an average could be used for final implementation.

The other problem associated with the levels is that the capacitive load on the output line is substantial and far in excess of the values used when specifying the characteristics of the various COP420 outputs. The significant effect of this is that it will take longer than "normal" for the output to reach its maximum value. In addition, it is likely that there will be dips in the output as it rises to its maximum value since the capacitor will start to draw charging current from the output. All of this will be fast relative to the other system times. Still it will affect the result since the level to which the capacitor is attempting to charge is not being applied uniformly and "instantaneously". It can be viewed as though the voltage V_1 is bouncing before it stabilizes.



Crystal oscillator values chosen to give 4 µs cycle time with divide by 16 option selected on COP 420 CKO/CKI Pins

$V_{CC} = +5V$

FIGURE 1. Basic Capacitor Charge Technique

TL/DD/6935-01

```

DCI 0 ;TURN OFF G TO DISCHARGE CAPACITOR
; INSERT SOME DELAY TO MAKE SURE CAPACITOR DISCHARGED
; USING 12 BIT COUNTER, BUT ONLY UPPER 8 USED IN TABLE
; LOOK UP DUE TO ACCURACY OF RC CHARGE METHOD. THE OTHER
; BITS COULD BE USED BUT THE COMPLICATIONS ARE NOT WORTH
; THE EFFORT FOR THIS PARTICULAR TECHNIQUE. ALSO, HERE THE
; INPUT RANGE IS RESTRICTED SO THAT THE TOP 3 BITS ARE ZERO

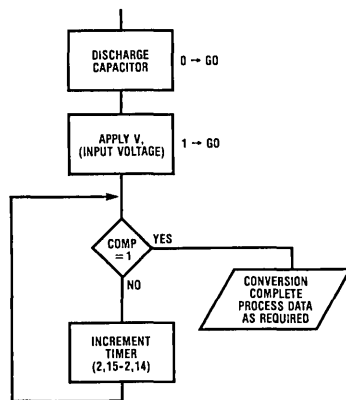
RCAD: DCI 1 ;TURN ON THE G LINE
LBI 2,13 ;BINARY INCREMENT OF 12 BIT COUNTER
BINPL1: SC ;LOWER FOUR BITS WILL BE DISCARDED
; ONLY TOP BITS USED IN TABLE LOOK UP
BINPL1: CLR A ;SPEED WOULD BE IMPROVED IF THE ADD HERE
ASC ;STRAIGHT LINE CODED--BUT COSTS MORE CODE
NOP
XIS
JP
JP BINPL1
ININ ;READ IN3 TO SEE IF COMPARATOR CHANGED
AISC 8
JP END
CLRA
JP INCR
END: DCI 0 ;TURN OFF THE G LINE AND DISCHARGE C
; DO ARITHMETIC HERE OR LOOK UP TABLE OR WHATEVER IS
; REQUIRED--SAMPLE LOOK UP TABLE CONTROL INDICATED BELOW
; SAMPLE TABLE WRITTEN CORRECTING FOR THE EXPONENTIAL
; RELATIONSHIP. THE TABLE ALSO INCORPORATES A CONVERSION
; TO BCD. THE VALUE IN THE TABLE IS THE RATIO OF
; THE CAPACITOR VOLTAGE V TO THE MAXIMUM VOLTAGE VMAX.
; THE NUMBER IS A TWO DIGIT BCD FRACTION. WE ARE USING
; A 5 BIT COUNT IN THIS EXAMPLE. ADDRESSING ARBITRARILY
; SET UP ASSUMING THAT CONTROL CODE IS IN PAGE 0 (OTHER
; THAN AT ADDRESS 0) AND THAT THE TABLE THEREFORE IS IN
; PAGE 1 (STARTING AT HEX ADDRESS 040).
;
LBI 2,15 ;POINT TO TOP 4 BITS
XDS ;TOP 4 IN A, POINTING TO LOWER 4 IN 2,14
AISC 4 ;THIS MERELY ADJUSTING FOR ADDRESS--NO
; OTHER FUNCTION
LDIR ;DO THE LOOK UP
CGMA ;FETCH THE ADJUSTED VALUE FROM G
; THE ADJUSTED VALUE IS NOW IN A AND M. FROM THIS POINT MAY
; USE THE VALUE IN OTHER CALCULATIONS OR OUTPUT THE INFORMATION,
; OR WHATEVER MAY BE REQUIRED BY THE APPLICATION.
LBI 2,13 ;CLEAR THE COUNTER
STII 0
STII 0
STII 0
JP RCAD: ;JUMP BACK AND REPEAT

;=X'040 ;SET UP TABLE ADDRESS
;WORD 000,003,006,008 ;SET UP THE TABLE VALUES
;WORD 011,014,016,019 ;HERE, COMPENSATED FOR EXPONENTIAL
;WORD 021,023,026,028 ;AND CONVERTED TO BCD FRACTION
;WORD 030,032,034,036 ;TABLE VALUE IS RATIO V/VMAX
;WORD 038,039,041,043
;WORD 045,046,048,049
;WORD 051,052,053,055
;WORD 056,057,059,060

```

TL/DD/6935-55

FIGURE 2A. Typical RC Charge A/D Code



TL/DD/6935-2

FIGURE 2B. Charge Flow Chart

A more general problem is that of the tolerance of RC time constant. The value of the voltage with respect to time is obviously related to the RC value. Therefore, a change in that value will result in a change in the voltage for a given time period t . The graph in *Figure 3* illustrates the effect of a $\pm 10\%$ variation in the RC value upon the voltage measured for a given time t . If one cares to work out the math, it comes out that the error is an exponential relationship in much the same manner as the capacitor voltage itself. The maximum error induced for $\pm 10\%$ RC variation is $\pm 3.9\%$.

Remember also that we are measuring time. Therefore variation in the RC value will have a direct, linear effect on the time required to measure a given voltage. It is also necessary that the time base for the COP420 be accurate. A variation in the accuracy in the operating frequency of the COP420 will have a direct impact on the accuracy of the result.

Given the errors mentioned so far and assuming that no changes are made in the hardware, the accuracy of the technique then is determined by the resolution of the time measurement. This is improved in two ways: increase the RC time constant so that there is a smaller change in capacitor voltage for a given time period or try to minimize the loop time required to increment the counter. Lengthening the RC time constant is easier but the cost is increased conversion time. The minimum time to increment a 5 to 8 bit binary counter and test an input is 13 cycle times. For a 9

to 12 bit binary counter this minimum time is 17 cycle times. Note also that the minimum time to perform the function does not necessarily correspond to the minimum number of code words required to implement the function. At a cycle time of $4 \mu\text{s}$, the 13 cycle times correspond to $52 \mu\text{s}$.

2.2 ACCURACY IMPROVEMENTS

Several options are available if it is desired to improve the accuracy of this method. Three such improvements are shown in *Figure 4*. *Figure 4A* is the smallest change. Here a pullup resistor has been added to the G output line and the G line is run open drain internally, i.e., the internal pullup is removed. This improves the "bounce" problem mentioned earlier. The G line will go to the high state and remain there with this setup. However, the addition of the resistor does little more than eliminate the bounce. The degree of improvement is not great, but it is an easy way to eliminate a minor source of error.

Figure 4B is the next step. A 74C04 is used as a buffer. The 74C04 was chosen because of its symmetric output characteristics. Any CMOS gate with such characteristics could be used. The software can easily be adjusted to provide the proper polarity. The COP420 output drives a CMOS gate which in turn drives the RC network. This change does make significant improvements in accuracy. With a light

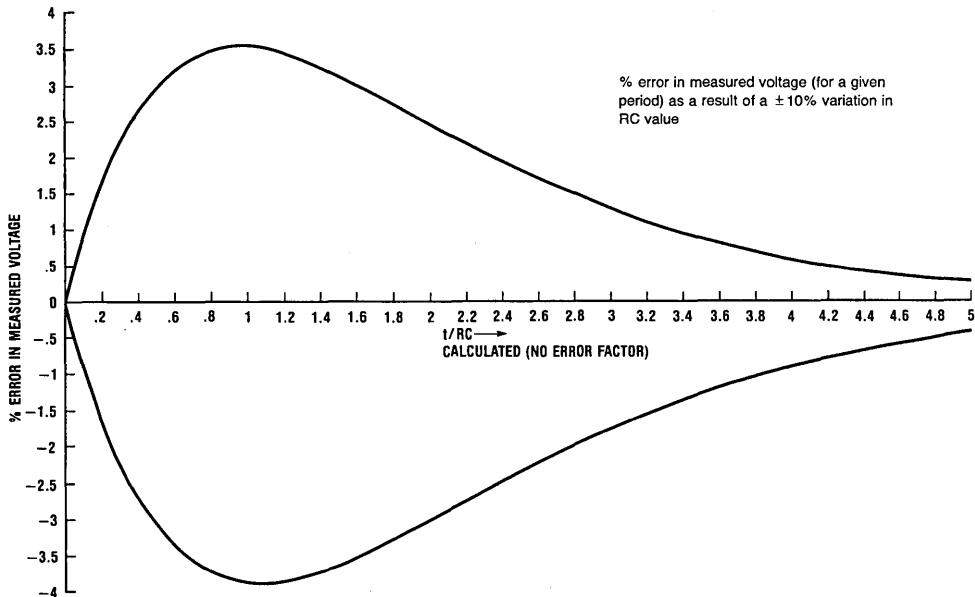


FIGURE 3

TL/DD/6935-3

load the CMOS gate will typically swing from ground to V_{CC} and its output level is not as likely to be affected by the capacitor discharge.

Figure 4C is the best approach, but it involves the greatest component cost. Here two G outputs are controlling analog switches. Ground is connected to the RC network to discharge the capacitor, and a positive reference is used to charge the capacitor. This reference can be any suitable voltage source: zener diodes, V_{CC} , etc. The controlling voltage tolerance is now clearly the tolerance of the reference. Precise voltage references are readily obtainable. Figure 4C also shows an analog switch connected directly across the capacitor to speed up the capacitor discharge time. When using this version of the basic scheme, remember to include the 'on' resistance of the analog switch connected to V_{REF} in the RC calculation. Failure to do so will introduce error into the result.

Note that the LM339 is a quad comparator. If these comparators are not otherwise needed in the system, they can be used in much the same manner as the CMOS gate mentioned above. They can be used to buffer the output of the COPS device and to reset the capacitor, or whatever other function is required. This has the advantage of fully utilizing

the components in the system and eliminates the need to add another package to the system.

2.3 CONCLUSIONS

This approach is an inexpensive way to perform an A/D conversion. However, it is not that accurate. With a 10% V_{CC} supply and a 10% tolerance in the RC value and 10% variation in the oscillator frequency the best that can be hoped for is about 25% accuracy. If a 1% reference voltage is used, this accuracy becomes about 15%.

Under laboratory conditions—holding all variables constant and using precise measured values in the calculations—the configuration of Figure 2 yielded 5 bit accuracy over an input range of 0 to 3.5V. Over the same range and under the same conditions, the circuit of Figure 4B yielded 7 to 8 bit accuracy. It must be emphasized that these accuracies were obtained under controlled conditions. All variables were held constant and actual measured values were used in all calculations. It is unlikely that the general situation will yield these accuracies unless adjustments are provided and a calibration procedure is used. This could defeat the low cost objective.

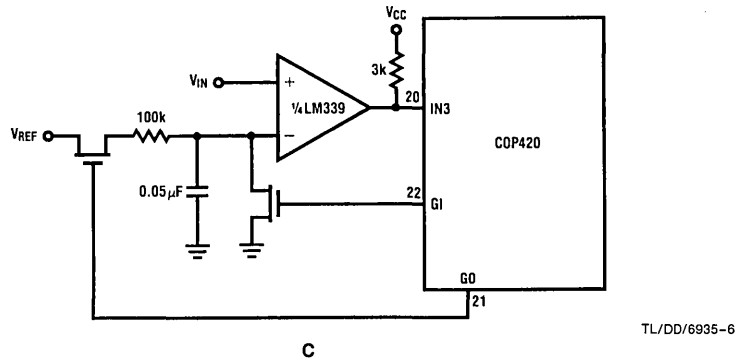
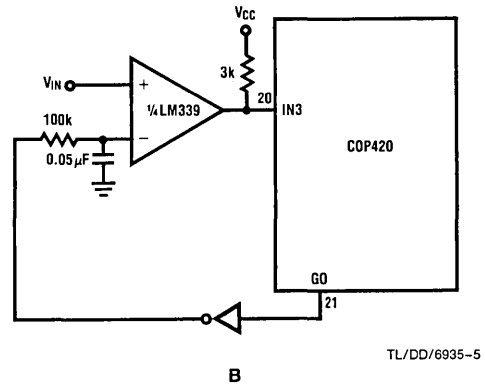
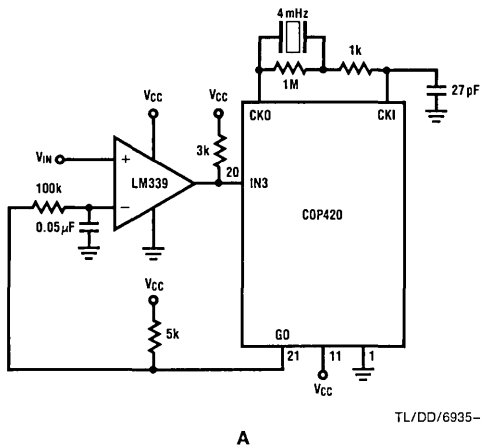


FIGURE 4

3.0 Pulse Width Modulation (Duty Cycle) Technique

3.1 MATHEMATICAL ANALYSIS

The pulse width modulation, or duty cycle, conversion technique is based on the fact that if a repetitive pulse waveform is applied to an RC network, the capacitor will charge to the average voltage of the waveform provided that the RC time constant is sufficiently large relative to the pulse period. See *Figure 5*.

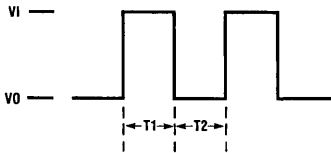
In this technique, the capacitor voltage V_C is compared to the voltage to be measured by means of an analog comparator. The duty cycle is then adjusted to cause V_C to approach the input voltage. The COPS device reads the comparator output and then drives one of its outputs high or low depending on the result, i.e., if V_C is lower than the input voltage, a positive voltage (V_1) is applied to charge the capacitor; if V_C is higher than the input voltage, a lower voltage (V_0) is applied to discharge the capacitor. Thus the capacitor voltage will seek a point where it varies above and below the input voltage by a small amount. *Figure 6* illustrates the capacitor voltage and the comparator output.

Some mathematical analysis here will be useful to help clarify the technique and to point out its restrictions. Referring to *Figure 6*, we have the following:

$$V_A = V_0 + [V_B - V_0][e^{-(t_1/RC)}]$$

$$V_B = V_A + [V_1 - V_A][1 - e^{-(t_2/RC)}]$$

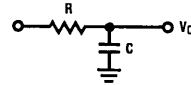
$$= V_1 + [V_A - V_1][e^{-(t_2/RC)}]$$



TL/DD/6935-7

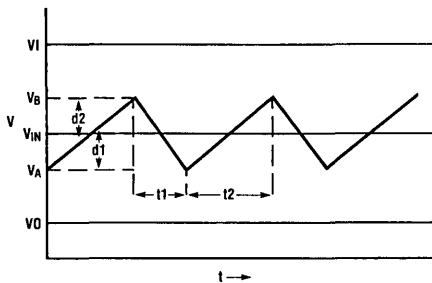
$$V_C = \frac{(V_1 - V_0) \times T_1}{T_1 + T_2}$$

FIGURE 5



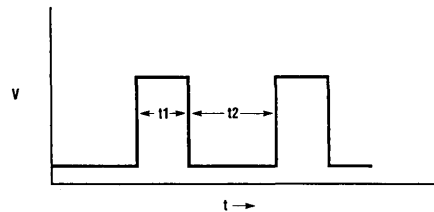
TL/DD/6935-8

Capacitor Voltage



TL/DD/6935-9

Comparator Output



TL/DD/6935-10

FIGURE 6

solving for t_1 and t_2 we have:

$$t_1 = -RC \ln[(V_A - V_0)/(V_B - V_0)]$$

$$t_2 = -RC \ln[(V_B - V_1)/(V_A - V_1)]$$

let:

$$V_A = V_{IN} - d_1$$

$$V_B = V_{IN} - d_2$$

substituting the above, the equations for t_1 and t_2 become:

$$t_1 = -RC \ln\left\{\frac{[1 - (d_1/(V_{IN} - V_0))]/[1 + d_2/(V_{IN} - V_0)]}{[1 - d_1/(V_{IN} - V_1)]}\right\}$$

$$t_2 = -RC \ln\left\{\frac{[1 - (d_2/(V_{IN} - V_1))]/[1 - d_1/(V_{IN} - V_1)]}{[1 - d_1/(V_{IN} - V_1)]}\right\}$$

the equations reduce by means of the following assumptions:

1. $d_1 = d_2 = d$
2. $|V_{IN} - V_0| \gg d$
 $|V_{IN} - V_1| \gg d$

applying these assumptions, we get the following:

$$t_1 = -RC \ln[(1 + x)/(1 - x)] \text{ where } x = -d/(V_{IN} - V_0)$$

$$t_2 = -RC \ln[(1 + x)/(1 - y)] \text{ where } y = d/(V_{IN} - V_1)$$

because of the assumptions above, the x and y terms in the preceding equations are less than 1, therefore the following expansion can be used:

$$\ln[(1 + z)/(1 - z)] = 2[z + (z^3)/3 + (z^5)/5 + \dots]$$

substituting we have:

$$t1 = -2RC[x + (x**3)/3 + \dots]$$

$$t2 = -2RC[y + (y**3)/3 + \dots]$$

under assumption 2 above, the linear term completely swamps the exponential terms yielding the following result (after substituting back into the equation):

$$t1 = 2dRC/V_{IN} - V0 \quad t2 = -2dRC/(V_{IN} - V1)$$

therefore:

$$t1/(t1 + t2) = (V1 - V_{IN})/(V1 - V0)$$

$$t2/(t1 + t2) = (V_{IN} - V0)/(V1 - V0)$$

solving for V_{IN} :

$$V_{IN} = [t2/(t1 + t2)][V1 - V0] + V0$$

$$\text{or } V_{IN} = V1 - [t1/(t1 + t2)][V1 - V0]$$

It follows from the above results that by measuring the times $t1$ and $t2$, the input voltage can be accurately determined. As will be seen the restrictions based upon the assumptions above do not cause any serious difficulty.

General Accuracy Considerations

In the preceding calculations it was assumed that the differential output above and below the input voltage was the same. If the comparator output is checked at absolutely regular intervals, and if the intervals are kept as small as possible this assumption can be fairly easily guaranteed—at least to within the comparator offset which is only a few millivolts. As we shall see, this aspect of the technique presents few, if any, difficulties. In addition, there is an RC network at the input of the comparator. The time constant of this network must be long relative to the time between checks of the comparator output. This will insure that the capacitor voltage does not change very much between checks and thereby help to insure that the differences above and below the input voltage are the same.

The next major approximation has to do with the difference between the input voltage and either $V1$ or $V0$. We have relied on this difference being much greater than the amount the capacitor voltage changes above and below the input voltage. This approximation allows the nonlinear terms in the logarithmic expansion to be discarded. In practicality, the approximation means that the input voltage must not be "close" to either $V1$ or $V0$. Therefore, it becomes necessary to determine how closely the input voltage can approach $V1$ or $V0$. It is obvious that the smaller the difference d can be made, the closer the input voltage can approach either reference. The following calculations illustrate the method for determining that difference d . Note, using either $V1$ or $V0$ produces the same result. Thus $V = V1 = V0$.

For at least 1% accuracy

$$x + (x**3)/3 < 0.01x$$

$$\text{therefore } x < 0.173$$

$$\text{since } x = d/(V_{IN} - V) \text{ we have } d < 0.173(V_{IN} - V).$$

Using the same analysis for 0.1% accuracy in the approximation we get $d < 0.0548(V_{IN} - V)$. By applying this relationship, the RC time constant can be adjusted so that, within the time interval, the capacitor voltage does not change by more than d V. The user may then select, within

reason, how close to the references he can allow the input voltage to go.

The next consideration is really just one of simplification. It is clear that if $V0$ is zero, it drops out of the first equation and the relationship is simplified. Therefore, it is desirable to use zero volts as the $V0$ value. The equation then becomes:

$$V_{IN} = V1t2/(t1 + t2).$$

It is obvious by now that the heart of the technique lies in accurately measuring the times $t1$ and $t2$. Clearly this requires that the time base of the COP420 be accurate. Short term variations in the COP420 time base will clearly impact the accuracy of the result. In addition to that there is a serious problem in being able to check the comparator output often enough to get any accuracy and resolution out of simply measuring the times $t1$ and $t2$. This problem is circumvented by measuring many periods of the waveform. Doing this gives a large average, which improves the accuracy and tends to eliminate any spurious changes. Of course, the trade off is increased time to do the conversion. However if the time is available, the technique becomes restricted only by the accuracy of the external components. Those of the comparator and the reference voltage are most critical.

It is clear from the equation above that the accuracy of the result is directly dependent upon the accuracy of the reference voltage $V1$. In other words, it is not possible to be more accurate than the reference voltage. If, however, all that is required is a ratio between the input voltage and the reference voltage, the accuracy of the reference will not be a controlling factor provided that the input voltage tracks the reference. This requires that the input voltage be generated from the reference voltage in some form, e.g., a voltage divider with V_{IN} coming off a variable resistance.

Finally, we have noted that the difference d must be small. If the capacitor had to charge or discharge a long way toward V_{IN} , the nonlinearity of the capacitor charge curve would be significant. This therefore requires that the conversion begin with the capacitor voltage close to the input voltage.

Note that the RC value is not part of the equation. Therefore the accuracy of the time constant has no effect on the result as long as the time constant is long relative to the time between checks of the comparator output.

The final point is that the reference voltages, whatever they may be, must be hard sources. Should these voltages vary or drift at all, they will directly affect the result. In those configurations where the references are being switched in and out, the voltage should not change when it is switched into the circuit.

3.2 BASIC IMPLEMENTATION

General

The objective, then, is to measure the times $t1$ and $t2$. This is accomplished in the software by means of two counters. One of the two counters counts the $t2$ time; the other counter counts the total time $t1 + t2$.

It is necessary to check the comparator output at regular intervals. Thus the software must insure that path lengths

through the test and increment loops are equal in time. Further it is desirable to keep the time required to increment the counters as short as possible. A trade off usually comes into play here. The shortest loop in terms of code required to implement the function is rarely the shortest loop in terms of time required to execute the function. The user has to decide which implementation is best for him. The choice will frequently be governed by factors other than the A/D conversion limits.

It must be remembered that we are now dealing with analog signals. If significant accuracy is required, we are handling very small analog signals. This requires the user to take precautions that are normally required when working with linear circuits, e.g., power supply decoupling and bypassing, lead length restrictions, crosstalk, op amp and comparator stabilization and compensation, desired and undesired feedback, etc. As greater accuracy is sought these factors are more and more significant. It is suggested that the reader refer to the National Semiconductor Linear Applications Handbook and to the data sheets for the various components involved to see what specific precautions should be taken both in general and for a specific device.

The Base Circuit

Figure 7 shows the diagram for the basic circuit required to implement the duty cycle conversion scheme. The flow chart and code required to implement the function are shown in Figure 8. Note that the flow chart and code do not change—except for possible polarity change on output to allow for an inverting buffer—for any of the improvements in accuracy discussed later. The only exception to this is the technique illustrated in Figure 10 and the variations there are minor.

The code and flow chart in Figure 8 implement the technique as described above. The large averaging technique is

used as it would be too difficult to measure the times t_1 and t_2 in a single period. The total time, $t_1 + t_2$, is the viewing window under complete control of the software. This window is a time equal to the total number of counts, determined by desired accuracy, multiplied by the loop time for a single count. A second counter is counting the t_2 time. Special care is taken to insure that all paths through the code take the same length of time since the integrity of the time count is the essence of the technique. The full conversion scheme would use the subroutine in Figure 8. Normally the subroutine would be called first just to get the capacitor charged close to the input voltage. The result obtained here would be discarded. Then the routine would be called a second time and the result used as required.

In the configuration in Figure 7, there is an RC network in both input legs of the comparator. This is to balance the inputs of the device. For this reason, $R_1 = R_2$. C_1 is the capacitor whose voltage is being varied by the pulse waveform. C_2 is in the circuit only for stabilization and symmetry and is not significant in the result. The comparator tends to oscillate when the + and - inputs are nearly equal without capacitor C_2 in the circuit.

As would be expected, the basic circuit has some difficulties. By far the most serious of these difficulties is the output level of the G line. To be sure of the high and low level of this output the levels should be measured. The "1" level will be between the spec minimum of 2.4V and V_{CC} (here assumed to be 5V). The "0" level will be between the 0.4V spec maximum and ground. With light loads, these levels are likely to vary from device to device. Furthermore, we have the same "1" level problem that was mentioned in the simplest technique: the capacitive load is large and the capacitor is charging while the output is trying to go to the high level.

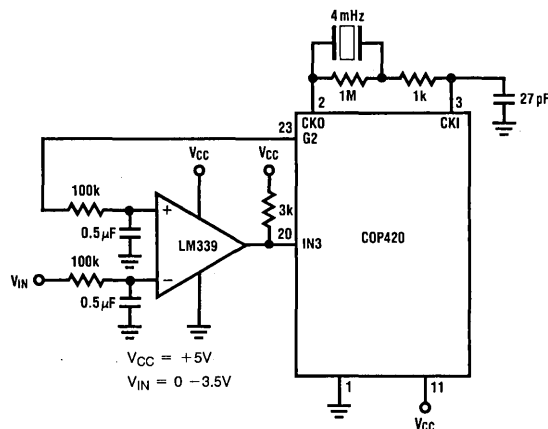


FIGURE 7. Basic Duty Cycle A/D

TL/DD/6935-11

There is also a problem with the low level. When the output goes low, the capacitor begins to discharge through the output device of the COP420. This discharge current has the effect of raising the "0" level and thereby introducing error. Note that we are not talking about large changes in the voltages, especially the low level. Typically, the change will only be a few millivolts but that can translate into a loss of accuracy of several bits.

Under laboratory conditions—holding all variables constant and using precise measured values in the calculations—the circuit of *Figure 7* yielded 5 bit \pm 1 bit accuracy over

the range of V_0 (here measured to be 0.028V) to 3.5V (the maximum specified input voltage for the comparator with $V_S = 5V$). Increasing the number of total counts had very little effect on the result. In the general case, the basic scheme should not be relied upon for more than 4 bits of accuracy, especially if one assumes that $V_1 = V_{CC}$ and $V_0 = 0$. As shall be seen, it is not difficult to improve this accuracy considerably.

```

; ATOD IS THE FULL CONVERSION SCHEME WRITTEN AS A SUBROUTINE
ATOD:  LBI    1, 10    ; MAKE SURE COUNTERS CLEARED
        JSRP   CLEAR
        LBI    2, 10
        JSRP   CLEAR
        LBI    1, 13    ; PRELOAD FOR TOTAL COUNT = 2048
        STII   0
        STII   0
        STII   8
ATOD1:  ININ    ; READ COMPARATOR--INPUT TO 420 = IN3
        AISC   8
        JP     SND01
SND1A:  LBI    3, 0    ; USING DMG BELOW TO SAVE STATE OF OTHER G
        ; VALUES IF IT WAS NECESSARY TO DO SO, ELSE USE DGI
        SMB    2    ; VIN > Vc, DRIVE Vc HIGHER
        DMG    ; THIS CODE STRAIGHT LINED FOR SPEED
        SC     ; APPLY POSITIVE REFERENCE
        CLRA   ; INCREMENT THE SUB COUNTER
        LBI    2, 13
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP    ; BINARY INCREMENT
        XIS    ; WOULD ELIMINATE THESE 4 WORDS IF 8 BIT
        CLRA   ; COUNTER OR LESS--HERE SET UP FOR UP TO 12 BIT
        ASC    ; COUNTER
        NOP
        X
        JP     TOTAL
SND01:  LBI    3, 0
        RMB    2
        DMG
        CLRA
        AISC   10    ; THIS PART OF THE CODE MERELY INSURES THAT
        NOP    ; ALL PATHS THROUGH THE ROUTINE ARE EQUAL IN TI
DI Y:   AISC   1
        JP     DLY
TOTAL:  CLRA
        LBI    1, 13
        SC     ; INCREMENT THE TOTAL LOOP COUNTER
        ASC    ; WHEN OVERFLOW, DONE SO EXIT
        NOP
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        JP     ATOD2
        RET
ATOD2:  X
        JP     ATOD1
        .PAGE  2
CLEAR:  CLRA
        XIS
        JP     CLEAR
        RET

```

TL/DD/6935-45

FIGURE 8A. Duty Cycle A/D Code

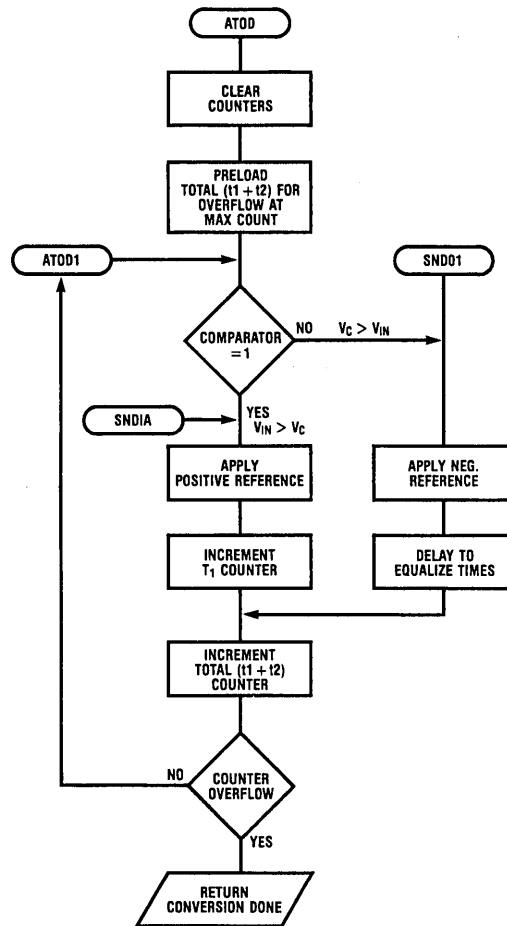


FIGURE 8B. Duty Cycle A/D Flow Chart

TL/DD/6935-12

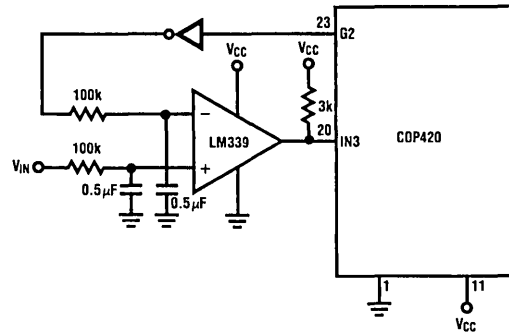
3.3 ACCURACY IMPROVEMENTS

General Improvements

Figure 9 illustrates circuit changes that will make significant improvements in the accuracy of the technique. In Figure 9A a CMOS buffer is used to drive the RC network. The output of the COP420 drives the CMOS gate, which here is a 74C04 because of its output characteristics. The main thing that this technique does is to reduce the difficulties with the output levels. Typically, V0 is 0V and V1 is VCC. We also have a "harder" source for the voltages — the levels don't change while the capacitor is charging or discharging. Now, even more clearly than before, the accuracy of VCC is the controlling voltage tolerance. The accuracy of the result will be no better than the accuracy of VCC (for a system requiring absolute accuracy).

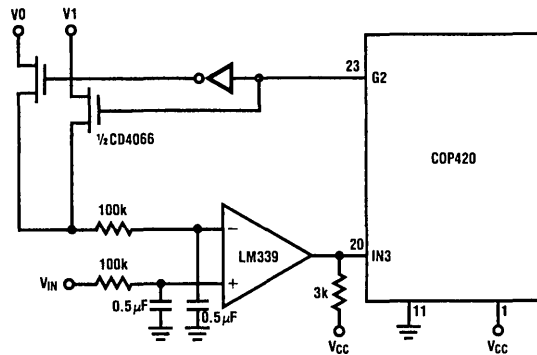
Under laboratory conditions, the circuit of Figure 9A yielded the accuracies as indicated below for various total counts. The accuracy increased with the total count until the count exceeded 2048. There was no significant increase in accuracy with this circuit for counts in excess of 2048. (Remember that these results were obtained under controlled conditions). We may then view the results obtained with 2048 counts as the upper limit of accuracy with the circuits of Figure 9A. The results were as follows:

Total Count	Resultant Accuracy
512	8 ± 1/2 bits
1024	9 ± 1bits
2048	9 ± 1/2 bits
4096	9 ± 1/2 bits



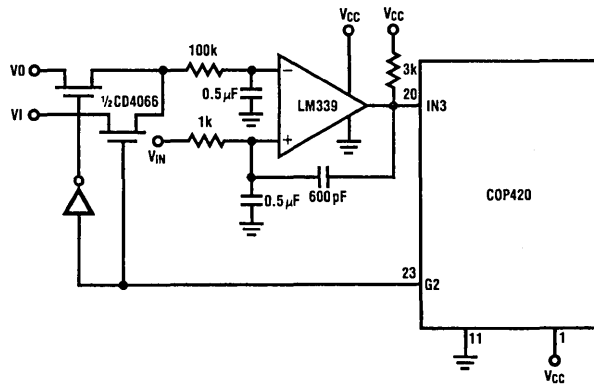
TL/DD/6935-13

A



TL/DD/6935-14

B



TL/DD/6935-15

C

FIGURE 9. Improvements to Duty Cycle A/D

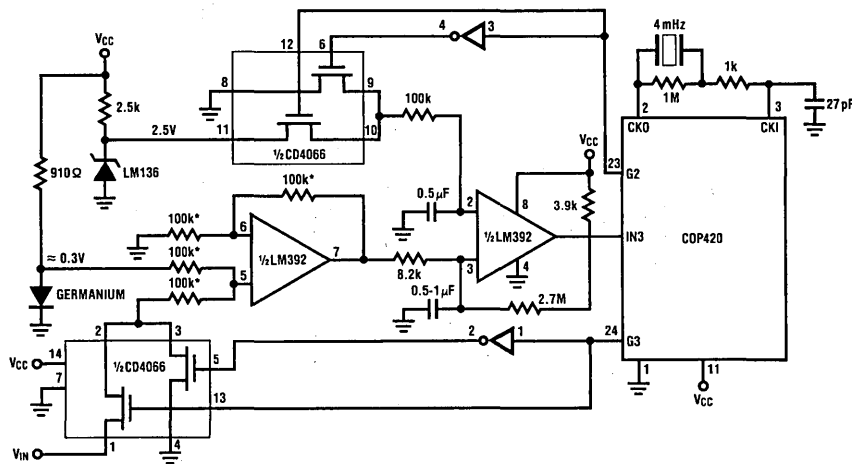
The circuit of *Figure 9B* makes a significant change to improve accuracy. Now the COP420 is controlling analog switches and switching in positive and negative references. Therefore the accuracy of the reference voltages is the controlling factor. Generally this will improve the accuracy over that obtained with *Figure 9A*. With the circuit of *Figure 9B*, with $V_0 = 1V$ (negative reference), and $V_1 = 3V$ (positive reference), 9 bit accuracy was achieved with a total count of 1024. V_0 and V_1 were arbitrarily chosen to place the input voltage approximately in the center of the allowable comparator input range with $V_S = 5V$. Remember, the accuracy of the references is controlling. The result can be no more accurate than the references. Furthermore, these references must be hard sources; i.e., they must not change when they are switched into the circuit as that contributes error into the result.

In *Figure 9C*, capacitive feedback was added to the comparator circuit and the series resistance to V_{IN} was decreased. The feedback added hysteresis and forced the comparator to slew at its maximum rate (significant errors are introduced if the comparator does not change state in a time shorter than the cycle time of the controller). Both of these changes resulted in increased accuracy of the result. With $V_0 = 0$, $V_1 = 5V$ (V_{CC}) and V_{CC} held steady at 5.000V, an accuracy of 10 bits ± 1 bit was achieved over the input range of 0 to 3.5V.

It is obviously possible to use any combination of the configurations in *Figure 9* for a given application. What is used will depend on the user and his specific requirements.

Figure 10 illustrates a further refinement of the basic approach. This configuration can be used if greater accuracies are needed. The major change is the addition of a summing amplifier to the circuit for the purpose of adding a fixed offset voltage to the input voltage. This has the effect of moving the input voltage away from the negative reference (which is 0V here). This offset voltage should be stable as the changes in it will directly affect the result. The offset voltage should be chosen so as to place the effective input voltage (the voltage at the comparator input) approximately in the center of the range between the two references. The precise value of the offset is not critical nor is its source. The forward voltage drop across a germanium diode is used as the offset in *Figure 10*, but this offset can be generated in any convenient manner. The forward voltage drop of the germanium diode is approximately 0.3V. Given this and the negative reference of 0V and a positive reference of 2.5V, the input voltage is restricted to a range of 0 to 2V. Therefore, the effective input voltage (at the comparator input) is approximately 0.3V to 2.3V — well within the limits of the two references. The circuit also includes provision for an autozero self calibration procedure.

Note that the resistors in the summing amplifier should be matched. The absolute accuracy of these resistors is not significant, but their accuracy relative to one another can have a significant bearing on the result. The restriction is imposed so that the output of the summing amplifier is exactly the sum of the input voltage and the offset voltage. This requires unity gain through the amplifier and that the



*Resistors should be matched

$V_{CC} = +5V$
 $0 \leq V_{IN} \leq 2V$

FIGURE 10. Improved Duty Cycle A/D with Autozero

TL/DD/6935-16

impedance in each summing leg be the same. These effects can become very serious if one is trying for significant accuracy—e.g., if 12 bit accuracy is being sought 1% matching of those resistors can introduce an error of 1% maximum. While 1% accurate is fairly good, it is significantly less than 12 bit accuracy. Related to this effect is a possible problem with the source impedance of the input voltage. If that impedance is significant in terms of its ratio to the summing resistor, errors are introduced just as if the resistors are mismatched. "Significant" is determined in terms of the desired system accuracy and the relative impedance values. The comparator section is using some feedback to provide hysteresis for stability and a low series resistance is used for the input to the comparator.

Most significantly, this configuration allows a true zeroing of the system. Through the additional analog switches shown, the COP420 can easily perform an autozero function by

tying the input to ground and measuring the result. Thus the system offsets can be calculated, stored and subtracted from the result. This improves the accuracy and is also more forgiving on the choice of the comparator and op amp selected. Furthermore, the offset can be periodically recomputed by the COP420 thereby compensating for drift in system offsets. Nonetheless, the accuracy of the reference is the controlling factor. It is NOT possible to obtain an absolute (as opposed to ratiometric) accuracy of 12 bits without a reference that is accurate to 12 bits. The LM136 used in *Figure 10* is a 1% reference. Although not inherently accurate to 12 bits, the voltage of the LM136 may be trimmed to exact value by means of a variable resistor. The data sheet of the LM136 illustrates this connection. Under laboratory conditions, the circuit of *Figure 1* yielded 11 bit ± 1 bit accuracy with a total count of 4096 over the input range of 0 to 2V. *Figure 11* indicates the flow chart and the code required to implement the technique of *Figure 10*.

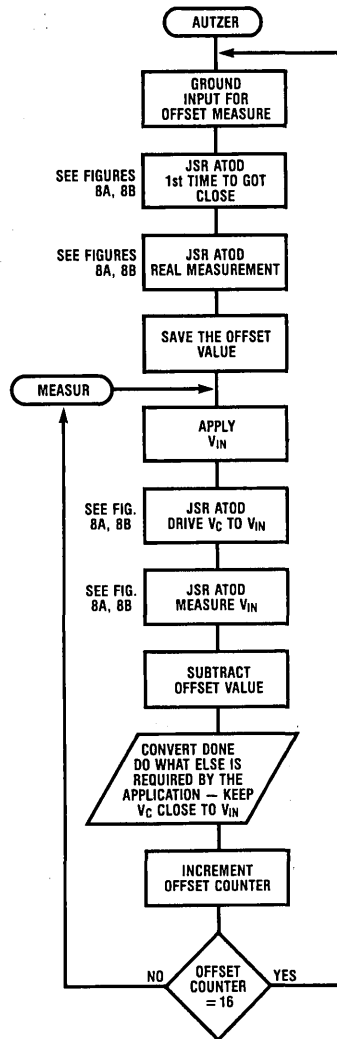
```

; CODE FOR IMPROVED A TO D PULSE WIDTH METHOD
; SEE FIGURE 8A FOR CODE FOR ROUTINE ATOD
;
AUTZER: LBI    3, 0    ; DO AUTO ZERO, 3, 0 CONTAINS 0 STATUS
        RMB    3    ; SET UP TO GRND INPUT & MEASURE OFFSET
        JSR   ATOD   ; FIRST TIME IS TO GET CLOSE
        JSR   ATOD   ; MEASURE THE OFFSET
XIFR:   LBI    2, 13   ; NOW SAVE THE OFFSET VOLTAGE
        LD    1    ; SAVE THE OFFSET VALUE IN M3
        XIS    1
        JP    XFER
        LBI    0, 0
        JP    INPUT
MFMASUR: ; NOW DO REAL MEASUR (1ST TIME IS OFFSET)
        JSR   ATOD   ; FIRST TIME TO GET CLOSE
        JSR   ATOD   ; NOW REAL MEASUREMENT
        JSRP  BINSUB ; SUBTRACT THE OFFSET
; HAVE THE VALUE AT THIS POINT (IN BINARY)—NOW DO WHAT
; THE APPLICATION REQUIRES. VALUE MUST BE MULTIPLIED
; BY (VREF+/TOTAL COUNT) TO GET FINAL VALUE IF SUCH IS
; DESIRED
        LBI    1, 0    ; INCREMENT COUNTER FOR NEW OFFSET MEASURE
        LD    1
        AISC   1
        JP    SAVE
        X     ; IS 16TH TIME, MEASURE OFFSET AGAIN
        JP    AUTZER
SAVI:   X
        LBI    3, 0
        SMB    3    ; SET BIT SO CAN MEASURE VIN
        JP    MEASUR
        .PAGE  2
BINSUB: LBI    3, 13
        SC
BNSUB?: LD    1
        CASC
        NOP
        XIS    1
        JP    BNSUB2
        RET

```

FIGURE 11A. Duty Cycle A to D, Improved Method

TL/DD/6935-46



TL/DD/6935-17

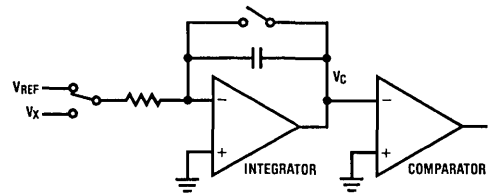
FIGURE 11B. Flow Chart for Improved Duty Cycle A/D

4.0 Dual Slope Integration Techniques

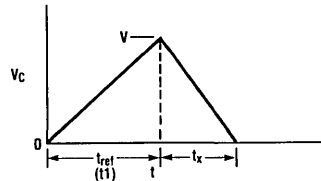
4.1 Mathematical Background

(Some of this background information is taken from National Semiconductor Linear Applications Note AN-155. The reader is referred to that document for other related general information.)

The basic approach of dual slope integration conversion techniques is to integrate a voltage across a capacitor for a fixed time, and then to integrate in the other direction with a known voltage until the starting point is reached. The ratio of the two times then represents the unknown voltage. Some of the math below in conjunction with Figure 12 will illustrate the approach.



TL/DD/6935-18



TL/DD/6935-19

FIGURE 12. Dual Slope Integration—Basic Concept

$$I_X = C \frac{dV}{dt} = V_X/R$$

$$V_X = RC \frac{dV}{dt}$$

$$\int_0^{T_1} V_X dt = \int_0^V RC dV$$

$$V_X T_1 = RC V$$

$$V = V_X T_1 / RC = I_X T_1 / C$$

Similarly:

$$I_{REF} = C \frac{dV}{dt} = V_{REF}/R$$

$$V_{REF} = RC \frac{dV}{dt}$$

$$\int_{T_1}^{T_1 + T_X} V_{REF} dt = \int_V^0 RC dV$$

$$V_{REF} T_X = -RC V$$

$$V = -V_{REF} T_X / RC$$

$$-V_{REF} T_X / RC = V_X T_1 / RC$$

$$V_X = -V_{REF} T_X / T_1$$

Two important facts arise from the preceding mathematics. First of all, there is a linear relationship involved in determining the unknown voltage. Secondly, the negative sign in the final equation indicates that the reference and the unknown, relative to some point (which may be 0V or some bias voltage), have opposite polarity. Thus, if it is desired to measure 0 to +5V, the reference voltage must be -5V. If the input is restricted to 2.5 to 5V, the reference can be 0V as the integrator and comparator are biased at +2.5V (then the 0V is in fact -2.5V relative to the biasing voltage, and the input range is 0 to 2.5V relative to the same bias voltage).

There are some difficulties with dual polarity conversion using the dual slope method. It is clear from the math above that if the input voltage will be dual polarity, it is necessary to have two references—one of each polarity. The midrange biasing arrangement briefly described above eliminates

the need for two different polarities but does not help very much since two references are still required—one at the positive value and one at the bias value. Ground is the other reference. Further, the need to select one of two references further complicates the circuitry involved to implement the approach. Also, the dual requirement brings up a difficulty with the bias currents of the integrator and comparator. They could add to the slope in one polarity and subtract in the other.

The only real operational difficulty in dual slope systems is establishing the initial conditions on the integrating capacitor. If this capacitor is not at the proper initial conditions, accuracy will be severely impaired. Figure 12 indicates a switch across the capacitor as a means of initializing it. In a software driven system, the initialization can be accomplished by doing two successive conversions. The result of the first conversion is discarded. It is performed only to initialize the capacitor. The second conversion produces the valid result. One need only insure that there is not significant time lapse between the two conversions. They should take place immediately after one another.

This approach obviously lengthens conversion time but it eliminates many problems. The alternative to this approach of two successive conversions is to take a great deal of care in insuring the initial state of the integrating capacitor and in selecting op amps and comparators with low offsets.

4.2 THE BASIC DUAL SLOPE TECHNIQUE

Figure 13 indicates an implementation of the basic dual slope technique. This is a single polarity system and thus requires only the single reference voltage. The circuit of Figure 13 is perhaps not the cheapest way to implement such a scheme but it is representative and illustrates the factors that must be considered.

Consider first the means of initializing the integrating capacitor C1. The routine here connects the input to ground and does a conversion on zero volts as a means of initialization. Subsequently—and this is typical of the more usual technique—two conversions are performed. The first conversion is to initialize the capacitor. The second conversion yields the result. Some form of initialization or calibration procedure is required to achieve optimum accuracy from dual slope conversion schemes.

The comparator in this circuit is used in the inverting mode and has positive feedback as recommended in the LM111 data sheet. The voltage reference is the LH0070, which is a 0.01% reference. A resistive voltage divider on the LH0070 creates the 5V value. The use of the voltage divider brings up two difficulties (which can be overcome if the LH0070 is used at its full value, thus eliminating the divider, and the result properly scaled in the microcontroller or series integrating resistor increased). First, the impedance of the reference must be small relative to the series resistance used in the integrator. If this were not the case, the slopes would

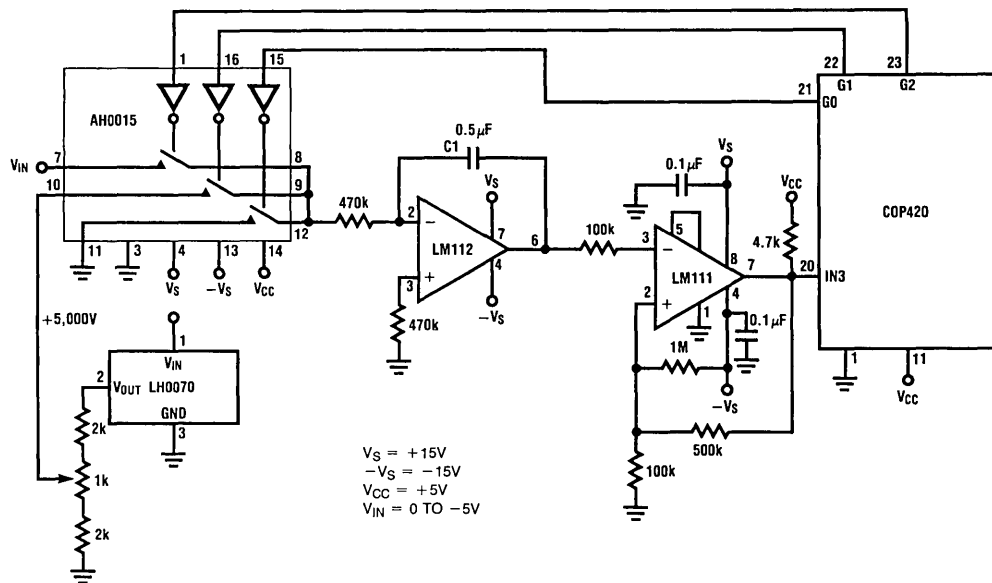


FIGURE 13. Basic Dual Slope Integration A/D Scheme

TL/DD/6935-20

show an effect due to the difference in the R value between the applied reference voltage and the unknown input. (By the same token, the output impedance of the source supplying the unknown must also be small relative to that series integrating resistor). Secondly, the bias currents of the integrator may be such as to affect the reference voltage when it is coming from a simple resistor divider. Both problems are reduced if small resistor values are used in the divider. Note also that current mode switching would reduce the problem as well. It should be pointed out that the errors introduced by these problems are not gross deviations from the expected value. They are small errors that will not make much difference in the majority of applications. They are, however the kind of errors that can make the difference between a system accurate to 10 bits and one accurate to 12 bits (assuming all other factors are the same).

Figure 14 shows the flow chart and code required to implement the basic dual slope technique as shown in Figure 13. Under laboratory conditions an accuracy of 12 bits ± 1 bit was achieved. The method is slow, with the maximum conversion time equal to $2 \times T_{REF}$. Notice that the accuracy of V_{CC} and that of the integrating resistor and capacitor are not involved in the accuracy of the result. The accuracy of V_{REF} is, of course, controlling if absolute accuracy—rather than ratiometric accuracy—is desired. The absolute accuracy of the circuit can be no better than the accuracy of the reference. If ratiometric accuracy is all that is required, there is no particular problem. The accuracy is merely relative to the reference. The R and C values do not impact the accuracy because the integration in both directions is being done through the same R and C. Results would be quite different if a different value of R or C was used for one of the slopes.

```

DUALP:  OGI      1      ; HOLD THE INPUT TO GROUND TO RESET THE
        LBI      2, 11   ; INTEGRATING CAPACITOR
        JSRP     CLEAR   ; CLEAR THE COUNTER
        JSR      INCRA   ; TO GET US CLOSE, NEXT READING IS REAL
CIFAAP:  LBI      2, 11   ; NOW CLEAR THE COUNTER
        JSRP     CLEAR   ; MAKE SURE COUNTER CLEARED TO ZERO
        ; 1, 15 = 0 AND START AT 1, 13 FOR COUNT = 4096
        ; 1, 15 = 14 AND START AT 1, 12 FOR COUNT = 8192
        ; 1, 15 = 12 AND START AT 1, 12 FOR COUNT = 16384
        ; FOLLOW SAME PATTERN FOR OTHER COUNTS

M-ACUR:  JSR      INCRA   ; RUN THRU THE INCREMENTS
        ; NOW HAVE THE BINARY VALUE, USE IT AS IS OR
        ; MULTIPLY BY (Vref/TOTAL COUNT) TO CREATE THE VOLTAGE
        ; RESULT--THEN CONTINUE WITH THE OPERATION
        LBI      2, 11
        JSRP     CLEAR   ; CLEAR THE COUNTER
        JSR      INCRA   ; TO GET CAP CLOSE TO 0 AGAIN
        JP       CLEAR2

; FOLLOWING SUBROUTINE INCRA IS THE REAL PART OF THE ROUTINE
; CONCERNED WITH THE COUNTING FOR THE CONVERSION.
INCRA:   LBI      1, 15   ; R1 IS CLEARED PRIOR TO START
        STII     15      ; PRESET THE COUNTER FOR 4096
        OGI      4       ; APPLY VIN
INCR:    LBI      1, 12
        SC

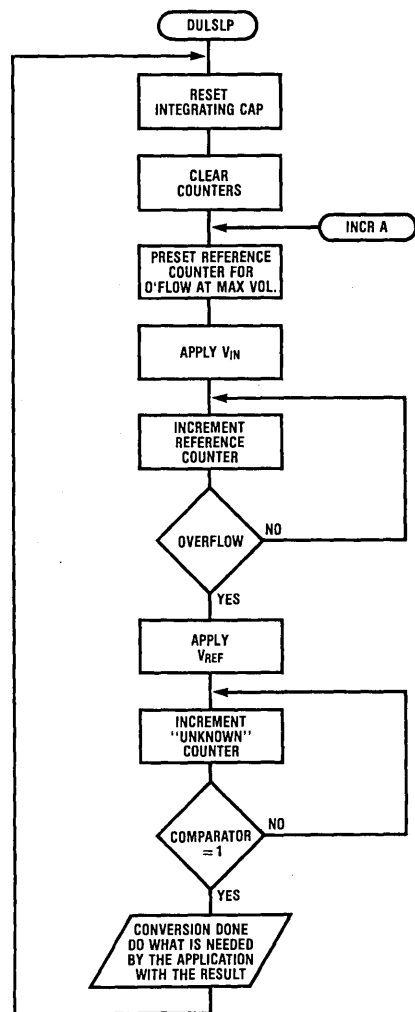
BINAD1:  CLRA
        ASC
        NOP
        XIS
        JP       BINAD1
        NOP      ; 2 NOPS TO EQUALIZE TIMES
        NOP
        SKC
        JP       INCR
        OGI      2       ; DONE, NOW APPLY VREF
INCRG:   LBI      2, 12   ; COUNT UNTIL COMPARATOR CHANGES
        SC

BINAD2:  CLRA
        ASC
        NOP
        XIS
        JP       BINAD2 ; STRAIGHT LINE THE ADD FOR SPEED
        ININ     ; SAVE WORDS BY USING G
        AISC     8       ; SEE IF IN3=1
        JP       INCR2  ; IN3 IS 0, KEEP COUNTING
OUTPUT1: OGI      1       ; KEEP INPUT AT 0
        RET

```

FIGURE 14A. Dual Slope A/D Code

TL/DD/6935-47



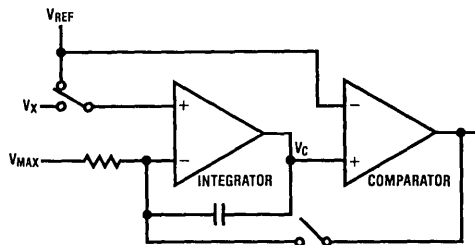
TL/DD/6935-21

FIGURE 14B. Basic Dual Slope A/D Flow Chart

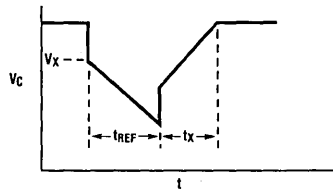
4.3 MODIFIED DUAL SLOPE TECHNIQUE

General

The basic idea of the modified dual slope technique is the same as that of the basic approach. The modified approach eliminates the need for dual polarity references and is also more forgiving in the selection of the op amp and comparator required. Figure 15 illustrates the basic idea.



TL/DD/6935-22



TL/DD/6935-23

FIGURE 15. Modified Dual Slope — Basic Concept

The math analysis is much the same:

$$I_X = C \frac{dV}{dt} = (V_X - V_{MAX})/R$$

$$V_X - V_{MAX} = RC \frac{dV}{dt}$$

$$(V_X - V_{MAX})T_1 = RC$$

$$V = (V_X - V_{MAX})T_1/RC$$

Similarly:

$$I_{REF} = C \frac{dV}{dt} = (V_{REF} - V_{MAX})/R$$

$$(V_{REF} - V_{MAX})T_X = -VRC$$

$$V = -(V_{REF} - V_{MAX})T_X/RC$$

$$(V_{MAX} - V_{REF})T_X = (V_X - V_{MAX})T_1$$

$$V_X = V_{MAX} + (V_{MAX} - V_{REF})T_X/T_1$$

The main difference between this and the basic approach is the offset voltage V_{MAX} . The main restriction is that all input voltage values (V_X) are less than V_{MAX} . It is also apparent that the total count is proportional to the difference between V_{MAX} and V_X . The only significant effect of this is, however, to slightly complicate the arithmetic required to arrive at a value for V_X .

Given that the input voltage V_X is always less than V_{MAX} , the modified dual slope technique is automatic polarity. This fact comes straight out of the equation above. Thus dual polarity references are not required. However, two precise voltages are required: V_{MAX} and V_{REF} . However, the V_{MAX} value can be used for a zero adjust as indicated in Figure 16. This means that the V_{MAX} value need not be so precise as it will be adjusted in a calibration procedure to produce a zero output. This adjustment amounts to a compensation for the bias currents and offsets. Thus the COP420 can use the supposed value of V_{MAX} with V_{MAX} later being "tweaked" to give the proper result at zero input. In addition, the initialization loop for the integrating capacitor includes the comparator. Thus the initial condition on the capacitor becomes

not zero but the sum of the offset voltages of the comparator and op amp. Thus the choice of these components is not critical in a modified dual slope approach.

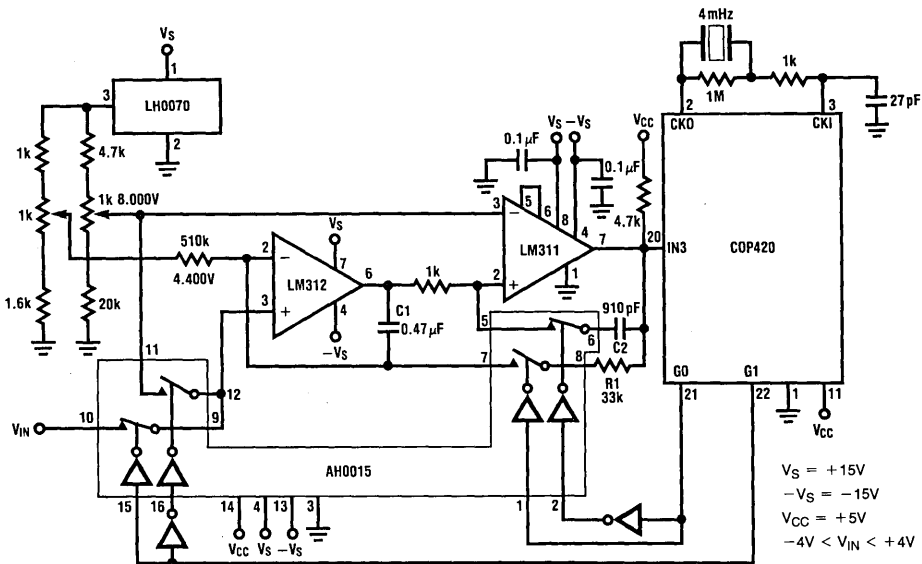
An Example of the Modified Dual Slope Approach

Figure 16 illustrates an implementation of the modified dual slope technique. The system is calibrated by holding V_{IN} to ground and then adjusting V_{MAX} for a "0" result. Capacitor C1 is the integrating capacitor. Capacitor C2 is used only to cause a rapid transition on the comparator output. C2 is especially useful if an op amp is being used as the comparator stage. Resistor R1 is just part of the capacitor initializing loop. An LH0070 is being used to generate the reference voltage and the V_{MAX} value. The discussion previously about these being hard sources is equally relevant here. In fact, this problem was much more significant in this particular implementation and made the difference between a 10 and 12 bit system. As shown, the technique was accurate to 10 bits. Another bit was obtained when the V_{MAX} and V_{REF} values were buffered. It must be remembered that when trying to achieve accuracies of this magnitude board layout, parts placement, lead length, etc. become significant factors that must be specifically addressed by the user.

There are some other considerations in using this technique. The amount of time required to count the specified number of counts starts to become a significant factor. If it takes "too long" to do the counting, the capacitor can charge to either supply voltage depending on which direc-

tion it is integrating. This causes the wave shape shown in Figure 15 to flatten out. This effectively limits the input range for all accuracy is lost once that waveform flattens out. In fact, this was the limiting factor on the accuracy in Figure 16 as shown. Given the amount of time required for an increment of the counter for T_{REF} (or T_X), it was not possible to reach the 4096 counts required for 12 bit accuracy before the waveform flattened out. Decreasing the total count solves the problem at the expense of accuracy. It is therefore desirable to keep the loop time required for an increment as fast as possible. The code to implement Figure 16 is shown in Figure 17 and reflects that concern. The other way to solve the problem is to use a large value for R and C. This is the easiest solution and preserves accuracy. Its cost is increased conversion time.

Both the basic and modified dual slope schemes can be very accurate and are commonly used. They tend to be relatively slow. In many applications, however, speed is not a factor and these approaches can serve very well. There are various approaches to dual slope analog to digital conversion which try to improve speed and/or accuracy. These are usually multiple ramping schemes of one form or another. The heart of the approach is the basic scheme described above. It is not the purpose here to delve into all the possible ways that dual slope conversion may be accomplished. The control software is not significantly different regardless of which particular variation is used. The basic ramping control is the same as that indicated here.



TL/DD/6935-24

FIGURE 16. Modified Dual Slope Integration

The number of components required to implement a dual slope scheme is not related to the desired accuracy. The approach is generally tolerant as to the op amps and comparators used as long as proper care is given to the initialization of the integrating capacitor.

Precise references are not required if a ratiometric system is all that is required. Cheaper switches can be safely used. The dual slope scheme controlled by a COPS microcontroller can be a very cost effective solution to an analog to digital conversion problem.

```

CIRCAP:  DGI      1      ; APPLY VREF AND ENABLE RESET PATH
CIFAR?:  LBI     2,11   ; NOW CLEAR THE COUNTER
          JSRP     CLEAR
          ; J, 15=15, 1, 14=4 AND START AT 1, 12 FOR COUNT = 3072
          ; J, 15 =15 AND START AT 1, 12 FOR COUNT = 4096
          ; J, 15 = 14 AND START AT 1, 12 FOR COUNT = 8192
          ; J, 15 = 12 AND START AT 1, 12 FOR COUNT = 16384
          ; FOLLOW SAME PATTERN FOR OTHER COUNTS
          ;
MFAWR:   JSR     INCRA   ; RUN THRU THE INCREMENTS
          ; HAVE THE VALUE AT THIS POINT. DO WHAT THE APPLICATION
          ; REQUIRES--REMEMBER, TO CREATE REAL VALUE MUST MULTIPLY
          ; RESULT BY (VREF-VMAX)/TOTAL COUNT AND THEN SUBTRACT
          ; THAT RESULT FROM VMAX--DO IT IN DECIMAL OR BINARY, WHICHEVER
          ; IS BEST FOR THE APPLICATION
          LBI     1, 11   ; MAKE SURE SPACE IS CLEARED
          JSRP     CLEAR
          LBI     2, 11
          JSRP     CLEAR
          JSR     INCRB   ; FOR TEST-KEEP IT CLOSE
          LBI     1, 11   ; MAKE SURE COUNTER IS CLEARED
          JSRP     CLEAR
          JP      CLEAR2
INCR1:   LBI     1, 14
          STII    4      ; PRESET HERE FOR SMALLER COUNT
          STII    15     ; PRESET THE COUNTER FOR 4096
INCR1:   DGI     2      ; APPLY VIN AND ENABLE FEEDBACK
INCR:    LBI     1, 12
          SC
BINAD1:  CLRA
          ASC
          NOP
          XIS
          JP      BINAD1
          NOP        ; 2 NOPS TO EQUALIZE TIMES
          NOP
          SKC
          JP      INCR
          DGI     0      ; DONE, NOW APPLY VREF
INCR?:   LBI     2, 12   ; COUNT UNTIL COMPARATOR CHANGES
          SC
BINAD2:  CLRA
          ASC
          NOP
          XIS
          JP      BINAD2 ; STRAIGHT LINE THE ADD FOR SPEED
          ; SAVE WORDS BY USING G
          ININ    8      ; SEE IF IN3=1
          AISC    8
          JP      INCR2  ; IN1 IS 0, KEEP COUNTING
OUTPUT:  DGI     1      ; CLEAR THE CAPACITOR, APPLY VREF
          RET
INCRB:   LBI     1, 14   ; MAKE THE PASS FOR CAP INIT SHORT
          STII    7
          STII    15
          JP      INCR1

```

TL/DD/6935-48

FIGURE 17A. Modified Dual Slope Code

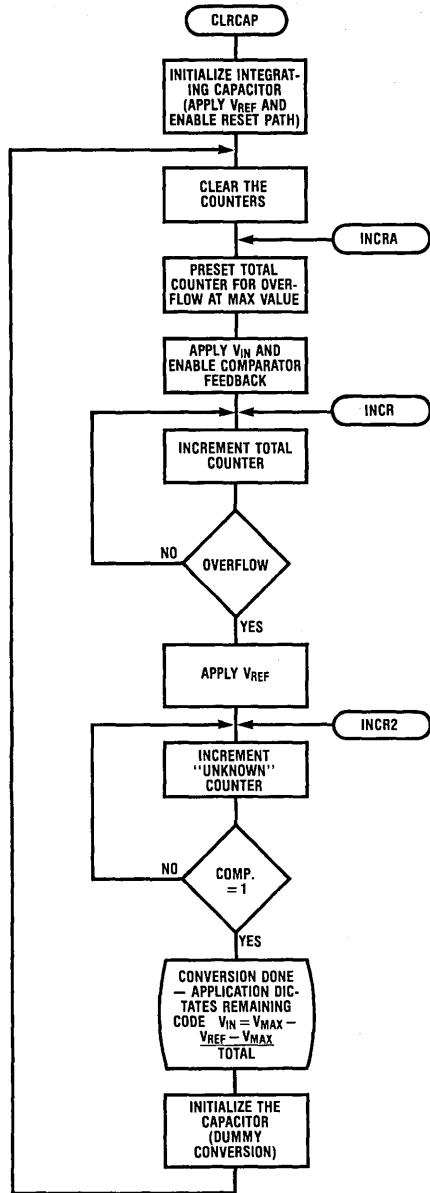


FIGURE 17B. Modified Dual Slope Flow Chart

TL/DD/6935-25

5.0 Voltage to Frequency Converters, VCO's

5.1 BASIC APPROACH

The basic idea of this scheme is simply to use the COP420 to measure the frequency output of a voltage to frequency converter or VCO. This frequency is in direct relation to the input voltage by the very nature of such devices. There are really only two limiting factors involved. First of all, the maximum frequency that can be measured is defined in the microcontroller by the amount of time required to test an input and increment a counter of the proper length. With the COP420 this upper limit is typically 10 to 15 kHz. The other limiting factor is simply the accuracy of the voltage to frequency converter or VCO. This accuracy will obviously affect the accuracy of the result.

Two basic implementations are possible and their code implementation is not significantly different. First, the number of pulses that occur within a given time period may be counted. This is straightforward and fairly simple to implement. The crucial factor is how long that given time period should be. To get the maximum accuracy from this implementation the time period should be one second. Such a time period would allow the distinction between the frequencies of 5000 Hz and 5001 Hz for example (assuming the V to F converter was that accurate or precise). Decreasing the amount of time will decrease the precision of the result. The alternate approach is to measure (by means of a counter) the amount of time between two successive pulses. This period measurement is only slightly more complicated than the pulse counting approach. The approach also makes it possible to do averaging of the measurement during conversion. This will smooth out any changes and add stability to the result. The time measurement technique is also faster than the pulse counting approach. Its accuracy is governed by how finely the time periods can be measured. The greater the count that can be achieved at the fastest input frequency — shortest period — the more accurate the result.

Figure 18 illustrates the basic concept. Figure 19A shows the flow charts and code implementation for both of the approaches discussed above. Note that whatever type of V to F converter is used, the code illustrated in Figure 19A is not significantly changed. In the code of Figure 19A, the interrupt is being used to test an input and thereby decreases the total time loop.

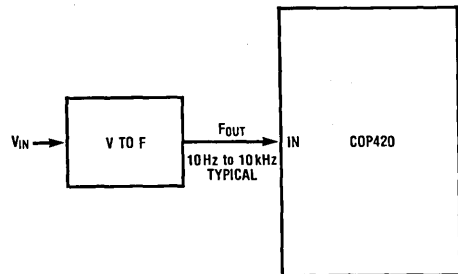


FIGURE 18. V to F Converter — Basic Concept

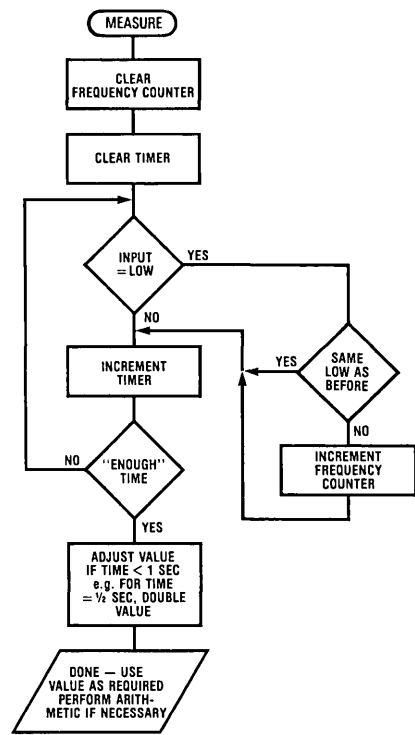
TL/DD/6935-26

```

MEASURE:          ; MEASURE BY COUNTING PULSES OF V TO F
;
LEI 2             ; ENABLE INTERRUPT
LBI 1,14         ; PRESET TIME FOR 122 COUNTS
STII 3           ; APPROX ONE HALF SECOND
STII 8
TIM: SKT         ; USE INTERNAL TIMER TO FIND
      JP TIME    ; THE 1/2 SECOND
BINP1: LBI 1,14 ; HAVE GOT IT, INCREMENT COUNTER
;
BINADD: CLRA
        ASC
        NOP
        XIS
        JP BINADD
        SKC           ; NOW SEE IF DONE
        JP TIME      ; NO COUNTER OVERFLOW, CONTINUE
        LEI 0         ; DONE, DISABLE INTERRUPT
FIN:   ; AT THIS POINT HAVE THE VALUE--CONVERT IT TO DECIMAL OR
        ; SEND IT OUT OR PROCESS IT FURTHER, WHATEVER IS REQUIRED
        ; BY THE APPLICATION. ARITHMETIC IS REQUIRED TO CREATE THE
        ; VOLTAGE VALUE, USUALLY A SIMPLE MULTIPLY
        ; MAY HAVE TO DOUBLE THE RESULT TO COMPENSATE LOOKING FOR
        ; ONLY 1/2 SECOND IN THIS CASE
;
        JP MEASURE   ; DO IT OVER AGAIN
; =X'OFF
ININT: NOP       ; SET ADDRESS TO OFF FOR INTERRUPT
ININT: LBI 2,12  ; ADDRESS OFF MUST BE NOP FOR INTERRUPT
ININT: SC        ; DO ADD OF THE VALUE FOR FREQ CNT
;
ININT: CLRA      ; STRAIGHT LINE THE CODE FOR SPEED
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        X
        LEI 2    ; ENABLE THE INTERRUPT AGAIN
        RET
    
```

TL/DD/6935-49

FIGURE 19A. V to F by Counting Pulses



TL/DD/6935-27

FIGURE 19B. V to F by Counting Pulses

```

; USE INTERRUPT FOR CATCHING THE PULSE EDGE
V-PR:  LBI 0,12  ; CLEAR COUNTER SPACE AND FLAG
        STII 0
        STII 0
        STII 0
        STII 0
        LBI 0,12
        LEI 2    ; NOW ENABLE THE INTERRUPT
        SC      ; DUMMY WAIT LOOP, WAITING FOR SIGNAL TO
        LBI 0,12 ; INTERRUPT THE CONTROLLER
        JP WAIT
; =X'OFF
ININT:  NOP     ; SET ADDRESS TO OFF--INTERRUPT ENTRY POINT
        LBI 0,12 ; REQUIRED FOR INTERRUPT ENTRY
        C(AN): LBI 0,12 ; NOW CHECKING TO SEE IF SECOND INTERRUPT
        SKMBZ 0   ; I. E., ARE WE DONE?
        JP DONE
        SMB 0    ; SET BIT FOR NEXT INTERRUPT
        LEI 2    ; ENABLE INTERRUPT AGAIN
PIUS1:  LBI 0,13 ; NOW START COUNTING
        SC
        CLRA    ; STRAIGHT LINE THE CODE FOR SPEED
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        X
        JP PLUS1
; FINISHED WHEN GET HERE--THE COUNT REPRESENTS THE PERIOD
; WITH ABOVE CODE, THE ACTUAL PERIOD IS THE COUNT MULTIPLIED
; BY 15 (THE NUMBER OF WORDS TO INCREMENT BY 1) PLUS AN OVERHEAD
; OF 9 CYCLE TIMES = 24 CYCLE TIMES. AT 4us THIS IS 96 us
; OR A FREQUENCY OF JUST OVER 10KHz. MAX COUNT HERE IS 4095.
; THIS GIVES A MAXIMUM PERIOD = 61434 CYCLE TIMES (=245.736ms AT
; 4us). THIS CORRESPONDS TO A FREQUENCY OF JUST OVER 4Hz
; NOTE, THIS IS 12 BIT RESOLUTION
    
```

TL/DD/6935-50

FIGURE 19C. A to D with VF Converter/VCO by Measuring Period

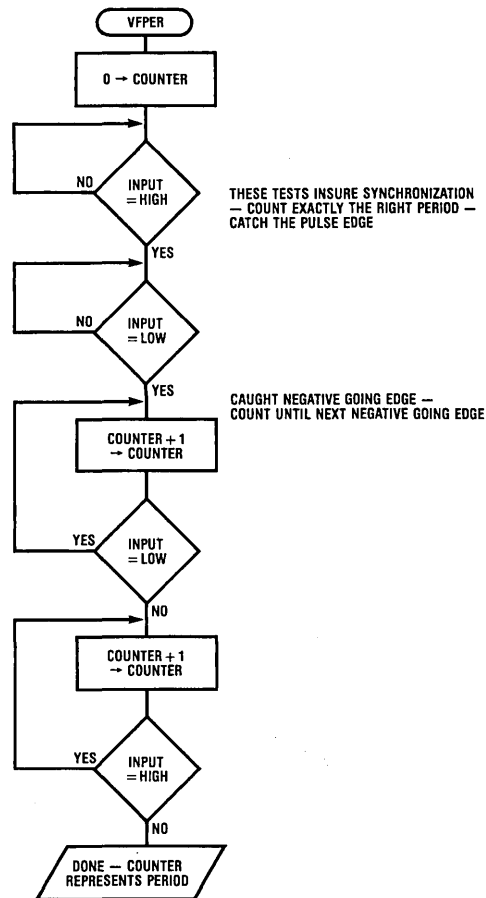


FIGURE 19D. V to F—Measure Period

5.2 THE LM131/LM231/LM331

The LM131 is a standard product voltage to frequency converter with a linear relationship between the input voltage and the resultant frequency. The reader should refer to the data sheet for the LM131 for further information on the device itself and precautions that should be taken when using the device. *Figure 20* is the basic circuit for using the LM131. *Figure 21* represents improvements that increase the accuracy (by increasing the linearity) of the result. Note that these circuits have been taken from the data sheet of the LM131 and the user is referred there for a further discussion of their individual characteristics. With the LM131 the frequency output is given by the relationship:

$$F_{OUT} = (V_{IN}/2.09) (1/R_T C_T) (R_S/RL)$$

It is clear from the expression above that the accuracy of the result depends upon the accuracy of the external com-

ponents. The circuit may be calibrated by means of a variable resistance in the R_S term (a gain adjust) and an offset adjust. The offset adjust is optional but its inclusion in the circuit will allow maximum accuracy to be obtained. The standard calibration procedure is to trim the gain adjust (R_S) until the output frequency is correct near full scale. Then set the input to 0.01 or 0.001 of full scale and trim the offset adjust to get F_{OUT} to be correct at 0.01 or 0.001 of full scale. With that calibration, the circuit of *Figure 20* is accurate to within $\pm 0.03\%$ typical and $\pm 0.14\%$ maximum. The circuit of *Figure 21* attains the spec limit accuracy of $\pm 0.01\%$.

5.3 VOLTAGE CONTROLLED OSCILLATORS (VCO's)

A VCO is simply another form of voltage to frequency converter. It is an oscillator whose oscillation frequency is dependent upon the input voltage. Numerous designs for VCO's exist and the reader should refer to the data sheets and application notes for various op-amps and VCO devices. The code in *Figure 19* is still applicable if a VCO is used. The only possible difficulty that might be encountered is if the relationship between frequency and input voltage is non-linear. This does not affect the basic code but would affect the processing to create the final result. A sample circuit, taken from the data sheet of the LM358, is shown in *Figure 22*. The accuracy of the VCO is the controlling factor.

5.4 A COMBINED APPROACH

Elements of the period measurement and pulse counting techniques can be combined to produce a system with the advantages of both schemes and with few problems. Such a system is only slightly more complicated in terms of its software implementation than the approaches mentioned above. Note that in a microcontroller driven system, no additional hardware beyond the voltage to frequency converter is required to implement this approach. Basically, the microcontroller establishes a viewing window during which time the microcontroller is both measuring time and counting pulses. The result can be very precise if two conditions are met. First, when the microcontroller determines that it needs the conversion information, the microcontroller does not begin counting time or pulses until the first pulse is received from the VFC (first pulse after the microcontroller "ready"). Note, the COPS microcontroller could provide a "start conversion" pulse to enable the VFC if such an arrangement were desirable. The time would be counted for a fixed period and the number of pulses would be counted. After the fixed period of time the controller would wait for the next pulse from the VFC and continue to count time until that pulse is received. The ratio of the total time to the number of pulse is a very precise result provided that all the system times are slow enough that the microcontroller can do its job. The speed limits mentioned previously apply here. It is clear that the total time is not fixed. It is some basic time period plus some variable time. This is a little more complicated than simply using a fixed time, but it allows greater accuracies to be achieved. Also, the approach takes approximately the same amount of time for all conversions. It is also faster than the simple pulse counting scheme.

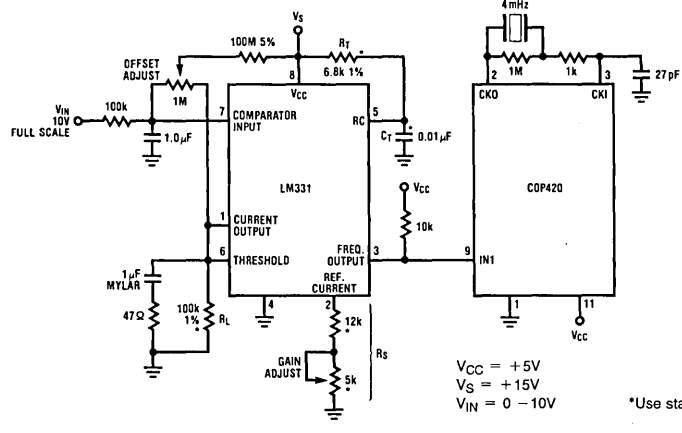


FIGURE 20. Basic LM331 Connection

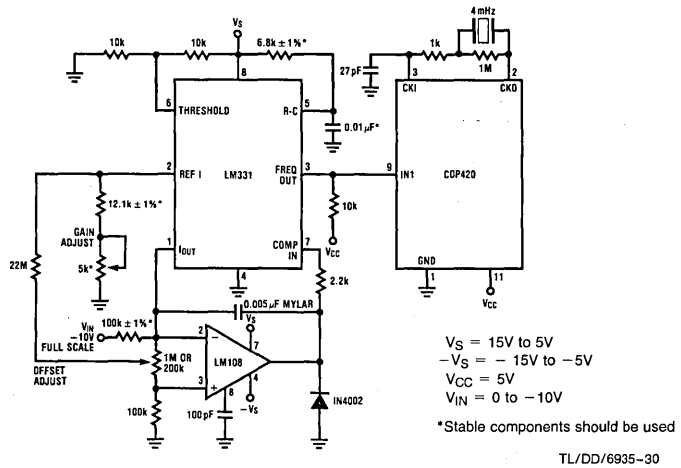


FIGURE 21. A to D with Precision Voltage to Frequency Converter

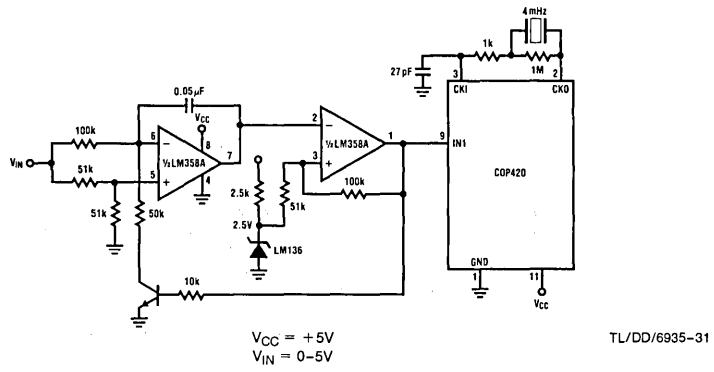


FIGURE 22. A to D with VCO

6.0 Successive Approximation

6.1 BASIC APPROACH

The successive approximation technique is one of the more standard approaches in analog to digital conversion. It requires a counter or register (here provided by the COP420), a digital to analog converter, and a comparator. *Figure 23A/B* illustrates the basic idea with the COP420. In the most basic scheme, the counter is reset to zero and then incremented until the voltage from the digital to analog converter is equal to the input voltage. The equality is determined by means of the comparator. *Figure 24B* illustrates the flow chart and code for this most basic approach. The preferred approach is illustrated in *Figure 25A/B*. This is the standard binary search method. The counter or register is set at the midpoint and the "delta" value set at one half the midpoint. The "delta" value is added or subtracted from the initial guess depending on the output of the comparator. The "delta" value is divided by 2 before the next increment or decrement. The method repeats until the desired resolution is achieved. While this approach is somewhat more complicated than the basic approach it has the advantage of always taking the same amount of time for the conversion

regardless of the value of the input voltage. The conversion time for the basic approach increases with the input voltage. The preferred approach is almost always faster than the basic approach. The basic approach is faster only for those voltages near zero where it has only a few increments to perform.

The accuracy of the approach is governed by the accuracy of the digital to analog converter and the comparator. Thus, the result can be as accurate as one desires depending on the choice of those components. Digital to analog converters of various accuracies are readily available as standard parts. Their cost is usually in direct relation to their accuracy. The reader should refer to the National Semiconductor Data Acquisition Handbook for some possible candidates for digital to analog converters. It is not the purpose here to compare those parts. The COPS interface to these parts is generally straightforward and follows the basic schematics shown in *Figure 23*. The user should take note and make sure the input and output ports of the converter are compatible — in terms of voltages and currents — with the COPS device. This is generally not a problem as most of the parts are TTL compatible on input and output. The precautions and restrictions as to the use of any given device are governed by that device and are indicated in the respective data sheets.

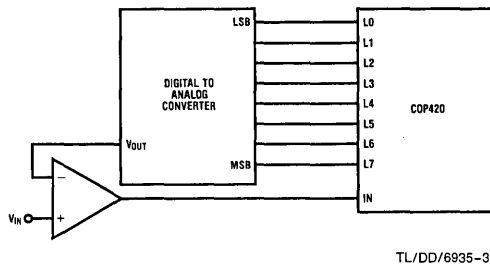
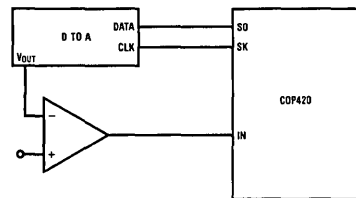


FIGURE 23A. Basic Parallel Implementation



TL/DD/6935-33

FIGURE 23B. Basic Serial Implementation

```

; B BIT SUCCESSIVE APPROXIMATION--BASIC SCHEME
; COMPARATOR INPUT TO COP = IN3
; OUTPUTS TO D TO A ARE L7 THRU L0 WITH L7 = MSB, L0 = LSB

CONVRT: LBI    2, 14    ; SET THE RESULT VALUE TO ZERO
        STII   0
        STII   0
        LEI    4        ; ENABLE THE L PORT AS OUTPUTS
        JP     OUTPUT

INCR:   SC           ; ROUTINE FOR INCREMENTING THE RESULT VALUE
PI USJ: CLRA
        LBI    2, 14
        ASC
        NOP
        XIS
        JP     PLUS1

OUTPUT: LBI    2, 15    ; SEND THE RESULT VALUE, STORED IN 2, 15-2, 14 TO
        LD     ; G AND THEREBY OUT THROUGH L
        XDS
        CAMG
        JSR    DELAY

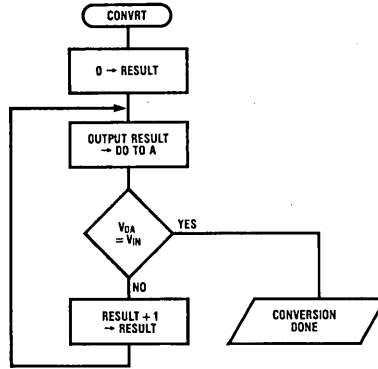
        ININ        ; NOW READ THE COMPARATOR INPUT TO COP
        AISC   8    ; COULD SAVE A WORD IF USE G LINE AS INPUT
        JP     INCR ; INPUT VOLTAGE STILL > CONVERTED ANALOG VOLTAGE

; CONVERSION DONE AT THIS POINT--THE COMPARATOR HAS CHANGED STATE
; HENCE, CONVERTED ANALOG VOLTAGE > INPUT VOLTAGE--SO STOP

```

FIGURE 24A. Code for Basic Approach of Successive Approximation

TL/DD/6935-51



TL/DD/6935-34

FIGURE 24B. Basic Approach, Successive Approximation

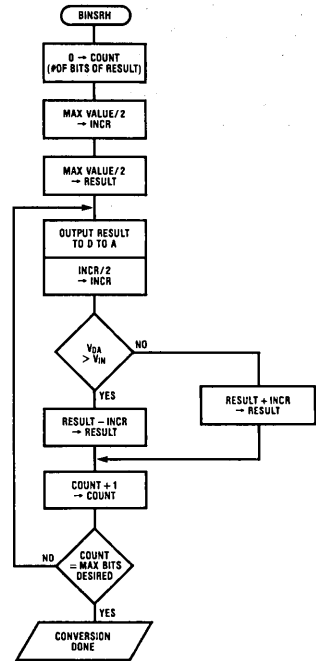
```

; 8 BIT BINARY SEARCH SUCCESSIVE APPROXIMATION
; INPUT TO COP IS IN3.L BUS IS OUTPUT TO D TO A. L7=MSB, L0=LSB
; COMPARTOR=0 WHEN D TO A VOLTAGE > VIN, OTHERWISE = 1

BINCRH:  LBI    3, 14 ; SET INCREMENT = MAX VALUE/2(WILL BECOME
          STII   0    ; MAX VALUE/4 BEFORE FIRST USE)
          STII   8
          LDI    2, 14 ; SET INITIAL VALUE OF RESULT TO MAX VALUE/2
          STII   0
          STII   8
          LEI    4    ; ENABLE THE L BUS AS OUTPUTS
          LDI    1, 15 ; NOW SET UP THE BIT COUNTER-OVERFLOW WHEN 8 BITS
          CLRA
          AISC   9    ; DO IT THIS WAY FOR COMPATIBILITY WITH INCREMENT
OUTPUT:  X      3    ; SAVE THE BIT COUNTER VALUE AND POINT TO RESULT
          LD     3
          XDS
          CAMQ
DIVIDE:  LBI    3, 15 ; DIVIDE THE INCREMENT VALUE BY 2, CAN BE DONE
          LD     4    ; IN SEVERAL WAYS SINCE THIS IS A VERY SPECIAL
          AISC   8    ; PURPOSE DIVIDE FUNCTION
          JP    DIV1 ; ALSO, DO THE DIVIDE HERE TO GIVE THE D TO A TIME
          STII  4    ; TO DO THE DIGITAL TO ANALOG CONVERSION
          JP    TEST
DIV1:    AISC   4    ;
          JP    DIV2
          STII  2
          JP    TEST
DIV2:    AISC   2
          JP    DIV3
          STII  1
          JP    TEST
DIV3:    LBI    3, 14
          AISC   1
          JP    DIVA
          STII  8
          STII  0
          ; DEPENDING ON THE D TO A USED, MAY NEED MORE DELAY HERE
          ; MUST BE SURE THE RESULT IS STEADY BEFORE TEST THE COMPARTOR
TEST:    LBI    3, 14
          ININ
          AISC   8    ; COULD SAVE A WORD IF USED Q LINE AS INPUT
          JP    INCR
          SC     1    ; INPUT LESS THAN D TO A CONVERTED VOLTAGE
          SUB   1    ; SUBTRACT THE INCREMENT VALUE FROM RESULT
          CASC
          NOP
          XIS    1
          JP    SUB
          JP    BITPL1
          INCR:  RC     1 ; INPUT > D TO A CONVERTED VOLTAGE
          LD     1    ; ADD THE INCREMENT VALUE TO RESULT VALUE
          ADD
          NOP
          XIS    1
          JP    ADD
          BITPL1: LBI   1, 15 ; NOW INCREMENT BIT COUNTER TO SEE IF DONE
          LD     1
          AISC   1
          JP    OUTPUT
          ; CONVERSION DONE AT THIS POINT
  
```

TL/DD/6935-52

FIGURE 25A. Binary Search Successive Approximation Code



TL/DD/6935-35

FIGURE 25B. Binary Search Successive Approximation Flow Chart

6.2 SOME COMMENTS ON RESISTOR LADDERS

If the user does not wish to use one of the standard digital to analog converters, he can always build one of his own. One of the most standard methods of doing so is to use a resistor ladder network of some form. *Figure 26* illustrates the basic forms of binary ladders for digital to analog converters. The figures also show the transition from the basic binary weighted ladder in *Figure 26A* to the standard R-2R ladder *Figure 26C*.

Consider *Figure 26A*. The choice of the terminating resistor is made by hypothesizing that the ladder were to go on ad infinitum. It can then be shown that the equivalent resistance at point X in that figure would be equal to $128R$, the same value as the resistor to the least significant bit output. This fact is used to create the intermediate ladder of *Figure 26B*. This step is done because it is usually undesirable to have to find the multitude of resistor values required in the basic binary ladder. Thus, the modification in *Figure 26B* significantly reduces the number of resistor values required. As stated earlier, the resistance looking down the ladder at point X in *Figure 2* is equal to the resistor connected to the binary output at that point; here the value is $2R$. Remembering the objective is to minimize the number of different values required, if we simply use the same R-2R arrangement as before with a termination of $2R$ we get an effective resistance at point Y of *Figure 26B* or $0.5R$. This means that a serial resistance of $1.5R$ is required to maintain the integrity of the ladder. If we carry this on through 8 bits, the circuit of

Figure 26B results. From this it is only a small step to create the standard R-2R network. The analysis is the same as done previously.

There is absolutely no restriction that the ladders must be binary. A ladder for any type of code can be constructed with the same techniques. Ladders comparable to *Figures 26A* and *26B* are shown in *Figure 27* for a standard 8421 BCD code. With the BCD code, the input must be considered in groups of digits with four bits creating one digit. This is the direct analog of 1 binary digit per unit. We need four inputs to create one decimal digit. Thus the resistor values in each decimal digit are 10 times the values in the previous decimal digit just as the resistor value for each successive binary digit was twice the value for the preceding binary digit. Note that this analysis can be easily extended to any code. The termination resistance is calculated in the same manner—assume the decimal digit groupings extend out to infinity. It can be shown that the resistance of the ladder at point X in *Figure 27A* is $480R$. Thus *Figure 27A* represents the basic 8241 BCD ladder for three digit BCD number. This termination resistance will vary with where it is placed. Basically this resistance is equal to nine times (for a decimal ladder) the parallel resistance of the last digit implemented. (This relation can be shown mathematically if one desired, the multiplier is a function of the type of ladder used—multiplier = 1 for binary systems, 9 for decimal systems, etc.) Thus the termination resistance would be $48R$ if the network were terminated after the 2nd digit and $4.8R$ if the network were terminated after the 1st digit implemented. In

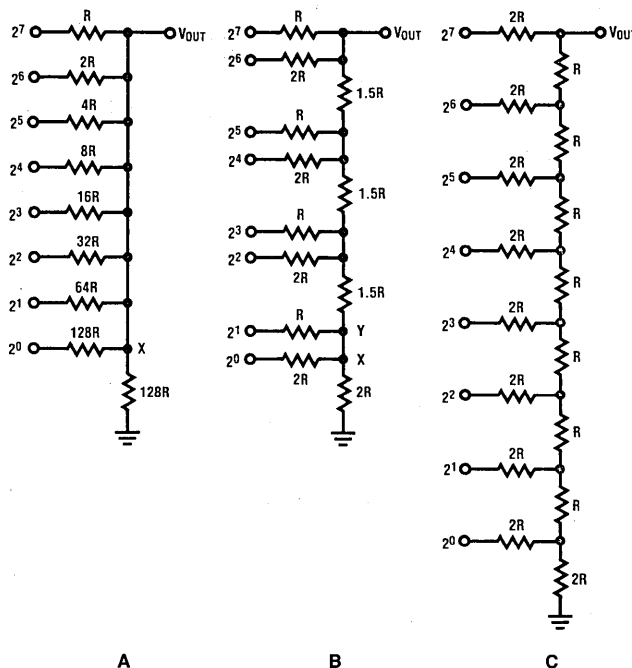


FIGURE 26. Binary Ladders

TL/DD/6935-36

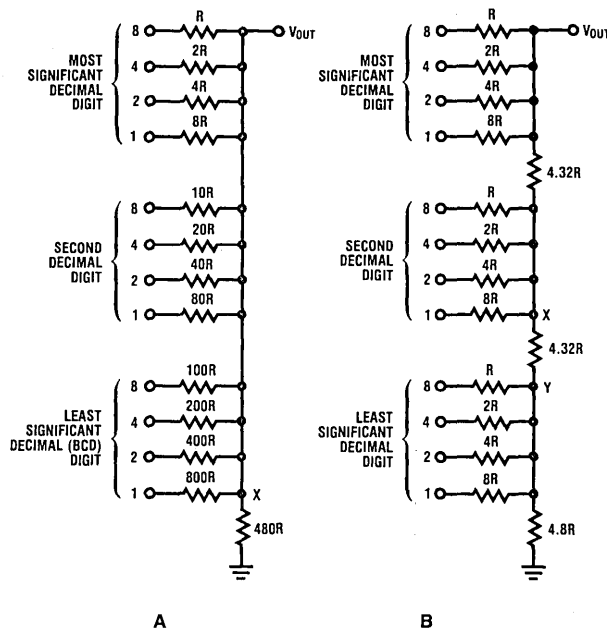
Figure 27B we are attempting to use only the resistor values for one decimal digit. This means that the last terminating resistor must be a $4.8R$ by the analysis above. Thus at point X in Figure 27B we must have an equivalent of resistance of $4.8R$. The equivalent resistance at point Y of Figure 27B, looking down from the ladder, is $0.48R$. Thus the other series resistance must be $4.32R$ ($4.8R - 0.48R$). Thus the network of Figure 27B results.

Generally, ladders can be very effective tools when understood and used properly. They can be significantly more involved than indicated here. There are a number of texts and articles that cover the subject very nicely and the reader is referred to them if more information on ladder design, the use of ladders, and advanced techniques with ladders is desired.

One final note is of some interest. The ladders may be readily constructed for any type of code to create the analog voltage. Note that there is no restriction that the code, or the ladder network, be linear. Thus, effective use of ladder networks may significantly reduce system difficulties and

complexities caused by the fact that the analog to digital conversion is being performed on a voltage source that changes nonlinearly, for example a thermistor temperature probe. By using the properly designed ladder network, the nonlinearity can effectively be eliminated from consideration in the code implementation of the analog to digital conversion.

The accuracy of ladders is a direct function of the accuracy of the resistors and the accuracy of the voltage source inputs. This is obvious since the analog voltage is in fact created by means of equivalent voltage dividers created when the various inputs are on or off. It is also essential that the ladder sources be the precise same value at all inputs to the ladder network. If this is not the case, errors will be introduced. In addition, the output impedance of the voltage source should be as small as possible. The success of the ladder scheme depends on the ratios of the resistance values. Inaccuracies are introduced if those ratios are disturbed. Some possible implementations of the successive approximation approach with a ladder network used for the digital to analog conversion are indicated in Figure 28.



TL/DD/6935-37

FIGURE 27. 8421 BCD Ladders

Note that these are functional diagrams. Feedback or hysteresis for comparator stabilization are not shown. The reader should be aware that his particular application may require that these factors be considered. *Figure 28A* is the simplest scheme and also the least accurate. With little or no load, the high output level of the L buffer should be very close to V_{CC} and the low level close to ground. Also the output impedance of the buffers must be considered. Therefore, rather large resistor values are used—both to keep the load very small and to dwarf the effect of the output imped-

ance. With the configuration in *Figure 28A*, four bit accuracy is about the best that can be achieved. By being extremely careful and using measured values, an additional bit of accuracy may be obtained but care must be used. However, the schematic of *Figure 28A* is very simple. *Figure 28B* represents the next step of improvement. Here we have placed CMOS buffers in the network. This eliminates the output impedance and reduces the level problems of the circuit of *Figure 28A*. The CMOS buffer will swing rail to rail, or nearly so. The accuracy of V_{CC} and the resistor network is then

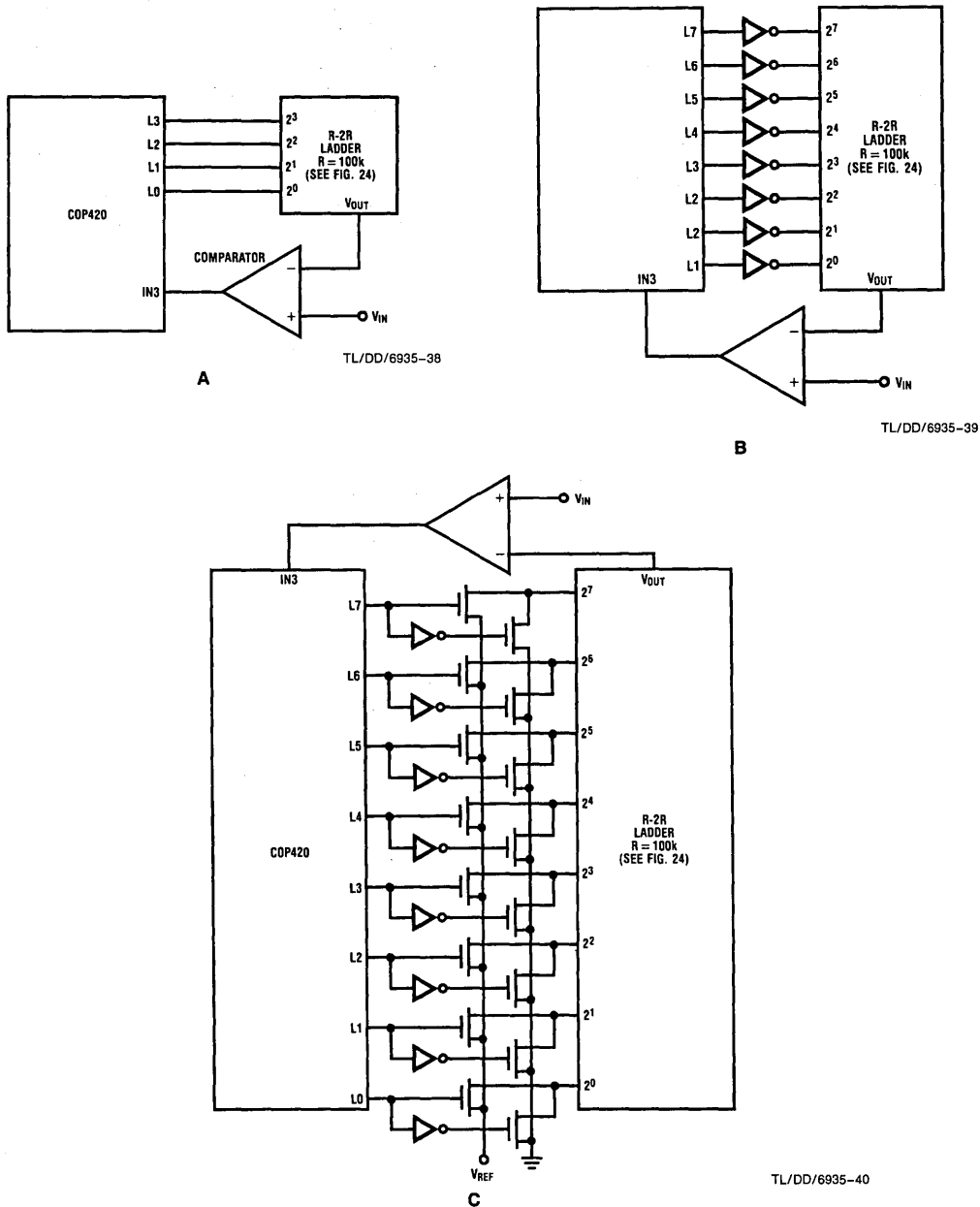


FIGURE 28. Interfaces to Ladder Networks

controlling. Using 1% resistors and holding V_{CC} constant, the user should be able to achieve 7 to 8 bit accuracy without much difficulty. Remember, however, that V_{CC} is one of the controlling factors. If V_{CC} is $\pm 5\%$, there is no point in using 1% resistors since the V_{CC} tolerance swamps their effect. *Figure 28C* is the final and most accurate approach. Naturally enough, it is the most expensive. However, one can get as accurate as one desires. Here, an accurate reference is required. That reference is switched into the network by means of the analog switch. Alternately, ground may be connected to the input. Now the user need only consider the accuracy of the reference and the accuracy of the resistors. However, the on impedance of the switches must be considered. It is necessary to make this on impedance as low as possible so as not to alter the effective resistor values.

7.0 "Offboard" Techniques

7.1 GENERAL COMMENTS

This section is devoted to a few illustrations of interfacing the COP420 to standard, stand alone analog to digital converters. These standard converters are used as peripherals to the COPS device. Whenever the microcontroller requires a new reading of some analog voltage, it simply initiates a read of the peripheral analog to digital converter. As a result, the accuracies and restrictions in using the converters are governed by those devices and not by the COPS device. These techniques are generally applicable to other A to D

converters not mentioned here and the user should not have difficulty in applying these principles to other devices. It should be pointed out that in almost every instance, the choice of COP420 inputs and outputs is arbitrary. Obviously, when there is an 8-bit bus it is natural, and most efficient, to use the L port to interface to the bus. Generally, the G lines have been used as outputs rather than the D lines simply because the G lines are, in many instances, somewhat easier to control. The choice of input line is also free. If the interrupt is not otherwise being used, it may be possible to utilize this feature of IN1 for reading a return signal from the converter. However, this is by no means required. If there is a serial interface it is clearly more efficient to use the serial port of the COP420 as the interface. If a clock is required, SK is the natural choice.

7.2 ADC0800 INTERFACE

The ADC0800 is an 8-bit analog to digital converter with an 8-bit parallel output port with complementary outputs. The ADC0800 requires a clock and a start convert pulse. It generates an end of conversion signal. There is an output enable which turns the outputs on in order to read the 8-bit result.

The reader is referred to the data sheet for the ADC0800 for more information on the device. The circuit of *Figure 29* illustrates the basic implementation of a system with the ADC0800. The interface to the COP420 is straightforward. The appropriate timing restrictions on the control signals are easily met by the microcontroller.

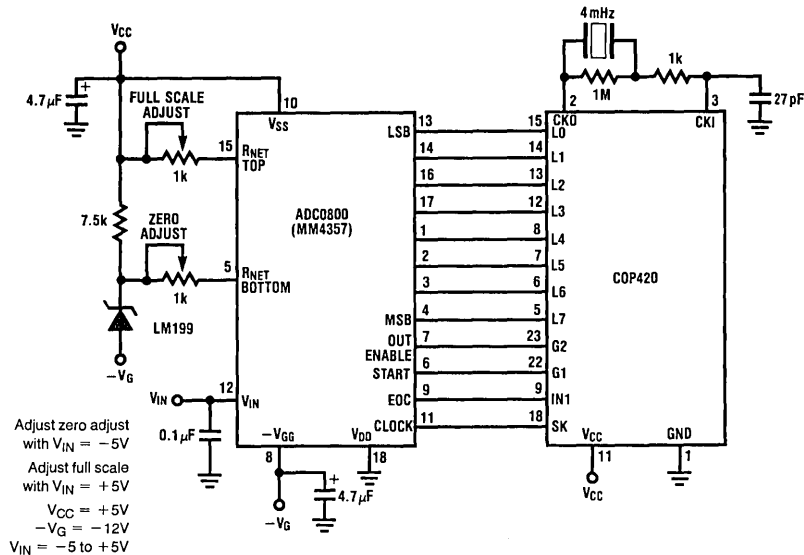


FIGURE 29. Simple A/D with ADC0800

TL/DD/6935-41

Figure 30 is the flow chart and code required to do the interfacing. As can be seen, the overhead in the COP420 device is very small. The choice of inputs and outputs is arbitrary. The only pin that is more or less restricted is the use of SK as the clock for the converter. SK is clearly the output to use for that function as, when properly enabled, it provides pulses at the instruction cycle rate.

7.3 ADC0801/2/3/4 INTERFACE

The ADC0801 family of analog to digital converters is very easy to interface and is generally a very useful offboard con-

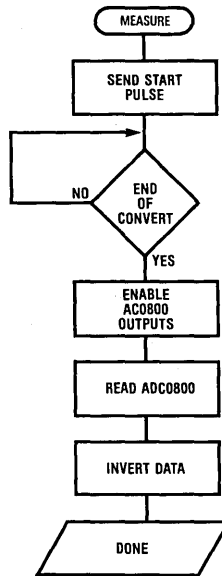
verter. The interface is not significantly different from that of the ADC0800, but the ADC0801 family are a much better device. The four control signals are somewhat different, although there are still four control lines. Here we have a chip select, a read, a write, and an interrupt signal. All are negative going signals. Start conversion is the ANDing of chip select and write. Output enable is the ANDing of chip select and read. The interrupt output is an end convert signal of sorts. The device may be clocked externally or an RC may be connected to it and it will generate its own clock for the conversion. In addition the device has differential inputs

```

MEASUR: LEI    0      ; FLDAT THE L LINES
        SC
START2: CLRA           ; MAKE SURE SD STAYS ZERO
        XAS           ; MAKE SURE SK STAYS CLOCK
        OGI    2      ; SEND START PULSE
        OGI    0
        LBI    2, 13
READ11: ININ
        AISC    14     ; WAIT FOR EOC SIGNAL
        JP     READ11
        OGI    4      ; HAVE EOC, ENABLE OUTPUTS
        INL           ; READ THE L LINES
        X
        COMP           ; CREATE PROPER POLARITY
        XDS
        COMP
        X
        OGI    0      ; DISABLE ADC0800 OUTPUT
        ; HAVE THE RESULT AT THIS POINT--USE IT IN WHATEVER
        ; MANNER IS REQUIRED BY THE APPLICATION
        LBI    2, 10
        JSRP    CLRR
        JP     MEASUR
    
```

TL/DD/6935-53

FIGURE 30A. A to D with ADC0800



TL/DD/6935-42

FIGURE 30B. ADC0800 Interface Flow

which allow the 8-bit conversion to be performed over a given window or range of input voltages. The reader should refer to the ADC0801 family data sheet for more information. *Figure 31* indicates a basic interface of the ADC0801 family to the COP420. Again, the interface is simple and straightforward. The code required to interface to the device is minimal. *Figure 32* illustrates the flow chart and code required to do the interface.

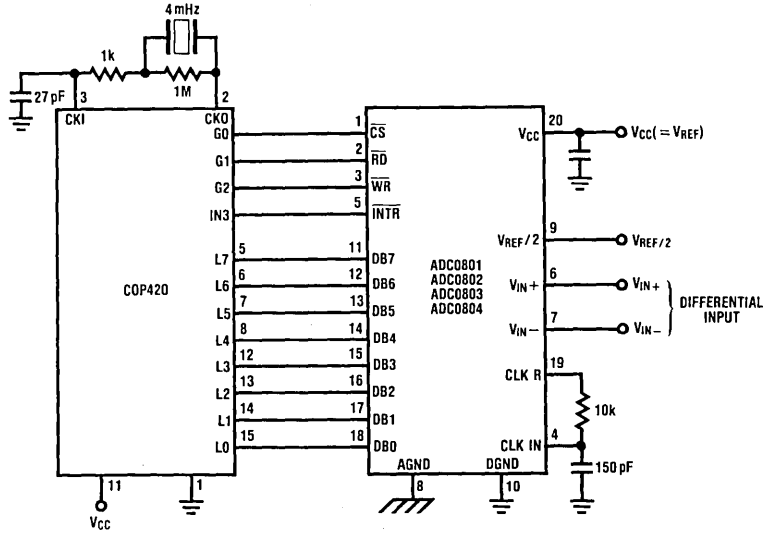


FIGURE 31. COP420—ADC0801 Family Interface

TL/DD/6935-43

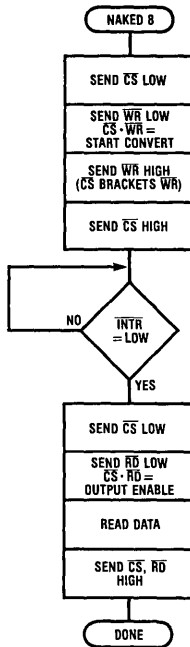
```

; INTERFACE TO NAKED B
;
NAKEDB: OGI 15 ; SET ALL G LINES HIGH (USUALLY DONE AT
; POWER UP
LOOP1: LEI 0 ; TRI STATE THE L LINES FOR READING
OGI 14 ; SEND CHIP SELECT LOW (CS BRACKETS OTHER SIGNAL)
OGI 10 ; CS LOW AND WR LOW = START CONVERSION
OGI 14 ; RAISE WR
OGI 15 ; RAISE CS, NAKED B IS NOW CONVERTING
LOOP2: ININ ; WAIT FOR THE INTR SIGNAL--COULD SAVE THIS TEST
AISC 8 ; IF USED IN1 AND THE INTERRUPT FEATURE OF COP4
JP READ ; INTR IS LOW, DATA IS READY
LOOP2: JP LOOP2
RI-AD: LBI 0,0 ; SET UP RAM LOCATION FOR READ
OGI 14 ; SEND CS
OGI 12 ; SEND CS AND READ = OUTPUT ENABLE
NOP ; WAIT--NEED WAIT ONLY 125NS, BUT 1 CYCLE IS MIN
; TIME WE CAN WAIT
INL ; READ THE L LINES
OGI 15 ; TURN OFF THE NAKED B--CS AND RD HIGH
;
; DONE AT THIS POINT, DO WHATEVER IS REQUIRED WITH THE RESULT
;

```

FIGURE 32A. COP420/ADC0801 Family Sample Interface Code

TL/DD/6935-54



TL/DD/6935-44

FIGURE 32B. COP420/ADC0801 Family Interface Flow

8.0 Conclusion

Several analog to digital techniques using the COPS family have been presented. These are by no means the only techniques possible. The user is limited only by his imagination and whatever parts he can find. The COPS family of parts is extremely versatile and can readily be used to perform the analog to digital conversion in almost any method. Generally, those techniques where the COPS device is doing the counting or timekeeping are slow. However, those techniques are generally slow inherently. The fastest methods are those where the conversion is being done offboard and the COPS device is merely reading the result of the conversion when required. Also, an attempt has been made to illustrate the lower cost techniques of analog to digital

conversion. This, by itself, restricts most of the techniques described to about 8-bits accuracy. As was mentioned several times, the greater the accuracy that is desired the more accurate the external circuits must be. Ten and twelve-bit accuracies, and more, require references that are accurate. These get very expensive very rapidly. There is nothing inherent in the COPS devices that prevents them from being used in accurate systems. The precautions are to be taken in the system regardless of the microcontroller. The only problem is that, in those accurate systems where the COPS device is doing the timekeeping and counting, this increased accuracy is paid for by increased time to perform the conversion.

Several devices have been used in conjunction with the COPS device in the previous sections. It is again recommended that the user refer to the specific data sheets of those devices when using any of those circuits. It must again be mentioned that the standard precautions when dealing with analog signals and circuits must be taken. These are described in the National Semiconductor Linear Applications Handbook and in the data sheets for the various linear devices. These precautions are especially significant when greater accuracy is desired.

The COPS family of microcontrollers has shown itself to be very versatile and powerful when used to perform analog to digital conversions. Most techniques are code efficient and the microcontroller itself is almost never the limiting factor. It is hoped that this document will provide some guidance when it is necessary to perform analog to digital conversion in a COPS system.

9.0 References

1. "Digital Voltmeters and the MM5330", National Semiconductor Application Note AN-155.
2. Walker, Monty, "Exploit Ladder Network Design Potential". Part One of two part article on ladder networks. Magazine and date unknown.
3. Wyland, David C., "VFC's give your ADC design high resolution and wide range". *EDN*, Feb. 5, 1978.
4. Redfern, Thomas P., "Pulse Modulation A/D Converter" *Society of Automotive Engineers Congress and Exposition Technical paper #780435*, March 1978.
5. National Semiconductor Linear Applications Handbook, 1978.
6. National Semiconductor Linear Databook, 1980.
7. National Semiconductor Data Acquisition Handbook, 1978.

The COP444L Evaluation

National Semiconductor
COP Note 4
Leonard A. Distaso



The 444L-EVAL is a software program intended to be used with the COP444LP to demonstrate operating characteristics and facilitate user familiarization and evaluation of the COP444L and the COPSTM family in general. This software program is available on Dial-a-Helper.

The 444L-EVAL has two mutually exclusive operating modes: an up/down counter/timer or a simple music synthesizer. The state of pin L7 at power up determines the operating mode.

1.0 THE 444L-EVAL AS A SIMPLE MUSIC SYNTHESIZER

Figure 1 indicates the connection of the 444L-EVAL as a simple music synthesizer. As the diagram indicates, the connections required for operation are minimal. The os-

illator may be a crystal circuit using CKI and CKO; an external oscillator to CKI; or an RC network using CKI and CKO. As should be expected, the crystal circuit provides the greatest frequency stability and precision. The RC network will provide an acceptable oscillation frequency but that frequency will be neither precise nor stable over temperature and voltage. The external oscillator, of course, is as good as its source. The frequencies for the various notes and delay times are set up assuming that the oscillator frequency is 2 MHz. Three modes of operation are available in the music synthesizer mode: play a note; play one of four stored tunes; or record a tune for subsequent replay.

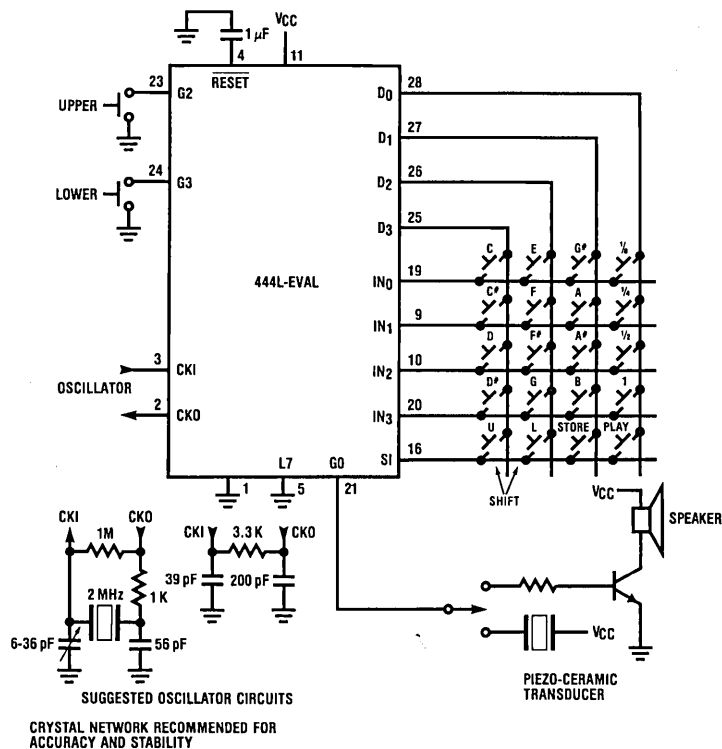


FIGURE 1. 444L-EVAL as Simple Music Synthesizer

TL/DD/6937-1

1.A. PLAY A NOTE

Twelve keys, representing the twelve notes in one octave, are labeled "C" through "B". Depressing a key causes a square wave of the corresponding frequency to output at GO. The user may drive a piezo-ceramic transducer directly with this signal. With the appropriate buffering, the user may use this signal to drive anything he wishes. A simple transistor driver is sufficient to drive a small speaker. The user can be as simple or as complex as he desires at this point—e.g. he can do some wave shaping, add an audio amplifier, and drive a high quality speaker.

The 444L-EVAL has a range of two and one-half octaves: the basic octave on the keyboard (which is middle C and the 11 notes above it in the chromatic scale), one full octave above the basic octave and one-half octave below the basic octave. The notes in the basic octave are played by depressing the appropriate key (one key at a time—the keyboard has no rollover provisions). A note in the upper octave is played by first depressing and releasing the U SHIFT key and then depressing the note key. Similarly, a note in the lower one-half octave is played by first depressing and releasing the L SHIFT key and then depressing the note key. Two other shift keys are present: UPPER and LOWER. All notes played while the UPPER key is held down will be in the upper octave. Similarly, note F# through B when played while the LOWER key is held down will be in the lower one-half octave. The lower octave notes C through F are not present and depressing any of these 6 keys while the LOWER key is held down or after depressing the L SHIFT key will play the note in the basic octave.

1.B. PLAY STORED TUNE

The 444L-EVAL can play four preprogrammed tunes. Depressing PLAY followed by "1/8", "1/4", "1/2", or "1" will cause one of these tunes to be played. The tunes are:

- PLAY 1 —Music Box Dancer
- PLAY 1/2 —Santa Lucia
- PLAY 1/4 —Godfather Theme
- PLAY 1/8 —Theme from Tchaikowsky Piano Concerto #1

1.C. RECORD A TUNE

Any combination of notes and rests up to a total of 48 may be stored in RAM for later replay. A note is stored by depressing the appropriate key(s), followed by the duration of the note (1/16 note, 1/8 note, 3/16 note, 1/4 note, 3/8 note, 1/2 note, 3/4 note, whole(1) note), followed by STORE. A rest is stored by selecting the duration and depressing STORE. The rests or durations of 1/16, 3/16, 3/8, and 3/4 are obtained by first depressing L SHIFT and then 1/8, 1/4, 1/2, or 1 respectively. When the tune is complete press PLAY followed by STORE. The tune will be played for immediate audition. Subsequent depression of PLAY and then STORE will play the last stored tune.

Only one tune may be stored, regardless of length. Attempts to store a new or second tune will erase the previously stored tune. There are no editing features in this

mode. (In a "real system" of this type some form of editing would be desirable. It would not be difficult to add editing features.)

Note: The accuracy of the tones produced is a function of the oscillator accuracy and stability. The crystal oscillator, or an accurate, stable external oscillator is recommended.

2.0. THE 444L-EVAL AS AN UP/DOWN COUNTER/TIMER

By connecting pin L7 to V_{CC} and providing power and oscillator the 444L-EVAL functions as an 8 digit binary/BCD up/down counter. In addition, an approximate 1 Hz signal is produced by the device. The 444L-EVAL can drive a single digit LED display directly. With the appropriate driver (COP472, COP470, MM5450/5451) the device can drive a 4 digit LCD, VF, or LED display. Any combination of these displays can be connected at any given time.

The binary/BCD and up/down modes are controlled by the states of input pins IN0 and IN2 as indicated below:

- IN0 = 1 (Default state) —BCD counter
- IN0 = 0 —Binary Counter
- IN2 = 1 (Default state) —Count Up
- IN2 = 0 —Count Down

The up/down control may be changed at any time. Changing the binary-BCD control during operation clears the counter before counting begins in the new mode.

Pins G2 and G3 provide display control to the user. He can choose to view either the most significant 4 digits of the counter or the least significant 4 digits of the counter. Further, the user can disable the update of the 4 digit displays. The controls are as follows:

- G2 = 1 (Default state) —Enable update of 4 digit displays
- G2 = 0 —Disable update of 4 digit displays
- G3 = 1 (Default state) —Display least significant 4 digits of counter
- G3 = 0 —Display most significant 4 digits of counter

The single digit LED display displays the least significant digit of the counter. (Note, the direct drive capability for the single digit LED display refers to a small LED digit—NSA1541A, NSA1166k, or equivalent.)

2.A. I/O MODE

The 444L-EVAL has the capability to allow the user to read or write the 8 digit counter through the L port. In the I/O mode, the single digit LED display is disabled. The 4 digit displays are not affected. In this mode pins D0 and IN3 are used for the handshaking sequence. D0 is a Ready/Write signal from the 444L-EVAL to the outside; IN3 is a Write/Acknowledge from the outside to the 444L-EVAL. Data I/O is via L0–L3 with L0 being the least significant bit. Data is standard BCD for the BCD counter mode or standard hex for the binary counter mode. The digit address is on pins L4–L6 with L4 being the least significant bit. Digit address

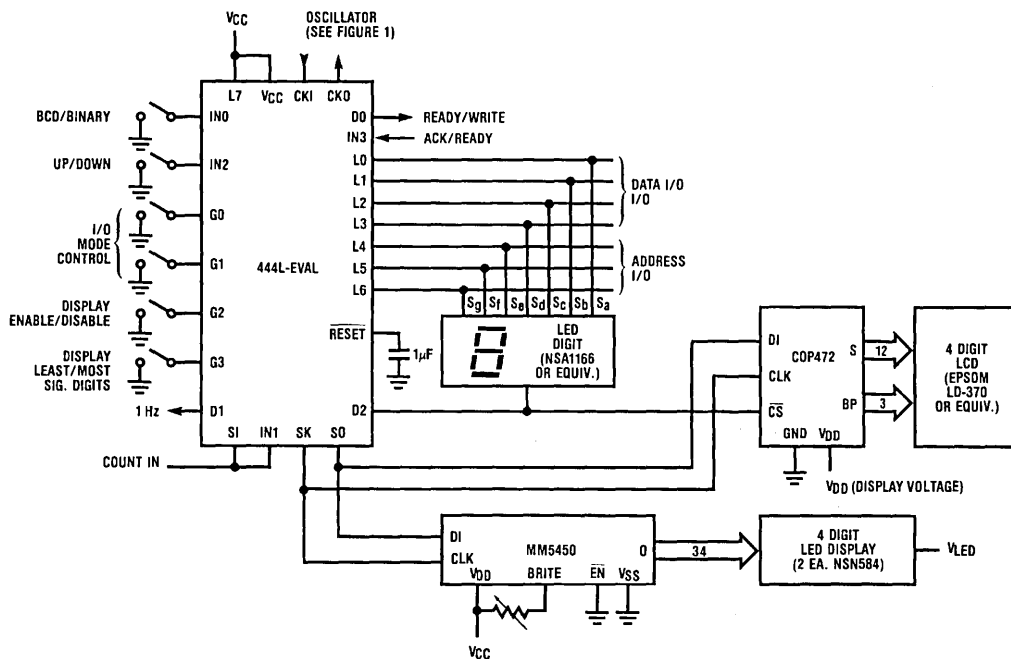


FIGURE 2. 444L-EVAL in Counter Mode

TL/DD/6937-2

0 is the least significant digit of the counter; digit address 7 is the most significant digit of the counter. The I/O modes are controlled by pins G0 and G1 as follows:

G0	G1	Output data with handshake, single digit LED off
0	0	Output data with handshake, single digit LED off
0	1	Input data with handshake, single digit LED off
1	0	Auto output, no handshake, single digit LED on
1	1	Default condition, No I/O, single digit LED displays least significant digit of counter

2.A.1. Output Data with Handshake

With this mode selected the 444L-EVAL will output data with a handshake sequence. Note that the outputting of data is relatively slow as the device is counting and updating displays between successive digit outputs.

Before data is output, or the next digit of the counter is output, the 444L-EVAL must see IN3 (Acknowledge or ready from the external world high). The Ready/Write pin (D0) is assumed to be high at this point. With D0 high and IN3 high, the device will output the data and digit address. After the data and address are output, the D0 line—functioning as a write strobe here—goes low. The 444L-EVAL then expects the signal at IN3 to go low indicating that the external world has read the data. When the device sees IN3 go low, D0 will be brought high indicating that the sequence

is ready to repeat as soon as IN3 goes high again. The counter digits are output sequentially from least significant digit (digit address 0) through most significant digit (digit address 7). The sequence will continuously repeat as long as this mode is selected.

2.A.2. Input Data with Handshake

The 444L-EVAL will take data supplied to it and load the counter. The sequence is similar to that described above for the output mode. The external device(s) supplies both the data and the digit address where that data is to be loaded. When sending data to the 444L-EVAL, the external circuitry must test that the device is ready to receive data (D0 high). Then the data and address should be presented at the L port. Then the Write signal (IN3) should be driven low. The 444L-EVAL will read the data and then drive D0 low. When D0 goes low, the external circuitry should bring IN3 high. After IN3 returns high, the 444L-EVAL will signal it is ready to receive data by sending D0 high. Note that this sequence is relatively slow. The 444L-EVAL is performing several operations between successive read operations.

2.A.3. Automatic Output Mode

In the automatic output mode, the single digit LED is on. It is not displaying the least significant digit of the counter in this mode. The display is on so that the user can connect this LED digit, select the automatic output mode, and observe the states of the L lines without having to put more sophisticated equipment or circuitry external to the 444L-EVAL. Segments a through d are pins L0 through L3; segments,

e, f, g are pins L4, L5, and L6. Thus the user can observe the digit address changing and observe the corresponding data.

In this mode, the state of pin IN3 is irrelevant. The 444L-EVAL sequentially outputs the digits of the counter.

D0 goes high when the data and address is being changed. D0 goes low when the data is valid. As in the other I/O modes, the process is slow. There is about 4 to 5 milliseconds between the successive digit outputs.

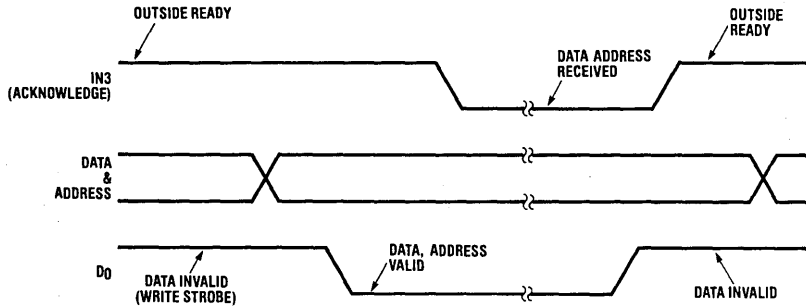


FIGURE 3A. Relative Timing—Output Handshake

TL/DD/6937-3

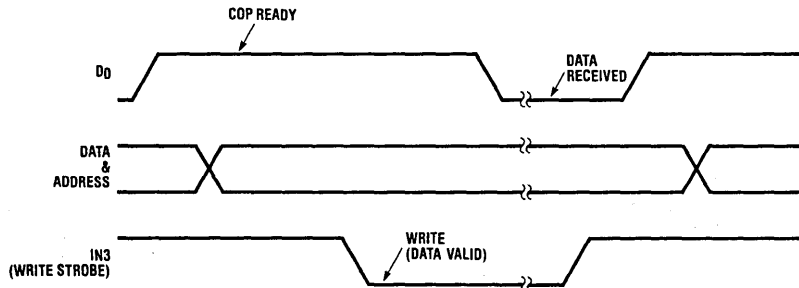


FIGURE 3B. Relative Timing—Input Handshake

TL/DD/6937-4

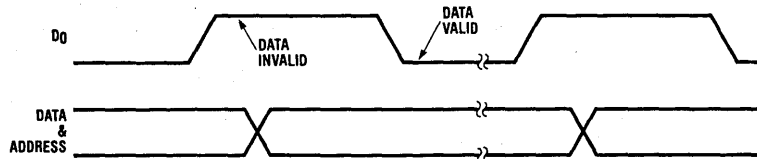


FIGURE 3C. Relative Timing—Automatic Output

TL/DD/6937-5

3.0 SELECTED OPTIONS

The 444L-EVAL has the following options selected:

GND	Option 1 = 0	
CKO	Option 2 = 0	CKO is clock generator output to crystal
CKI	Option 3 = 0	CKI oscillator input divide by 32
RESET	Option 4 = 0	Load device to V_{CC} on RESET
L7	Option 5 = 0	Standard output on L7
L6	Option 6 = 2	High current LED direct segment drive on L6
L5	Option 7 = 2	High current LED direct segment drive on L5
L4	Option 8 = 2	High current LED direct segment drive on L4
IN1	Option 9 = 0	Load device to V_{CC} on IN1
IN2	Option 10 = 0	Load device to V_{CC} on IN2
V_{CC}	Option 11 = 1	4.5V to 9.5V operation
L3	Option 12 = 2	High current LED direct segment drive on L3
L2	Option 13 = 2	High current LED direct segment drive on L2
L1	Option 14 = 2	High current LED direct segment drive on L1
L0	Option 15 = 2	High current LED direct segment drive on L0
SI	Option 16 = 0	Load device to V_{CC} on SI
SO	Option 17 = 2	Push-pull output on SO
SK	Option 18 = 2	Push-pull output on SK
IN0	Option 19 = 0	Load device to V_{CC} on IN0
IN3	Option 20 = 0	Load device to V_{CC} on IN3
G0	Option 21 = 0	Very high current standard output on G0
G1	Option 22 = 2	High current standard output on G1
G2	Option 23 = 4	Standard LSTTL output on G2
G3	Option 24 = 4	Standard LSTTL output on G3
D3	Option 25 = 0	Very high current standard output on D3

D2	Option 26 = 0	Very high current standard output on D2
D1	Option 27 = 0	Very high current standard output on D1
D0	Option 28 = 0	Very high current standard output on D0
	Option 29 = 0	Standard TTL input levels on L
	Option 30 = 0	Standard TTL input levels on IN
	Option 31 = 0	Standard TTL input levels on G
	Option 32 = 0	Standard TTL input levels on SI
	Option 33 = 1	Schmitt trigger inputs on RESET
	Option 34 = 0	CKO input levels, not used here
	Option 35 = 0	COP444L
	Option 36 = 0	Normal RESET operation

4.0 CONCLUSION

The 444L-EVAL demonstrates much of the capability of the COP444L. It does not indicate the limits of the device by any means. The I/O features were included to demonstrate that capability. The fact that they are slow is due strictly to the program. If such I/O capability were a necessary part of an application it could be accomplished much much faster than was done here. The counter modes are quite versatile and are generally self explanatory. It was fairly easy to provide a counter with the versatility of that included here. The music synthesis mode demonstrates clearly the program efficiency of the device.

The 444L-EVAL is intended for demonstration. There is no question that aspects of its operation could be improved and tailored to a specific application. It is unlikely that this particular combination of features would be found in any one application. It is also interesting to note that the program memory in the device is not full. There is still a significant amount of room left in the ROM. This should serve to make it clear that the capabilities of the device have not been stretched at all in order to include these demonstration functions.

Oscillator Characteristics of COPS™ Microcontrollers

National Semiconductor
COP Note 5



Table of Contents

1.0 INTRODUCTION

2.0 RC OSCILLATOR OPTION

3.0 CRYSTAL OR INVERTER OPTION

3.1 COP420/COP402

3.1.1 L, LC, and RLC Networks

3.2 COP420L

3.3 COP410L

3.4 General Notes

4.0 CONCLUSION

1.0 INTRODUCTION

COPS microcontrollers will operate with a wide variety of oscillator circuits. This paper focuses on two of the oscillator options available on COPS microcontrollers: the internal RC oscillator, and the crystal or inverter oscillator. The typical behavior of the RC oscillator with temperature and voltage (and typical values of R and C) is documented. For the crystal or inverter option, circuit configurations (RC, RL, RLC, R, LC, L) are presented which will allow the microcontroller to operate properly without the use of ceramic resonator or crystal.

The passive components used were inexpensive, uncompensated devices: standard carbon resistors, ceramic or foil capacitors, and air core or iron core inductors. To provide reasonably clear data on the characteristics of the microcontroller itself, no attempt at compensation for the external components was made.

2.0 RC OSCILLATOR OPTION

With the RC oscillator option selected, the graphs in *Figures 1* through *6* indicate the variation of the instruction cycle time of the microcontroller with temperature and voltage. Typical R and C values, as recommended in the respective device data sheets, were used. The graphs are composite graphs reflecting the worst case variations of the devices tested. Therefore, the graphs show a percentage change of the instruction cycle time from a base or reference value. Where the results are plotted against voltage the reference is the value at $V_{CC} = 5V$. Where the results are plotted against temperature, the reference is the value at $T = 20^{\circ}C$. A positive percent variation indicates a longer instruction cycle time and therefore a slower oscillator frequency. Similarly, a negative percent variation indicates a shorter instruction cycle time and therefore a faster oscillator frequency.

The measurements were taken by holding the RESET pin of the device low and measuring the period of the waveform at pin SK. In this mode the SK period is the instruction cycle time. For divide by 4 the oscillator frequency is given by the following:

$$\text{frequency} = \frac{4}{\text{SK period}}$$

Measurements were taken at temperatures between $-40^{\circ}C$ and $+85^{\circ}C$ and at V_{CC} values between 4.5V and 9.5V. However, the reader must remember that the COP400 series is specified only between $0^{\circ}C$ and $+70^{\circ}C$. The reader must also remember that the COP420 is specified at V_{CC} levels between 4.5V and 6.3V only. The data here is usable for the COP300 series, which is specified at the extended temperature range of $-40^{\circ}C$ to $+85^{\circ}C$. However, the reader must keep in mind the generally more restricted V_{CC} range for some of the various COP300 series microcontrollers.

The graphs in *Figures 1* through *6* reflect the variation of the microcontroller only. The resistor and capacitor were not in the temperature chamber with the COPS device. Obviously, the results will be affected by the variation of the R and C with temperature. However, this can vary dramatically with the type of components used. The user will have to combine the data here with the characteristics of the external components used to determine what type of variation may be expected in his system.

3.0 CRYSTAL OR INVERTER OPTION

With the crystal or inverter option selected on the COPS microcontroller there is, effectively, an inverter between the CKI and CKO pins. CKI is the input to the inverter and CKO is the output. Various passive circuits were connected between CKI and CKO and the results documented. Of the operational circuits, a subset was tested over temperature with the microcontroller only in the temperature chamber. A smaller subset was tested over temperature with both the microcontroller and the oscillator network in the temperature chamber.

The data with the oscillator network in the temperature chamber is obviously highly dependent on the particular components used. This data was taken with standard, inexpensive, uncompensated components. Neither high precision nor high stability components were used. This data is included only to provide the user with some very general indication of how the oscillator frequency may vary with temperature in a real system.

3.1 COP420/COP402

Except for the ROM, the COP420 and COP402 are equivalent devices. The internal circuitry of each device is identical. Therefore, data taken for one of the devices is equally

applicable to the other. The following discussion will refer to the COP420 but all such references apply equally well to the COP402. Similarly, the graphs for the COP420 apply to the COP402 and *vice versa*.

With the crystal option selected, the COP420 oscillator circuitry will readily oscillate with almost any circuit configuration between CKI and CKO. What difficulty there is lies in finding the network of the device. With the appropriate divide option selected, oscillator frequencies between 800 kHz and 4 MHz are valid for the COP420. No data was taken for any network that produced an oscillation frequency outside the valid range.

3.1.1 L, LC, and RLC Networks

Various L, LC, and RLC networks were connected with varying results. Certain networks produced results much more stable than the RC networks; others were no better than the RC networks. With a single inductor connected between CKI and CKO, frequencies between 1 MHz and 4 MHz were easily obtained. However, the input gate capacitance at CKI (typically 5 pF to 10 pF) and the series resistance of the inductance become factors that impact the oscillation frequency and its stability over temperature.

The addition of a capacitor between CKI and ground tends to reduce the effects of the internal gate capacitance. For the single L, single C network of this type, the capacitor value should be greater than about 50 pF to begin to effectively swamp out the effects of the input gate capacitance. As might be expected, LC combinations which had their resonant frequencies within the valid COP420 frequency range produced the best results.

The addition of another capacitor(s) to the basic two-component LC network, as shown in *Figure III.1*, produced very good results. Varying the capacitor values in these networks — especially those capacitors between CKI and ground and CKO and ground — provided a great deal of control over the oscillation frequency. In *Figure III.1*, varying C1 from 25 pF to 0.01 μ F produced oscillation frequencies between about 3 MHz and 1.6 MHz (C2 = 25 pF, L = 56 μ H). In *Figure III.2*, with C1 = 330 pF, L = 56 μ H, and C2 = 27 pF, varying C3 between 10 pF and 0.003 μ F produced oscillation frequencies between about 2 MHz and 1.1 MHz. Varying C2 in *Figure III.3* produced a similar kind of control.

As the graphs indicate, various types of RLC networks were also tried. The range of possible usable circuits here is limited only by the user's imagination and his favorite type of RLC oscillator circuit. When their resonant frequency is

within the valid frequency range of the COP420, LC and RLC networks can be a very effective substitute for a crystal. The only potential problem is that a good RLC, or even LC, oscillator circuit may not be a cost-effective substitute for a crystal in a COP420 system. The user will have to make that determination.

3.2 COP420L

The valid input frequency range for the COP420L, with the appropriate divide option selected, is between 200 kHz and 2.097 MHz. With the crystal option selected the COP420L oscillated much less readily than the COP420.

The LC networks gave outstanding results with the COP420L. With the simple two-component LC network shown in the graphs, holding C at 50 pF and varying L from 200 μ H to 700 μ H gave oscillation frequencies from about 2 MHz to 1 MHz. Holding L at 390 μ H and varying C from 10 pF to 700 pF gave oscillation frequencies of about 2 MHz to 1.6 MHz. Similar results were obtained when a capacitor was placed in parallel with the inductance.

3.3 COP410L

The COP410L has a valid input frequency range of 200 kHz to 530 kHz.

The LC networks also gave very good results. With the simple LC network shown in the graphs, holding L at 4700 μ H and varying C from 25 pF to 0.003 μ F gave oscillation frequencies of about 460 kHz to 225 kHz.

3.4 GENERAL NOTES

With the crystal or inverter option selected on COPS microcontrollers, a wide variety of networks may be used in place of the ceramic resonator or crystal.

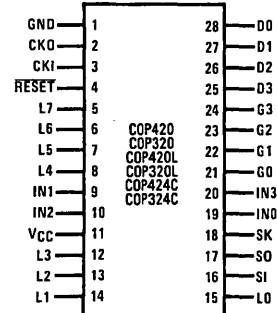
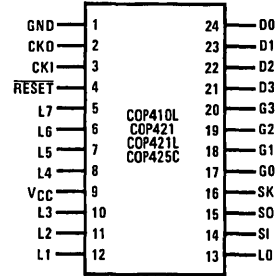
LC and RLC networks can be used in any of the devices. Appropriately designed, these networks will provide a stable oscillation frequency for the microcontroller. The user will have to allow for the variation of the external components with temperature when using these networks. The problems with networks such as these is that they may not be cost-effective alternatives to the crystal or resonator, especially if high stability, temperature compensated components are used. The user will have to make the determination of cost-effectiveness.

A final note is that all of these networks place a load on the CKO output. If the signal from CKO is needed elsewhere in the system and a circuit similar to one of those discussed in this document is used, it will probably be necessary to buffer the CKO output.

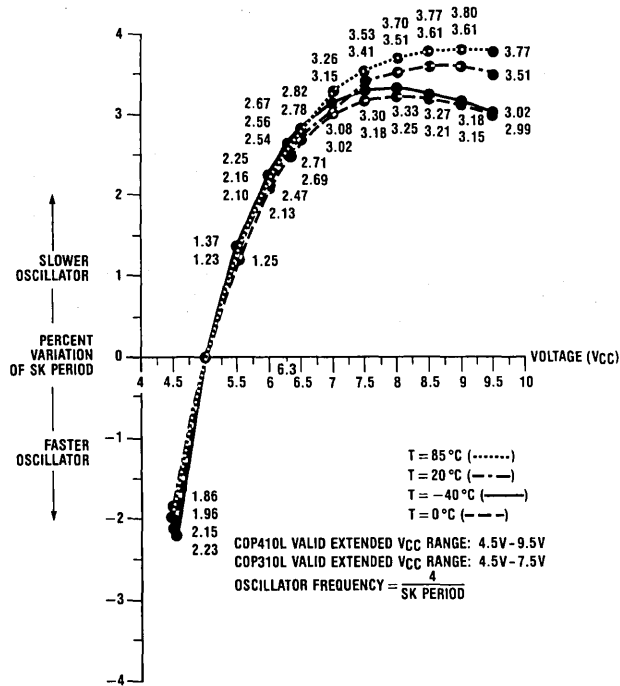
4.0 Conclusion

The networks described are generally simple and inexpensive and have all been observed to be functional. The results obtained provide greater flexibility in the oscillator selection in a COPs system and gives the user some general indication as to what may be expected with the various circuits described.

COP Microcontroller Pinouts



TL/DD/6938-1



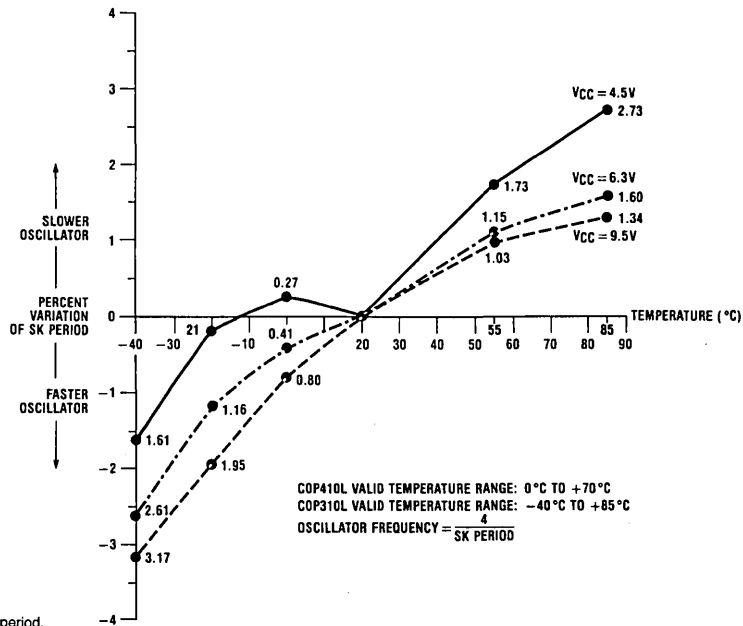
TL/DD/6938-2

Note 1: Base period at VCC = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

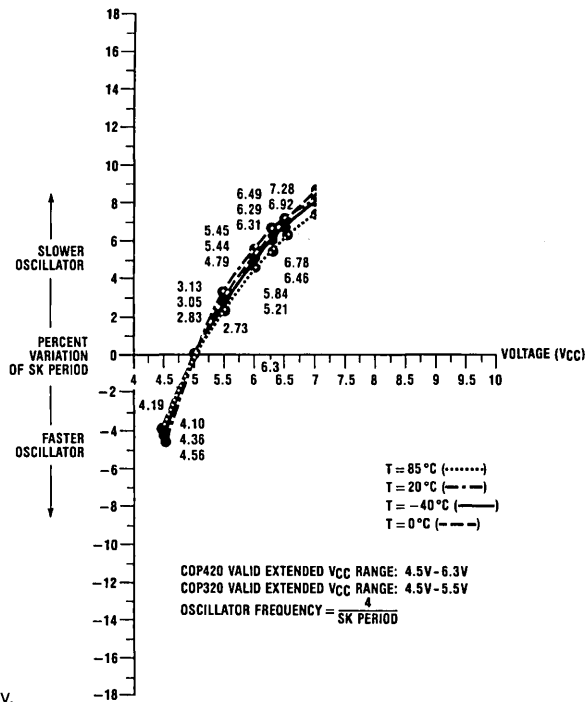
FIGURE 1. COP310L/COP410L RC Oscillator Variation with VCC



- Note 1: 20°C = base period.
- Note 2: Device variation only. Graph does not include RC variation with temperature.
- Note 3: SK period = instruction cycle time.

TL/DD/6938-3

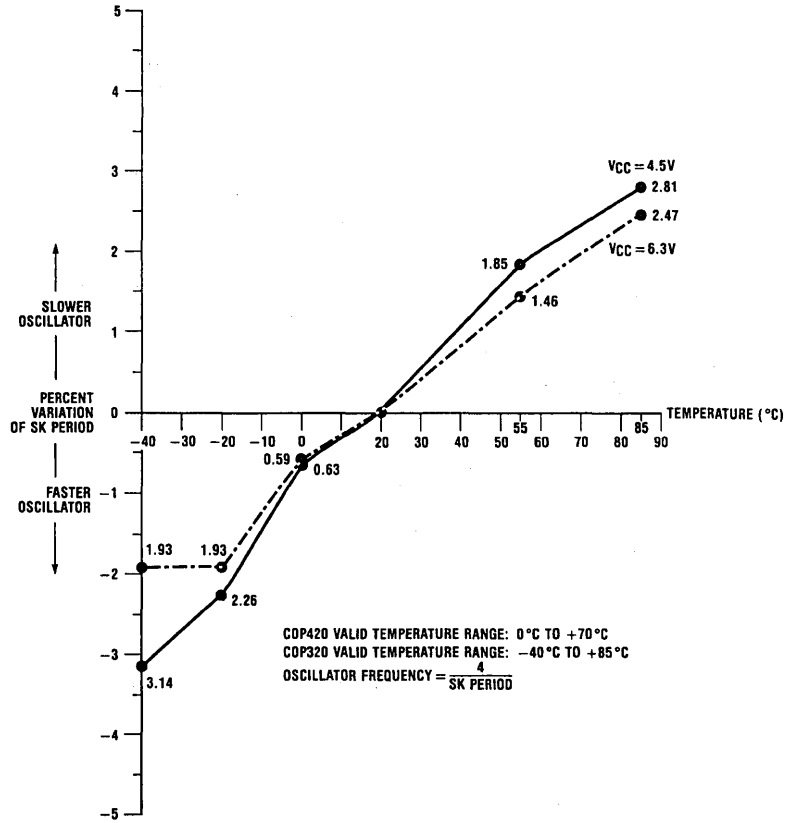
FIGURE 2. COP310L/COP410L RC Oscillator Variation with Temperature



- Note 1: Base period at VCC = 5.0V.
- Note 2: Device variation only. Graph does not include RC variation with temperature.
- Note 3: SK period = instruction cycle time.

TL/DD/6938-4

FIGURE 3. COP320/COP420 RC Oscillator Variation with VCC



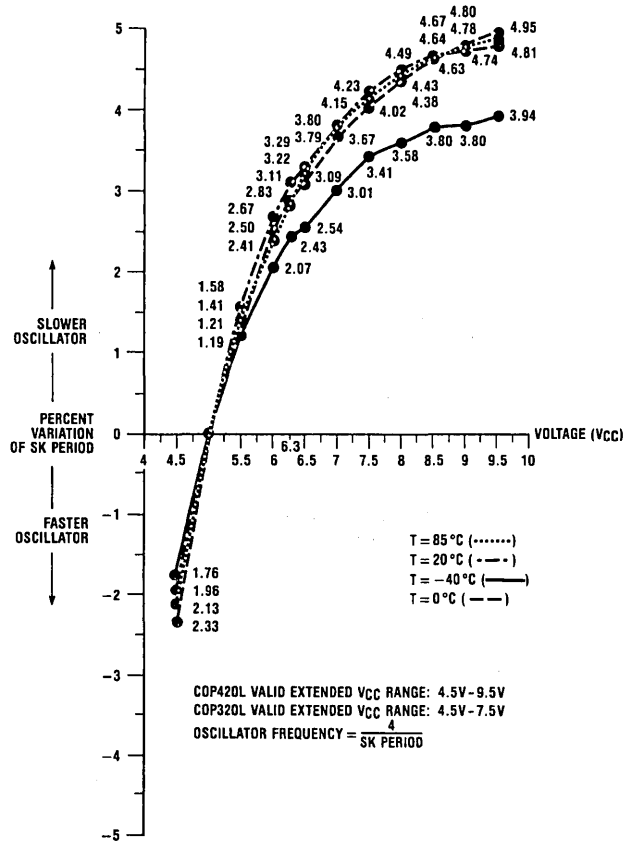
Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

TL/DD/6938-5

FIGURE 4. COP320/COP420 RC Oscillator Variation with Temperature



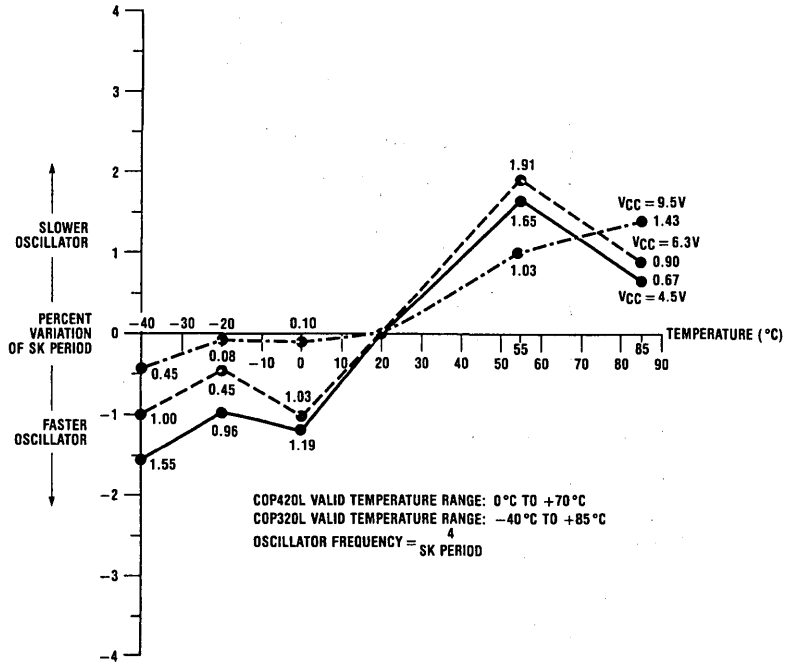
Note 1: Base period at V_{CC} = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 5. COP320L/COP420L RC Oscillator Variation with V_{CC}

TL/DD/6938-6



TL/DD/6938-7

Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 6. COP320L/COP420L RC Oscillator Variation with Temperature

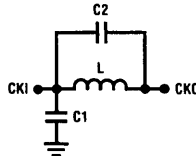


FIGURE III.1

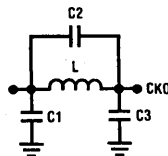


FIGURE III.2

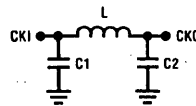
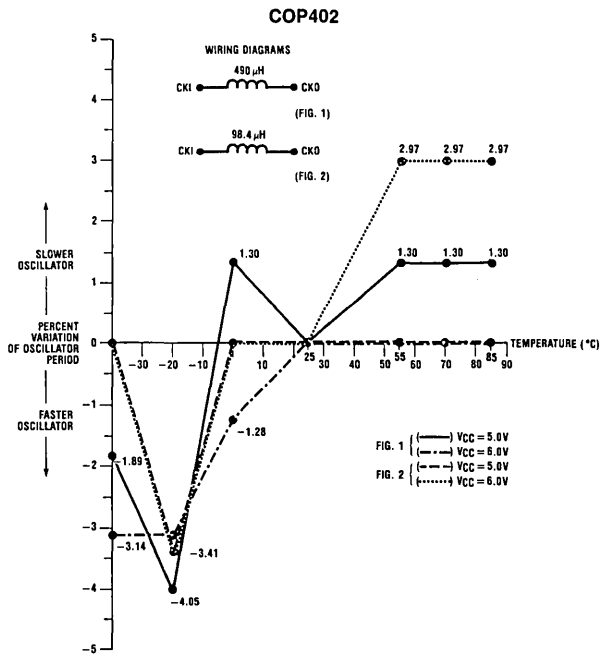


FIGURE III.3

TL/DD/6938-8

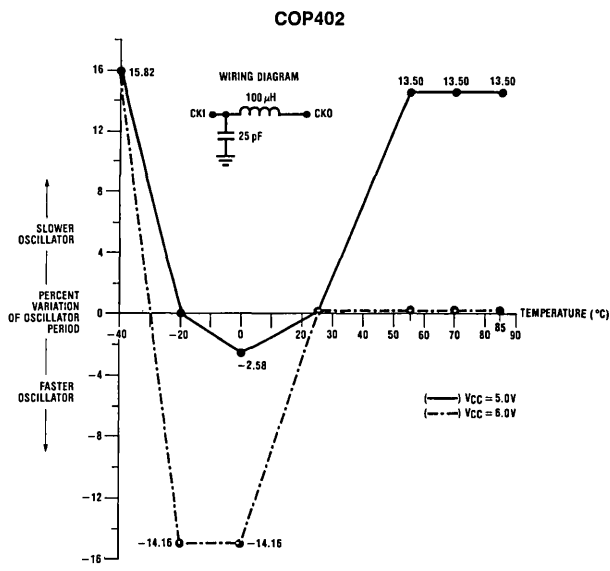


Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include "L" variation with temperature.

TL/DD/6938-9

FIGURE 7



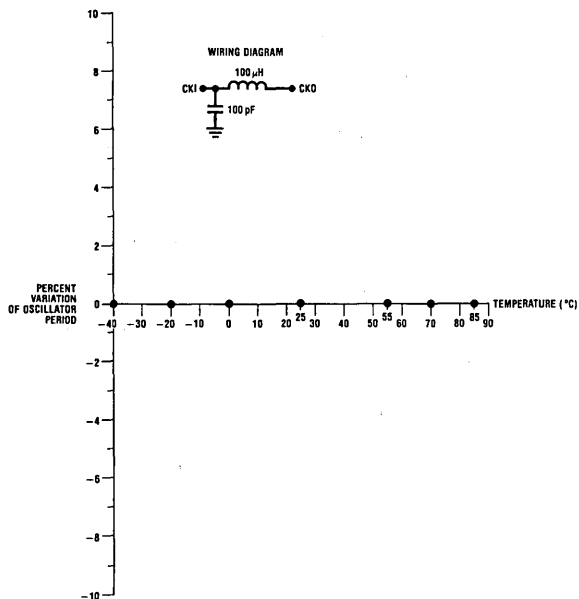
Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

TL/DD/6938-10

FIGURE 8

COP420



TL/DD/6938-11

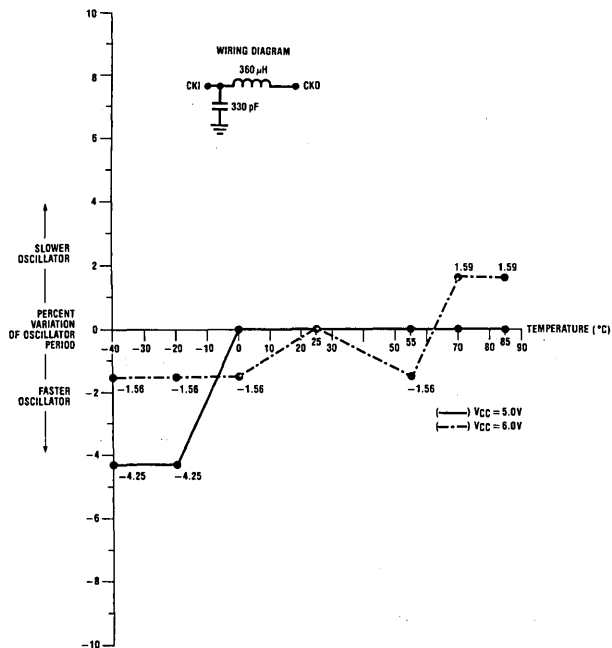
Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

No measurable variation over temperature.

FIGURE 9

COP420

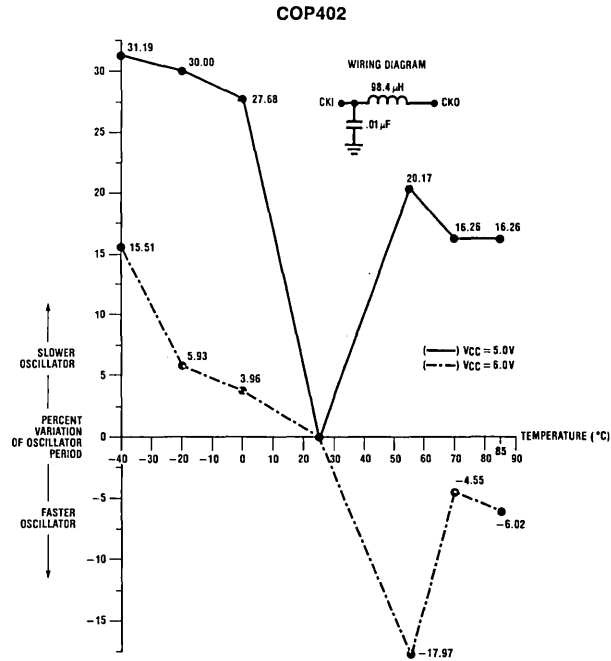


TL/DD/6938-12

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 10

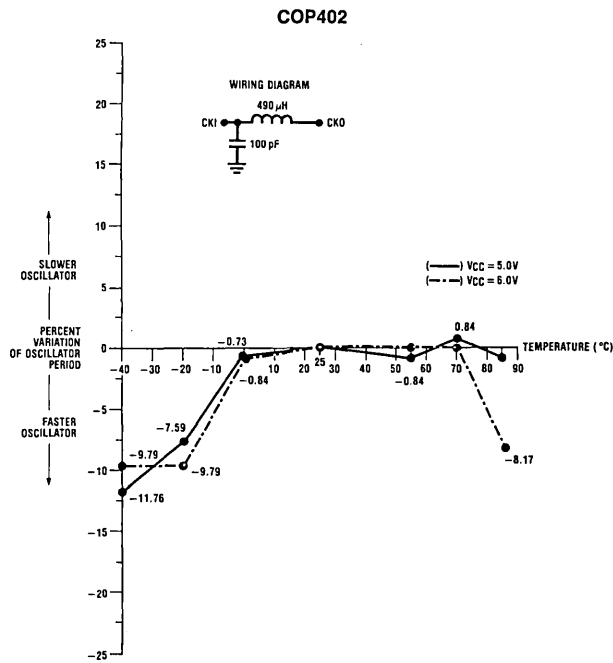


TL/DD/6938-13

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 11



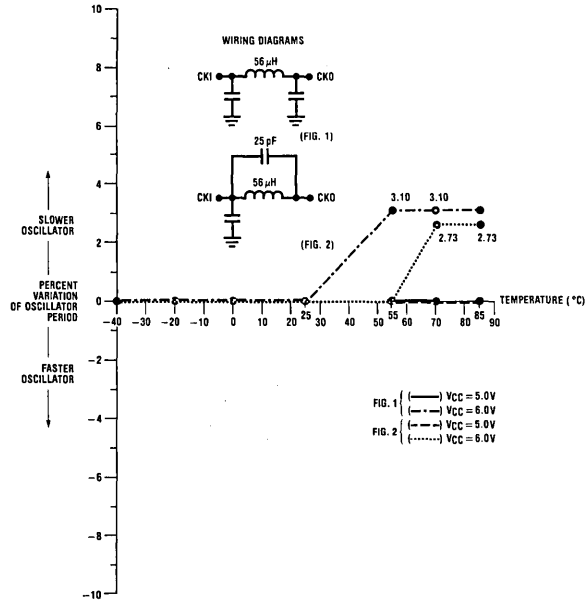
TL/DD/6938-14

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 12

COP402



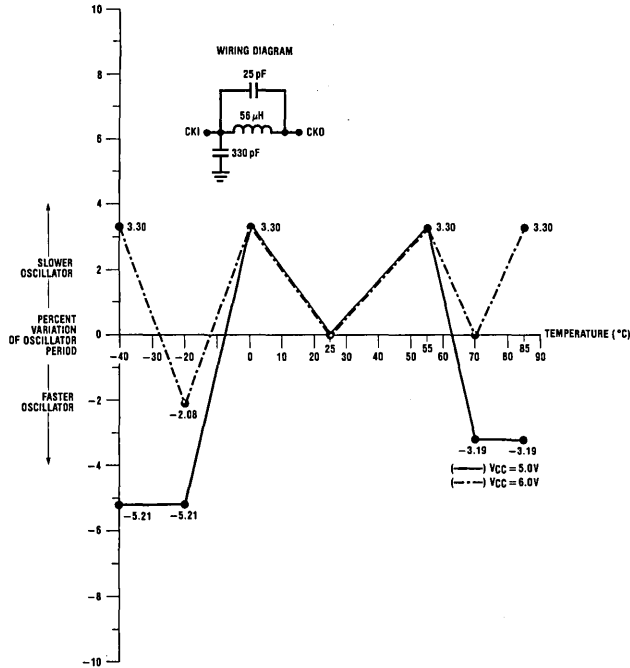
Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

TL/DD/6938-15

FIGURE 13

COP402

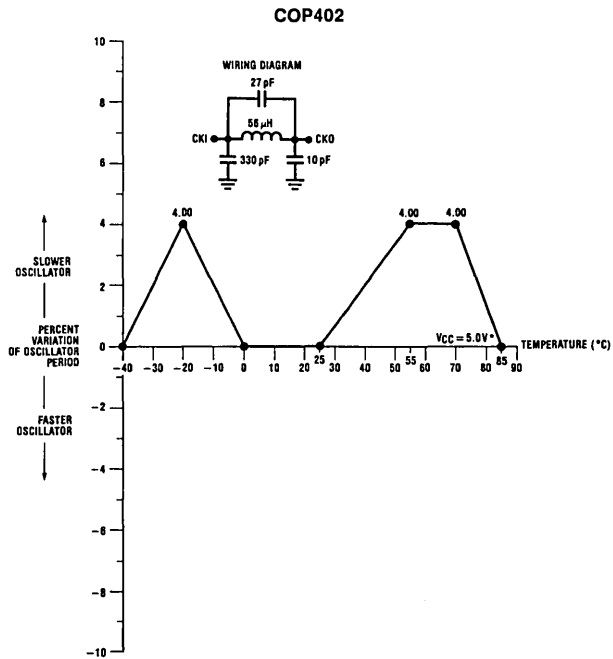


Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

TL/DD/6938-16

FIGURE 14

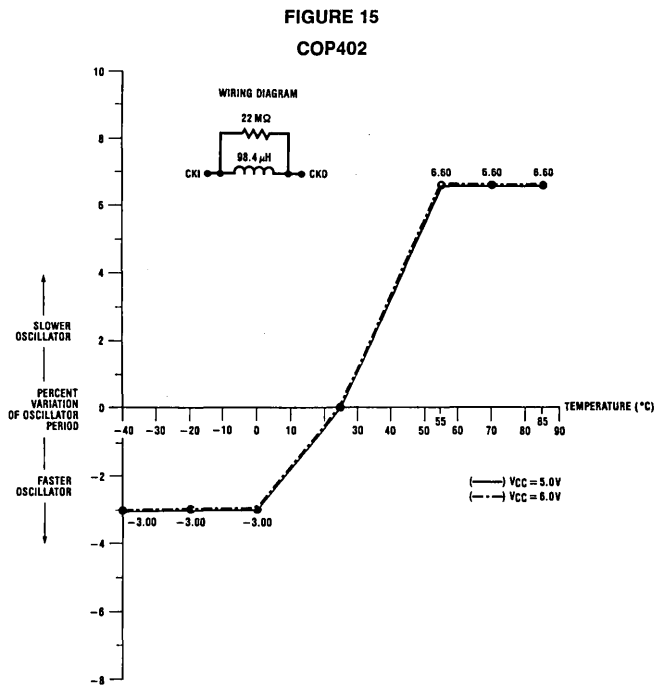


TL/DD/6938-17

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

*No variation at 6V.

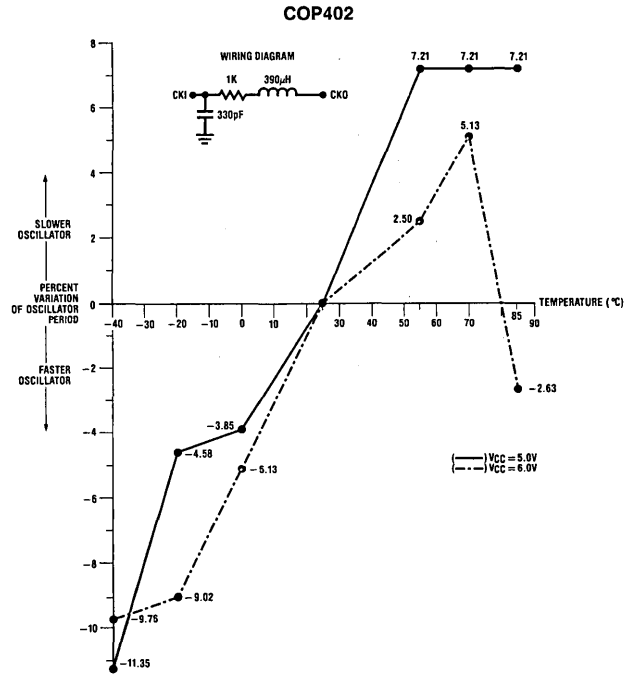


TL/DD/6938-18

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RL variation with temperature.

FIGURE 16

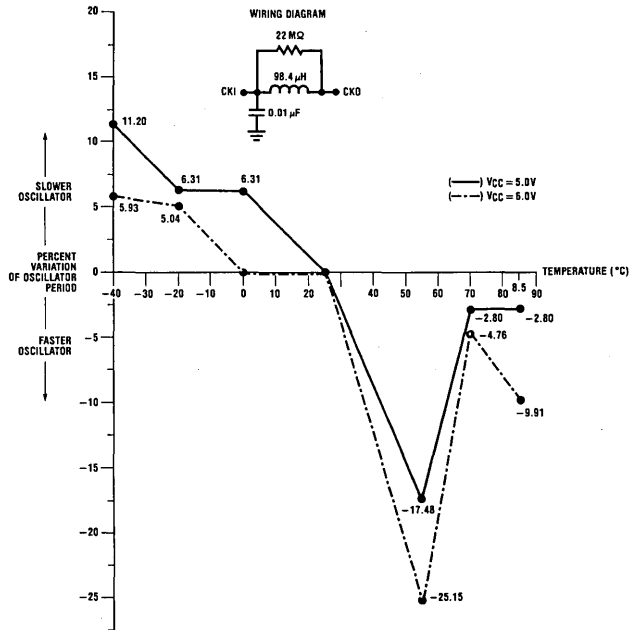


Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

TL/DD/6938-19

FIGURE 17
COP402

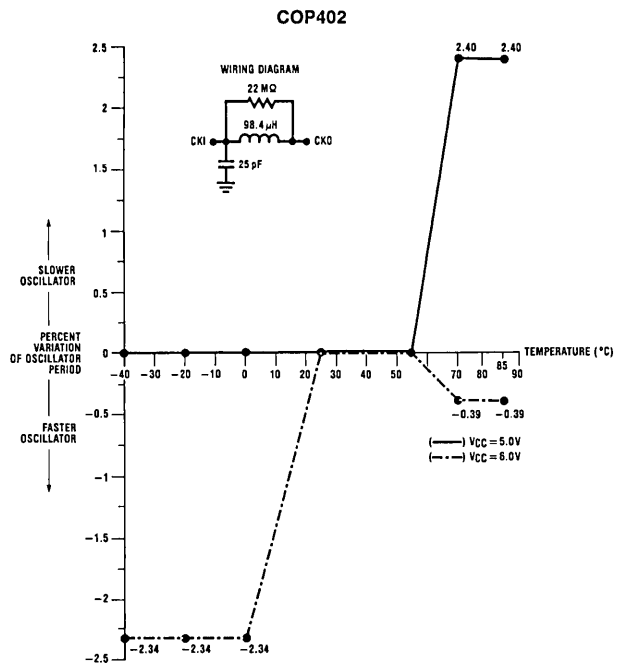


Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

TL/DD/6938-20

FIGURE 18

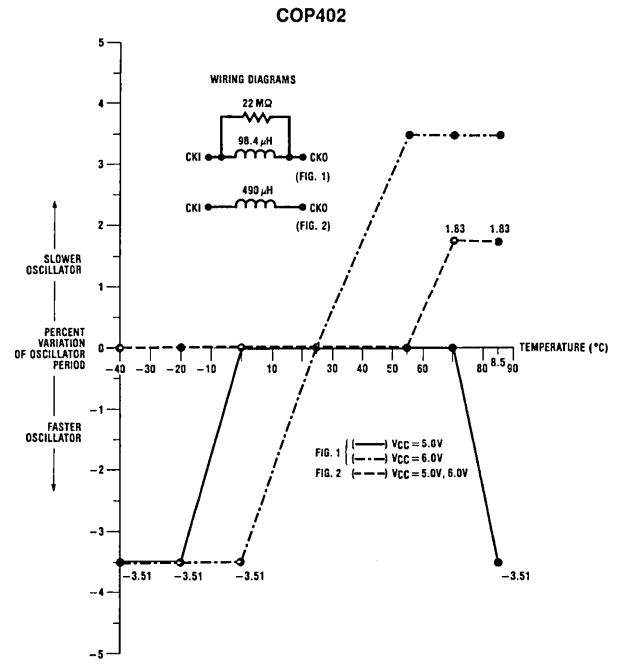


TL/DD/6938-21

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 19

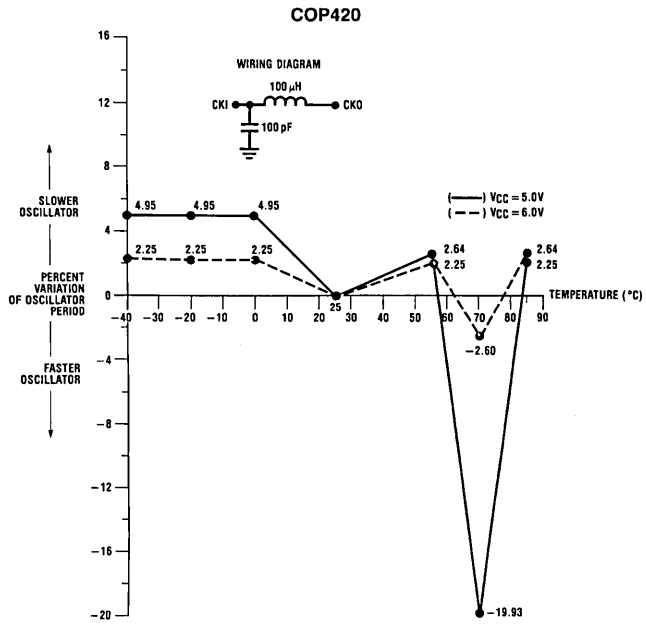


TL/DD/6938-22

Note 1: 25°C = base period.

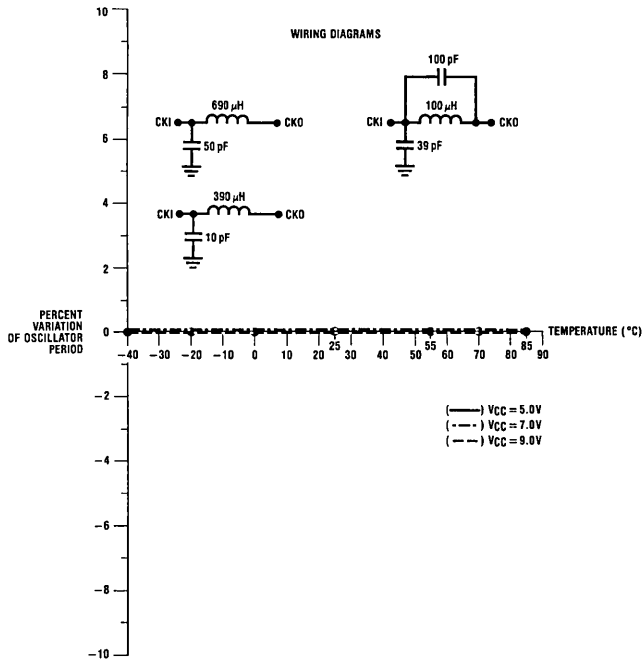
Note 2: RL in oven with COP402.

FIGURE 20



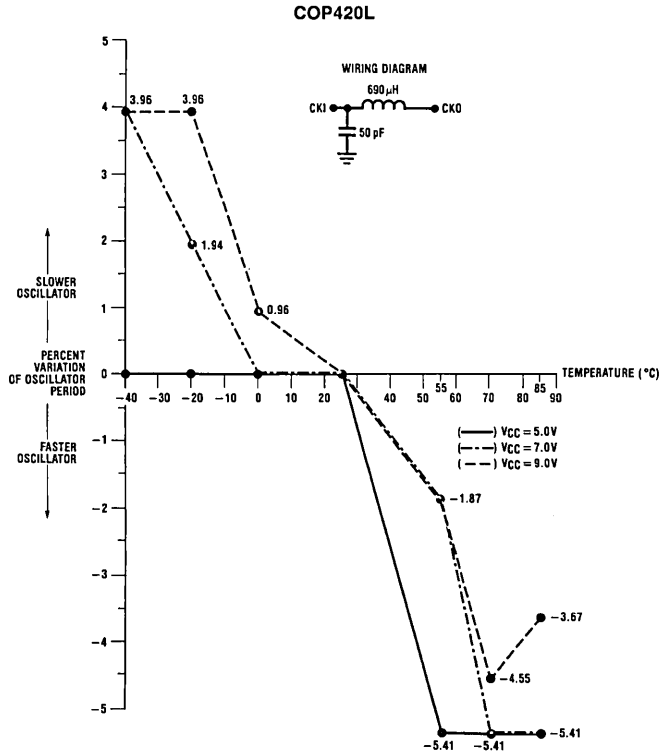
Note 1: 25°C = base period.
Note 2: LC in oven with COP402.

FIGURE 21
COP420L



Note 1: No measurable variation for all three circuits above.
Note 2: 25°C = base period.
Note 3: Device variation only. Graph does not include LC variation with temperature.

FIGURE 22

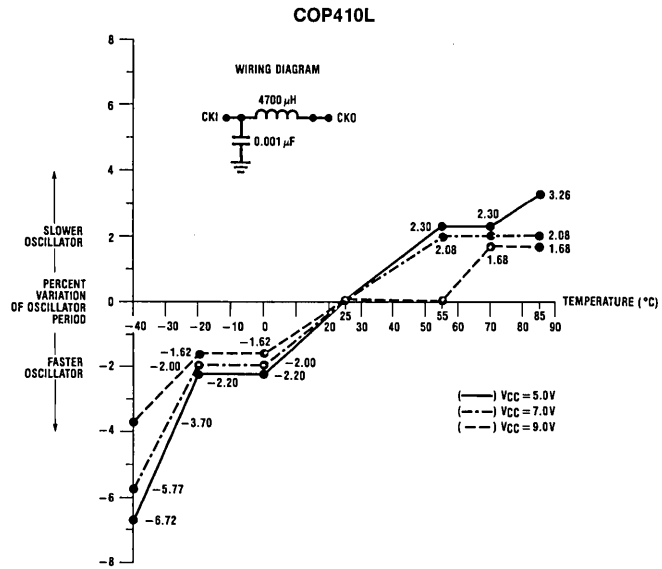


TL/DD/6938-28

Note 1: 25°C = base period.

Note 2: LC in oven with COP420L.

FIGURE 23

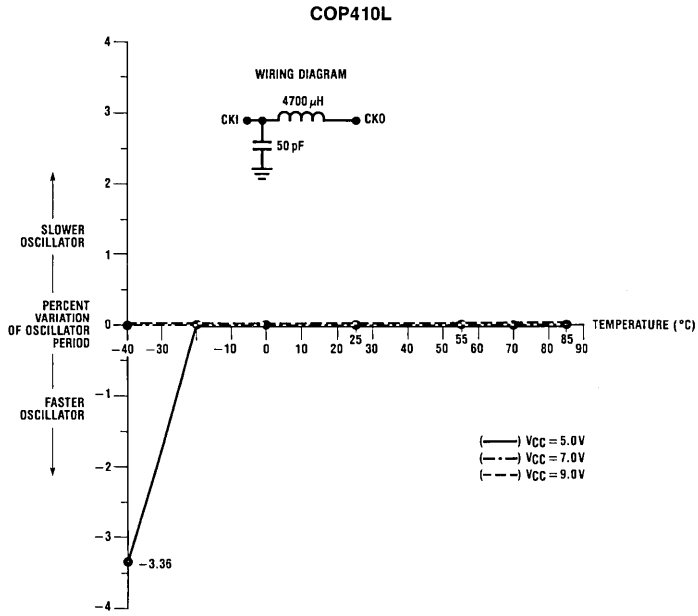


TL/DD/6938-29

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 24



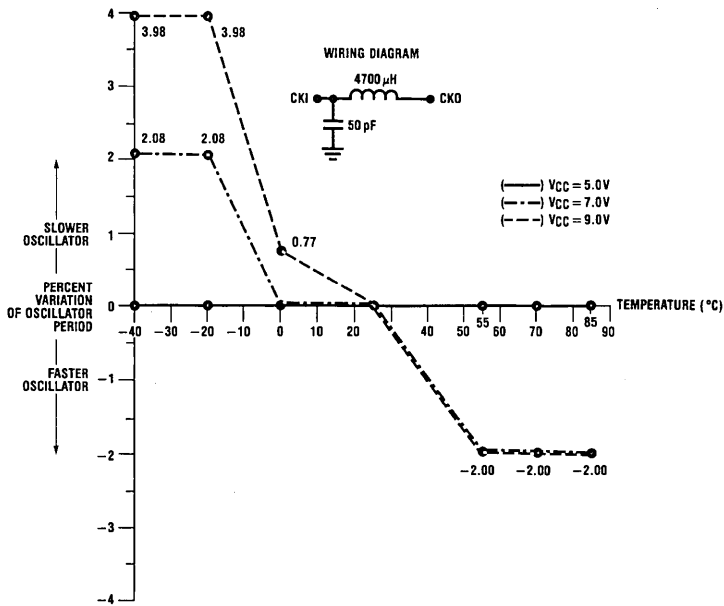
TL/DD/6938-30

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 25

COP410L



TL/DD/6938-32

Note 1: 25°C = base period.

Note 2: LC in oven with COP410L.

FIGURE 26

Triac Control Using the COP400 Microcontroller Family

National Semiconductor
COP Note 6



COP NOTE 6

Table of Contents

1.0 TRIAC CONTROL

- 1.1 Basic Triac Operation
- 1.2 Triggering
- 1.3 Zero Voltage Detection
- 1.4 Direct Couple
- 1.5 Pulse Transformer Interface
- 1.6 False Turn-on

2.0 SOFTWARE TECHNIQUES

- 2.1 Zero Voltage Detection
- 2.2 Processing Time Allocations
 - Half Cycle Approach
 - Full Cycle Approach
- 2.3 Steady State Triggering

3.0 TRIAC LIGHT INTENSITY CONTROL CODE

- 3.1 Triac Light Intensity Routine

1.0 Triac Control

The COP400 single-chip controller family members provide computational ability and speed which is more than adequate to intelligently manage power control. These controllers provide digital control while low cost and short turn-around enhance COPST[™] desirability. The COPS controllers are capable of 4 μ s cycle times which can provide more than adequate computational ability when controlling 60 Hz line voltage. Input and output options available on the COPS devices can contour the device to apply in many electrical situations. A more detailed description of COPS qualifications is available in the COP400 data sheets.

The COPS controller family may be utilized to manage power in many ways. This paper is devoted to the investigation of low cost triac interfaces with the COP400 family microcontroller and software techniques for power control applications.

1.1 BASIC TRIAC OPERATION

A triac is basically a bidirectional switch which can be used to control AC power. In the high-impedance state, the triac blocks the principal voltage across the main terminals. By pulsing the gate or applying a steady state gate signal, the triac may be triggered into a low impedance state where conduction across the main terminals will occur. The gate signal polarity need not follow the main terminal polarity; however, this does affect the gate current requirements. Gate current requirements vary depending on the direction of the main terminal current and the gate current. The four trigger modes are illustrated in *Figure 1*.

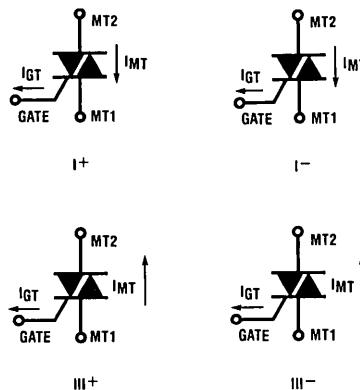


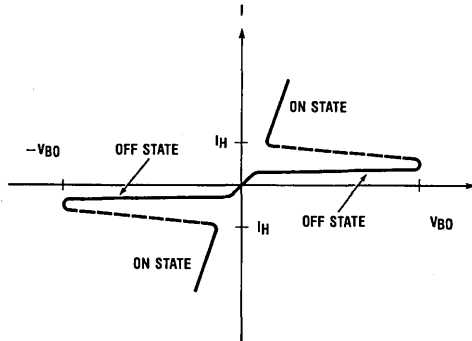
FIGURE 1. Gate Trigger Modes. Polarities Referenced to Main Terminal 1.

TL/DD/6939-1

3

The breakover voltage (V_{BO}) is specified with the gate current (I_{GT}) equal to zero. By increasing the gate current supplied to the triac, V_{BO} can be reduced to cause the triac to go into the conduction or on state. Once the triac has entered the on state the gate signal need not be present to sustain conduction. The triac will turn itself off when the main terminal current falls below the minimum holding current required to sustain conduction (I_H).

A typical current and voltage characteristic curve is given in *Figure 2*. As can be seen, when the gate voltage and the main terminal 2 (MT2) voltages are positive with respect to MT1 the triac will operate in quadrant 1. In this case the trigger circuit sources current to the triac (I+ MODE).



TL/DD/6939-2

FIGURE 2. Voltage-Current Characteristics

After conduction occurs the main terminal current is independent of the gate current; however, due to the structure of the triac the gate trigger current is dependent on the direction of the main terminal current. The gate current requirements vary from mode to mode. In general, a triac is more easily triggered when the gate current is in the same direction as the main terminal current. This can be illustrated in the situation where there is not sufficient gate drive to cause conduction when MT2 is both positive and negative. In this case the triac may act as a single direction SCR and conduction occurs in only one direction. The trigger circuit must be designed to provide trigger currents for the worst case trigger situation. Another reason ample trigger current must be supplied is to prevent localized heating within the pellet and speed up turn-on time. If the triac is barely triggered only a small portion of the junction will begin to conduct, thus causing localized heating and slower turn-on. If an insufficient gate pulse is applied damage to the triac may result.

1.2 TRIGGERING

Gate triggering signals should exceed the minimum rated trigger requirements as specified by the manufacturer. This is essential to guarantee rapid turn-on time and consistent operation from device to device.

Triac turn-on time is primarily dependent on the magnitude of the applied gate signal. To obtain decreased turn-on times a sufficiently large gate signal should be applied. Faster turn-on time eliminates localized heat spots within the pellet structure and increases triac dependability.

Digital logic circuits, without large buffers, may not have the drive capabilities to efficiently turn on a triac. To insure proper operation in all firing situations, external trigger circuitry might become necessary. Also, to prevent noise from disturbing the logic levels, AC/DC isolation or coupling techniques must be utilized. Sensitive gate triacs which require minimal gate input signal and provide a limited amount of main terminal current may be driven directly. This paper will focus on 120V_{AC} applications of power control.

1.3 ZERO VOLTAGE DETECTION

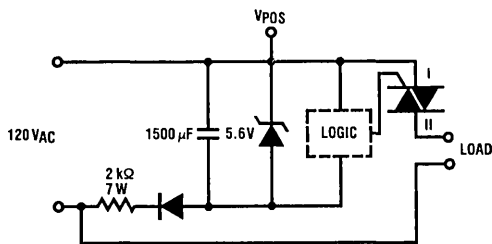
In many applications it is advantageous to switch power at the AC line zero voltage crossing. In doing this, the device being controlled is not subjected to inherent AC transients. By utilizing this technique, greater dependability can be obtained from the switching device and the device being switched. It is also sometimes desirable to reference an event on a cyclic basis corresponding to the AC line frequency. Depending on the load characteristics, switching times need to be chosen carefully to insure optimal performance. Triac controlled AC switching referenced to the AC 60 Hz line frequency enables precise control over the conduction angle at which the triac is fired. This enables the COPS device to control the power output by increasing or decreasing the conduction angle in each half cycle.

A wide variety of zero voltage detection circuits are available in various levels of sophistication. COPS devices, in most cases, can compensate for noisy or semi-accurate ZVD circuits. This compensation is utilized in the form of debounce and delay routines. If a noisy transition occurs near zero volts the COPS device can wait for a valid transition period specified by the maximum amount of noise present. Some software considerations are presented in the software section and are commented upon. The minimal detection circuit is shown in *Figure 9*.

1.4 DIRECT COUPLE

Isolation associated problems can be overcome by means of direct AC coupling. One such method is illustrated in *Figure 3*. This circuit incorporates a half-wave rectifier in conjunction with a filter capacitor to provide the logic power supply. The positive half-cycle is allowed to drop across the zener diode and be filtered by the capacitor. This creates a low cost line interface; however, only a limited supply current is available. In order to control the current capabilities of this circuit the series resistor must be modified. However, as more current is required, the power that must be dissipated in the series resistor increases. This increases the power dissipation requirements of the series resistor and the system cost. For applications which require large current sources an alternative method is advisable. In order to assure consistent operation, power supply ripple must be mini-

mized. COPS devices can be operated over a relatively wide power supply range. However, excessive ripple may cause an inadvertent reset operation of the device.

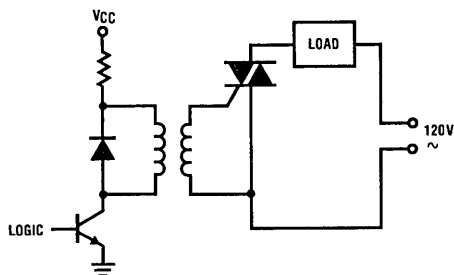


TL/DD/6939-3

FIGURE 3. AC Direct Couple

1.5 PULSE TRANSFORMER INTERFACE

Digital logic control of triacs is easily accomplished by triggering through pulse transformers or optical coupling. The energy step-up gained by using a pulse transformer should provide a more than adequate gate trigger signal. This complies with manufacturers' suggested gate signal requirements. Pulse transformers also provide AC/DC isolation necessary in control logic interfaces. Minimal circuit interface to the pulse transformer is required as shown in Figure 4. Optical coupling circuits provide isolation, and in some cases adequate gate drive capabilities.



TL/DD/6939-4

FIGURE 4. Pulse Transformer Interface

A logic controlled pulse is applied to the base of the transistor to switch current through the primary of the pulse transformer. The transformer then transfers the signal to the secondary and causes the triac to fire. The energy transfer that is now available on the secondary is more than adequate to turn on the triac in any of its operating modes. When the pulse transformer is switched off a reverse EMF is generated in the primary coil which may cause damage to the transistor. The diode across the primary serves to protect the collector junction of the switching transistor. Another major advantage is AC isolation; the gate of the triac is now completely isolated from the logic portion of the circuit.

1.6 FALSE TURN-ON

When switching an inductive load, voltage spikes may be generated across the main terminals of the triac which have

the potential of a non-gated turn-on of the triac. This creates the undesirable situation of limited control of the system. In a system with an inductive load the voltage leads the current by a phase shift corresponding to the amount of inductance in the motor. As the current passes near zero, the voltage is at a non-zero value, offset due to the phase shift. When the principal current through the triac pellet decreases to a value not capable of sustaining conduction the triac will turn off. At this point in time the voltage across the terminals will instantaneously attain a value corresponding to the phase shift caused by the inductive load. The rapid decay of current in the inductor causes an $L di/dt$ voltage applied across the terminals of the triac. Should this voltage exceed the blocking voltage specified for the triac, a false turn-on will occur.

In order to avoid false turn-on, a snubber network must be added across the terminals to absorb the excess energy generated by this situation. A common form of this network is a simple RC in series across the terminals. In order to select the values of the network it is necessary to determine the peak voltage allowable in the system and the maximum dV/dt stress the triac can withstand. One approach to obtaining the optimal values for R_S and C_S is to model the effective circuit and solve for the triac voltage. The snubber in conjunction with the load can now be modeled as an RLC network. Due to the two storage elements (L motor, C snubber) a second order differential equation is generated. Rather than approach this problem from a computer standpoint it becomes much easier to obtain design curves generated for rapid solution of this problem. These design curves are available in many triac publications. (For instance, see RCA application note AN 4745.)

2.0 Software Techniques

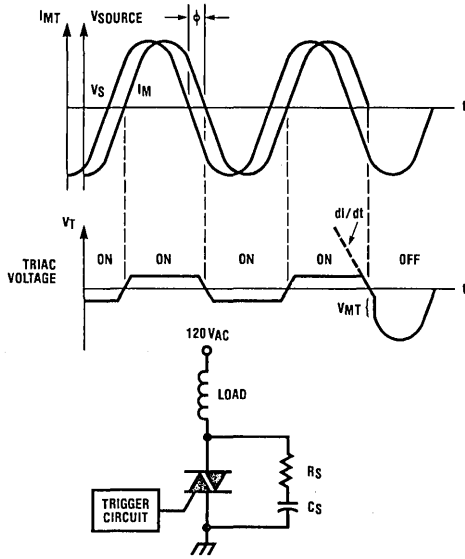
2.1 ZERO VOLTAGE DETECTION

In order to intelligently control triacs on a cyclic basis, an accurate time base must be defined. This may be in the form of an AC, 60 Hz sync pulse generated by a zero voltage detection circuit or a simple real time clock. The COP400 series microcontrollers are suited to accommodate either of these time base schemes while accomplishing auxiliary tasks.

Zero voltage detection is the most useful scheme in AC power control because it affords a real time clock base as well as a reference point in the AC waveform. With this information it is possible to minimize RFI by initiating power-on operations near the AC line voltage zero crossing. It is also possible to fire the triac for only a portion of the cycle, thus utilizing conduction angle manipulation. This is useful in both motor control and light intensity control.

Sophisticated zero voltage detection circuits which are capable of discriminating against noise and switch precisely at zero crossing are not necessary when used in conjunction with a COPS device. COPS software is capable of compensating for noisy or semi-accurate zero voltage detection circuits. This can be accomplished by introducing delays and debounce techniques in the software routines. With a given reference point in the AC waveform it now becomes easy

to divide the waveform to efficiently allocate processing time. These techniques are illustrated in the code listing at the end of this paper.



TL/DD/6939-5

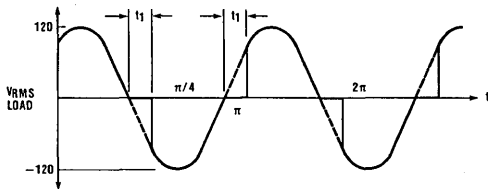
FIGURE 5. Current Lag Caused by Inductive Load, Snubber Circuit

2.2 PROCESSING TIME ALLOCATIONS

Half Cycle Approach

In order to accomplish more than triac timing, dead delay time must be turned into computation time. It appears that the controller is occupied totally by time delays, which leaves a very limited amount of additional control capability. There are, however, many ways to accomplish auxiliary tasks simultaneously.

On each half cycle an initial delay is incorporated to space into the cycle. This dead time may be put to use and very little voltage to the load is sacrificed. For example, if the load is switched on at $\pi/4$ RAD, the maximum applied RMS voltage to the load is $114V_{RMS}$ (assuming $V_{SUPPLY} = 120V_{RMS}$). This is illustrated in the figure below.



TL/DD/6939-6

FIGURE 6. Full Cycle Approach

If a delay of $\pi/4$ RAD (45 degrees) is inserted after each zero crossing detection the RMS voltage to the load can be determined in the following manner:

$$V_{LOAD} = \sqrt{\frac{(120\sqrt{2})^2}{(2)\pi} (2) \int_{\pi/4}^{\pi} \sin^2(a) da}$$

$$V_{LOAD} = \sqrt{\frac{(120\sqrt{2})^2}{(2)\pi} (2) (1.428)}$$

$$V_{LOAD} = 114.4 V_{RMS}$$

$$\pi/4 \text{ RAD} = 45 \text{ degrees} \quad @60 \text{ Hz} \quad t = 2.08 \text{ ms}$$

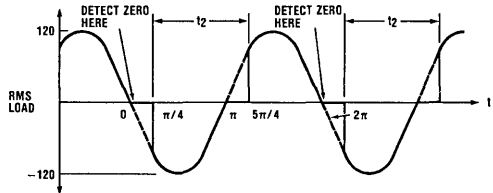
As can be seen the dead time on each half cycle can be 2.08 ms and the load will still see $114.4 V_{RMS}$ of a V_{SUPPLY} of $120 V_{RMS}$. If this approach is implemented the initial delay of 2.08 ms can be used as computation time. The number of instructions which can be executed when operating at $4 \mu s$ instruction cycle time is:

$$2.08 \text{ ms} / 4 \mu s = 520 \text{ instructions}$$

(130 instructions at $16 \mu s$ cycle time)

Full Cycle Approach

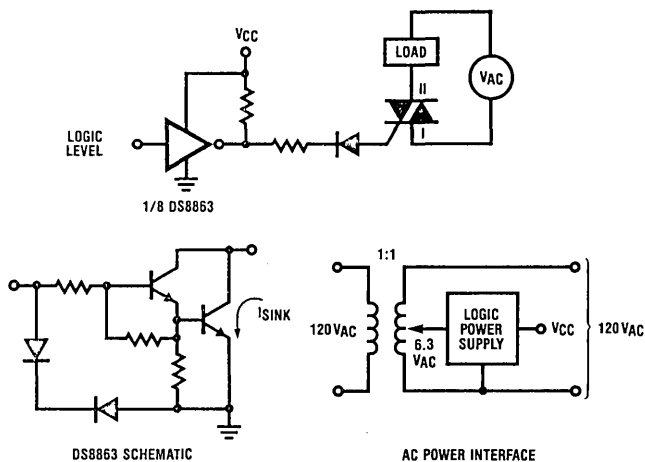
The methods of half cycle and full cycle triggering are very similar in procedure. The main difference is that all timing is referenced from only one (of the two) zero voltage detection transition in each full AC cycle. For most all applications, when varying the conduction angle it is desirable to fire at the same conduction angle each half cycle to maintain a symmetric applied voltage. In order to accomplish this the triac may be fired twice from one reference point. When applying this technique an 8.33 ms delay must be executed to maintain the symmetric applied voltage. This approach provides the most auxiliary computation time in that the 8.33 ms delay may be turned into computational time. The basic flow for this technique is illustrated below.



TL/DD/6939-7

FIGURE 7. Full Cycle Approach

In the above example the zero crossing pulse is debounced on the one-to-zero transition, thus marking the beginning of a full cycle. Once this transition has been detected, an ini-



TL/DD/6939-8

FIGURE 8. Steady State Triggering

tial delay of $\pi/4$ RAD is incorporated and the triac is fired. At this time exactly 8.33 ms is available until the triac need be triggered again. This will provide a symmetric voltage to the load only if the delay is 8.33 ms. During this period the number of instructions which can be executed when operating at $4 \mu\text{s}$ is:

$$8.33 \text{ ms} / 4 \mu\text{s} = 2082$$

(520 instructions at $16 \mu\text{s}$)

An alternative approach may be to take the burden from the COPS device by using peripheral devices such as static display controllers, external latches, etc.

2.3 STEADY STATE TRIGGERING

It is possible to trigger a triac with a steady state logic level. This is accomplished by allowing the triac gate to sink or source current during the desired on-time. When utilizing this method it becomes easier to trigger the triac and leave it on for many cycles without having to execute code to retrigger. This approach is advantageous when the triac must be fired is for relatively long periods and conduction angle firing is not desired, thus more time is available to accomplish auxiliary tasks. A steady state on or off signal and external circuitry can accomplish triac firing and free the processor for other tasks. If it is desired to use a pulse

transformer, an external oscillator must be gated to the triac to provide the trigger signal. A pulse train of 10 to 15 kHz is adequate to fire the triac each half cycle. This calls for external components and is relatively costly. If isolation associated problems can be tolerated or overcome (dual power supply transformers, direct AC coupling, etc.), a simple buffer may be utilized in triggering the triac. This method is illustrated in *Figure 8*. The National Semiconductor DS8863 display driver is capable of steady state firing of the triac. National offers many buffers capable of driving several hundred milliamps, which are suitable for driving triacs. On the market today there are many suppliers of sensitive gate triacs which may be triggered directly from a COPS device or in conjunction with a smaller external buffer.

The DS8863 display driver is capable of sinking up to 500 mA, which is adequate to drive a standard triac. In the off state the driver will not sink current. When a logic "1" is applied to the input the device will turn on. Keeping the device off (output "1") will prevent the triac from turning on because the buffer does not have the capability of sourcing current. A series resistor limits the current from the triac gate and the diode isolates the negative spikes from the gate. Since the drive circuit will only sink current in this configuration, the triac will be operating in the I- and III- modes.

3.0 Triac Light Intensity Control Code

The following code is not intended to be a final functional program. In order to utilize this program, modifications must be made to specialize the routines. This is intended to illustrate the method and is void of control code to command a response such as intensify or deintensify. The control is up to the user and full understanding of the program must be attained before modifications can be implemented.

This program is a general purpose light intensifying routine which may be modified to suit light dimmer applications. The delay routines require a $4.469 \mu\text{s}$ cycle time which can be attained with a 3.578 MHz crystal (CKI/16 option). This program divides the half cycle of a 60 Hz power line into 16 levels. Intensity is varied by increasing or decreasing the conduction angle by firing the triac at various levels. The program will increase the conduction angle to a maximum specified intensity in a fixed amount of time. The time required to intensify to the maximum level is dependent on the number of fire-times per level that is specified (FINO). This code illustrates a half cycle approach and relies on the parameters specified by the programmer in the control selection.

Zero crossings of the 60 Hz line are detected and software debounced to initiate each half cycle; thus the triac is serviced on every half cycle of the power line. A level/sublevel approach is utilized to vary the conduction angle and provide a prolonged intensifying period. The maximum intensity is specified by the "LEVEL" RAM location and time required to get to that level is specified by the "FINO" RAM location.

Once a level has been specified, the remaining time in the half cycle is then divided into sublevels. The sublevels are increased in steps to the maximum level. The "FINO" RAM location contains the number of times that the triac will be fired per sublevel, thus creating the intensity time base. There are 15 valid sublevels and up to 15 fire-times per sublevel. Both these parameters may be increased to provide better resolution and longer intensify periods. To make the triac de-intensity (dim) the sublevels need only to be decremented rather than incremented. If this is done, the conduction angle will start out at the maximum level and dim by means of stepping down the sublevels. When modifying this routine to incorporate more resolution or increased versatility, care must be taken to account for transfer of control instructions to and from the delay routines.

The following is a schematic diagram of the COPS interface to 120V_{AC} lamps. The program will intensify or de-intensify the lamps under program control.

3.1 TRIAC LIGHT INTENSIFY ROUTINE

This program intensifies a light source by varying the conduction angle applied to the load. The maximum level of intensity is stored in "LEVEL," and the time to get to that level is specified by "FINO." Both these parameters may be altered to suit specific applications. To cause the program to de-intensify the light source, the sublevels must be decremented rather than incremented.

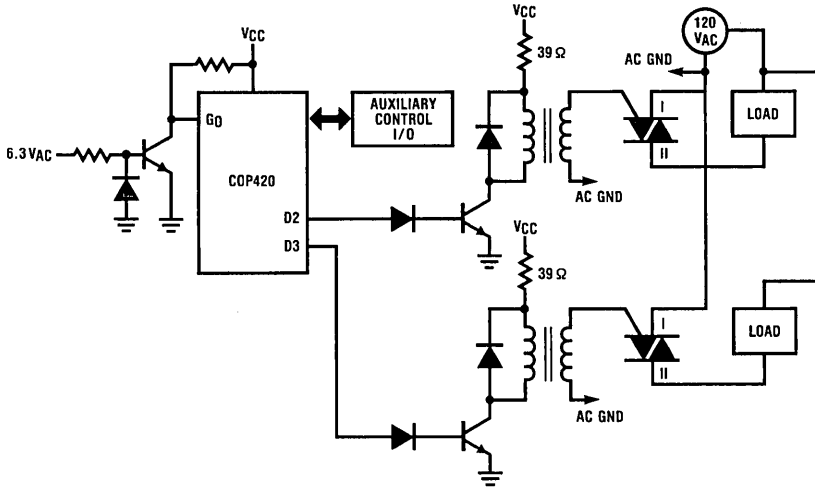


FIGURE 9. Triac Interface for COPS Program

TL/DD/6939-9

```

; TRIAC LIGHT INTENSIFY ROUTINE
;
;
; THIS PROGRAM INTENSIFIES A LIGHT SOURCE BY VARYING THE
; CONDUCTION ANGLE APPLIED TO THE LOAD. THE MAX LEVEL
; OF INTENSITY IS STORED IN 'LEVEL' AND THE TIME TO GET TO
; THAT LEVEL IS SPECIFIED BY 'FIND'. BOTH THESE PARAMETERS
; MAY BE ALTERED TO SUIT SPECIFIC APPLICATIONS. TO CAUSE
; THE PROGRAM TO DE-INTENSIFY THE LIGHT SOURCE, THE
; SUBLEVELS MUST BE DECREMENTED RATHER THAN
; INCREMENTED.
;
; HERE THE OPERATING PARAMETERS ARE DEFINED AND LEVEL
; INITIATION IS SPECIFIED
;
; THIS SECTION INITIATES CONTROL ON POWER UP OR RESET
; AND SYNCHRONIZES THE COPS DEVICE TO THE 60 HZ AC LINE
;
; OFFSET OF THE DETECTION CIRCUIT
;
; THIS SECTION PROVIDES THE DEBOUNCE FOR THE ZERO
; VOLTAGE DETECTION INPUT AND COMPENSATES FOR THE
;
; GETS HERE ON FIRST TRANSITION
; DID A LITTLE DELAY, IS IT STILL 0
; MUST HAVE HAD SOME NOISE GO BACK AND WAIT FOR TRUE ZC
;
; WELL, DO WE HAVE A CLEAN
; TRANSITION
; YES, GO TO MAIN ROUTINE
;
; FALSE ALARM, TRY AGAIN
; DO A DELAY TO COMPENSATE
; FOR NON SYMMETRIC ZC
; KEEP DELAY GOING
; GO TO MAIN ROUTINE
;
; THIS IS THE MAIN ROUTINE FOR THE INTENSIFY/DE-INTENSIFY
; OPERATIONS. TRANSFER OF CONTROL TO THIS SECTION
; OCCURS AFTER ZERO VOLTAGE CROSSING EACH HALF CYCLE.
; THIS MAKE USE OF TEMP REGISTERS THUS PARAMETERS
; NEED NOT BE REDEFINED FOR EACH OPERATION.
;
; DELAY INTO WAVEFORM
; USE TEMP REG
; DO DELAY
; POINT TO LEVEL TO INITIATE
; DELAY
; DELAY TO MAX LEVEL
; USE TEMP DIGIT TO DELAY
; ARE WE AT THE LEVEL ?
; MADE IT TO THE LEVEL
; NO
; DO SERIES OF .SMS TO GET
; THERE
; KEEP DOING IT
; AT MAX FIRE LEVEL
; INIT FOR SUBLEVEL DELAY
; AT SUB LEVEL ?
; NO DO DELAY
; YES
; VARIABLE DELAY
; DEC FIRE NUMBER
; TEST IF FIND AT 15
; NO KEEP FIRING AT THAT LEVEL
; YES INC SUBLEVEL
; IS MAX SUBLEV REACHED
; NO INC SUBLEV
; YES FIRE IT
; GO TO NEXT SUBLEVEL
; SET FIRE TIME
; GO FIRE
;
; FORM
; PAGE 1
;
; FORM
; PAGE 2

```

COP NOTE 6

```

; SUBROUTINE PAGE
ING:  CLRA
      AISC      1
      JP      ADEX      ; GO ADD ONE TO DIGIT
DEC:  CLRA      ; 0 TO A
      COMP     ; CREATE A 15
ADEX: ADD      ; ADD A TO RAM
      X        ; PUT BACK (D - 1 IN A NOW)
      RET
DE5:  LBI      0,10    ; DELAY ROUTINE
      CLRA    ; WILL BE REPLACED LATER
      AISC    3
      JP      , -1
      LD
      XIS
      JP      , -5
      RET     ; DONE DELAY
FIRE: LBI      0,15    ; PULSE D OUTPUT
      OBD
      NOP

NOP
NOP
LBI      0,0
OBD
SKBGZ    0      ; TEST WHICH DEBOUNCE IS
                ; NEEDED
                ; DEBOUNCE ONE TO ZERO
                ; DEBOUNCE ZERO TO ONE
                ; TEMP1 IS A TEMP REG
                ; VALUE IN TEMP1 DICTATES
                ; THE AMOUNT OF DELAY
JMP      HI
JMP      LO
LBI      TEMP1
PORT:   LD
        AISC    1
        JP      FOY
        OUT:   LBI      LEVEL      ; ALSO USED TO COPY LEVEL
        LD      1      ; RESTORE LEVEL
        X
        RET
        X
        FOY:   X
        JP      PORT
        .END

```

Testing of COP400 Family Devices

National Semiconductor
COP Note 7



Table of Contents

1.0 INTRODUCTION

2.0 PHILOSOPHY

3.0 BUILT-IN TEST FEATURES

- 3.1 Sync between DUT and Tester
- 3.2 Internal Logic Test
- 3.3 RAM Test
- 3.4 ROM Dump

This note will provide some insight into the test mode, the mechanics of testing, and the philosophy of how to implement a test of the COP-400 microcontrollers. Other than the obvious, (verifying that the part meets the specifications), the reason for the test must be considered. Somewhat different criteria may hold, depending on the objective. The manufacturer wafer sort or final test can differ from an incoming inspection at the user's plant, or a field reject test. The first two tests have limited interest as this is not a justification of the testing done on the part during manufacture. Rather, this is a guide for those doing user functional testing.

1.0 INTRODUCTION

Since the introduction of the very first semiconductor devices, testing has been a major problem and expense in their production and use. As the complexity has risen, testing has become a more significant factor. With today's single chip microcontrollers like the COP^{STM} devices this is particularly true as one has a complete computer system in a chip. In order to reduce the testing burden, the facilities to ease the testing have been built into the COPS devices. With the test ability built into the device for production test, the user need only follow set procedures to verify the chip at incoming inspection or field test.

2.0 PHILOSOPHY

The basic test philosophy requires that four major areas be exercised. These areas are:

- 1) Synchronize the device and tester.
- 2) Test the internal logic and I/O.
- 3) Test the RAM.
- 4) Verify the ROM program.

If the devices perform all of these four properly, the device is good. This is a reasonable assumption with a standard device that has a debugged test routine and is ROM programmed. A custom circuit just going into production might not have the accumulated test background. By attacking the problem on a "sum of the parts" approach, one need not do any exhaustive functional test on routine production parts. This will be a major gain where lengthy time consuming or time dependent routines are involved. If one attempts to do a functional test of the chip, a sequence that is unique to the application is needed. Thus, a test program must be written and debugged for each ROM pattern. Further, a test box/board must be designed, built, debugged, documented, and maintained for each one. If testing has been considered from the beginning, the chip will have built-in capabilities to exercise the various parts of it. The different functional parts and instructions are tested to verify proper operation at the voltage and frequency limits.

3.0 BUILT-IN TEST FEATURES

The first step in testing the COP400 devices is to understand the built-in test control features. This will involve the SI/O and the L lines. The SO pin has been designed to be the control node for testing. The pin will normally be in an active low state and when forced high externally, places the chip in the test mode. It should be noted that this output can sink considerable current and one should not force the pin to the V_{CC} rail. By limiting the voltage to the 2.0/3.0V range one can not damage the device where the application of a higher voltage could. When forced into the test mode the SI pin controls the sub mode of the chip. With SI high the data placed on the L port is used as an instruction. When SI is low (and the L output is enabled) the contents of the ROM will be dumped out through the L port. Certain other internal functions have been implemented to allow these modes but these are not part of the basic operation. Included in this category is the activation of the skip signal to prevent the program counter from jumping out of sequence by executing a program control instruction.

3.1 Sync Between Tester and DUT

In order to be able to test a COPS chip, the tester must be in sync with the device under test (DUT). By using an external oscillator the two may be run at the same frequency. This is true regardless of the option or type of oscillator chosen for the chip. Even the RC configuration may be overridden with an external signal that meets the level requirements. In addition to running at the same frequency, the chip and tester must be in sync on a bit basis. See *Figure 1*. The supportive features mentioned above include the condition of the SK signal being a bit (instruction) clock until stopped by software in the program. Hence, one can start the tests based on an edge change of SK. It is important that this be accurate because all data I/O changes will be relative to the SK timing (see the appropriate device data sheet).

It should also be noted that the oscillator frequency is programmed to a rate of 4-32 higher than SK. If one is building a test fixture for more than one device, some method must be available to enter this number. If one is testing a COP420 or COP421 near its upper limit it would be wise to do the SK sync operation at a lower rate and then increase the input frequency. This is desirable because the phase relationship is close to TTL propagation delays at the upper limit. Implementation of the area could be a preset counter that is gated on after a zero to one transition is seen on SK. Continual comparison could be made but once in sync, there should not be any need for the comparison as they should remain in sync.

The basic use of this "sync counter" is to derive the proper timing for loading data and instructions into the chip and verify the outputs. The COP402 data sheet should be used as a guide for these times, modified properly for the L and C parts. For those designing testers, it is suggested that one not attempt to test worse case timing changes as these could be very difficult to implement. Like other parametric tests these should in general be left to the professional test equipment.

3.2 Internal Logic Test

With the device and the tester in sync, actual testing may begin. See the sequence control circuit of *Figure 2*. To place the chip into the test mode the SO output is pulled to a one level (between 2.0 and 3.0 volts). It should be pulled with a circuit that will limit the upper voltage to 3V as this output can have a significant current sink capability. On power up (or after reset) the SO line is set to a zero by the internal logic. An internal sense line will detect the forced condition and provide test control. A delay of 10 ms should be taken after power-up to allow the power on reset circuit to time out before instructions can be executed. If the reset pin is activated in mid-program for some reason, several instructions cycle times should be ignored to insure complete operation.

The tester should at this point force instructions into the L port. These instructions will be executed as if they were from the ROM. The sequence of the instructions is not particularly critical. Table 1 gives an example sequence. The main steps are to be able to detect an output change (OGI) early to verify connection/operation. It is much better to find a problem before going through the steps of loading RAM and then finding that the chip doesn't work. All instructions should be exercised although certain ones should be postponed. Enabling the Q register to the L port is an example. This would interfere with the insertion of instructions on the L port. Another problem is the SO test which could be set up with an XAS and then released from the test mode to check proper data output.

Certain commands will require more effort than others. To check the program counter during JMP's and sub-routine operation will require that known info at the new address be available. One should execute a JSRP at some known address and release the test mode to see that the operation in the subroutine (e.g., SC) is done and that a return is made to $N + 1$. At this point test mode can be re-established to continue the test. The main point to remember is to provide a positive indication of the success of that specific test.

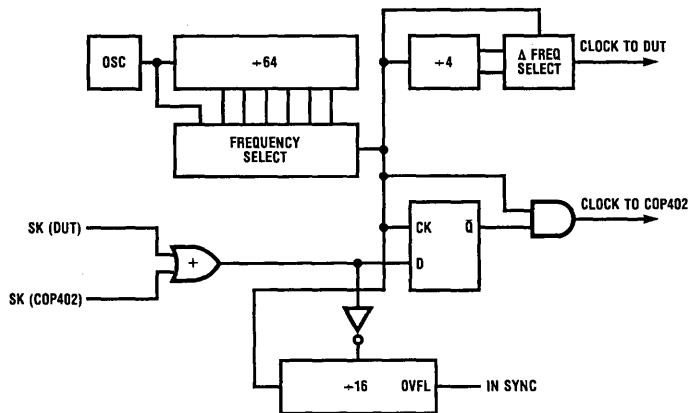


FIGURE 1. Tester Clock Generation and Synchronization Circuit

TL/DD/6940-1

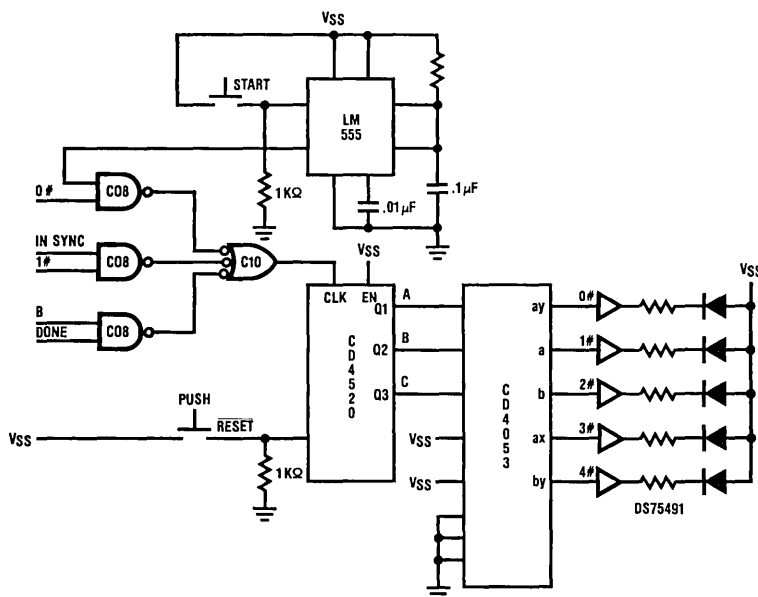


FIGURE 2. Tester Mode Sequencer

TL/DD/6940-2

3.3 RAM Test

The verification of RAM is a part of the internal logic test, but is treated separately here. One must check both the RAM and its address register to find all faults. An example of this testing would be to load RAM with a string of STIL commands. By then going back and reading this data to the outside (through an OMG instruction in a loop) the tester could verify both RAM and address were functional. One could then load RAM with all 6's and 9's (or 5's and 10's) sequentially to insure that all bits were functional and adjacent bits not shorted. Other similar tests could be run at the discretion of the user to do further testing. All of these tests would utilize the output of data via the G ports to validate the data. See the comparator circuit *Figure 3*.

3.4 ROM Dump

Successful operation of the internal logic tests and RAM will lead to the final test phase, ROM comparison. In order to

check the ROM contents, the ROM dump mode must be entered. One should force a JMP to an address near the end of the ROM space (3FF for a 420 chip, 1FF for a 410). A desirable point might be 3FA. The program counter will step ahead on each instruction cycle unless a program control is executed. The next step is to load the Q register with a non-conflicting value so that the enabling of the L outputs will not destroy the second byte of the LEI instruction as control is passed into the ROM dump mode. After going to this address, one should execute an enable of the L lines to the output port (LEI 4). Having done this the external buffers should be disabled and the SI pin taken low. This will allow data out and remove potential level conflicts. By letting the PC step ahead to address zero one can then begin the byte by byte comparison of data. In this mode the controller is not executing the code because the skip line is enabled throughout the sequence. By halting a counter on a failure, one could determine the questionable address.

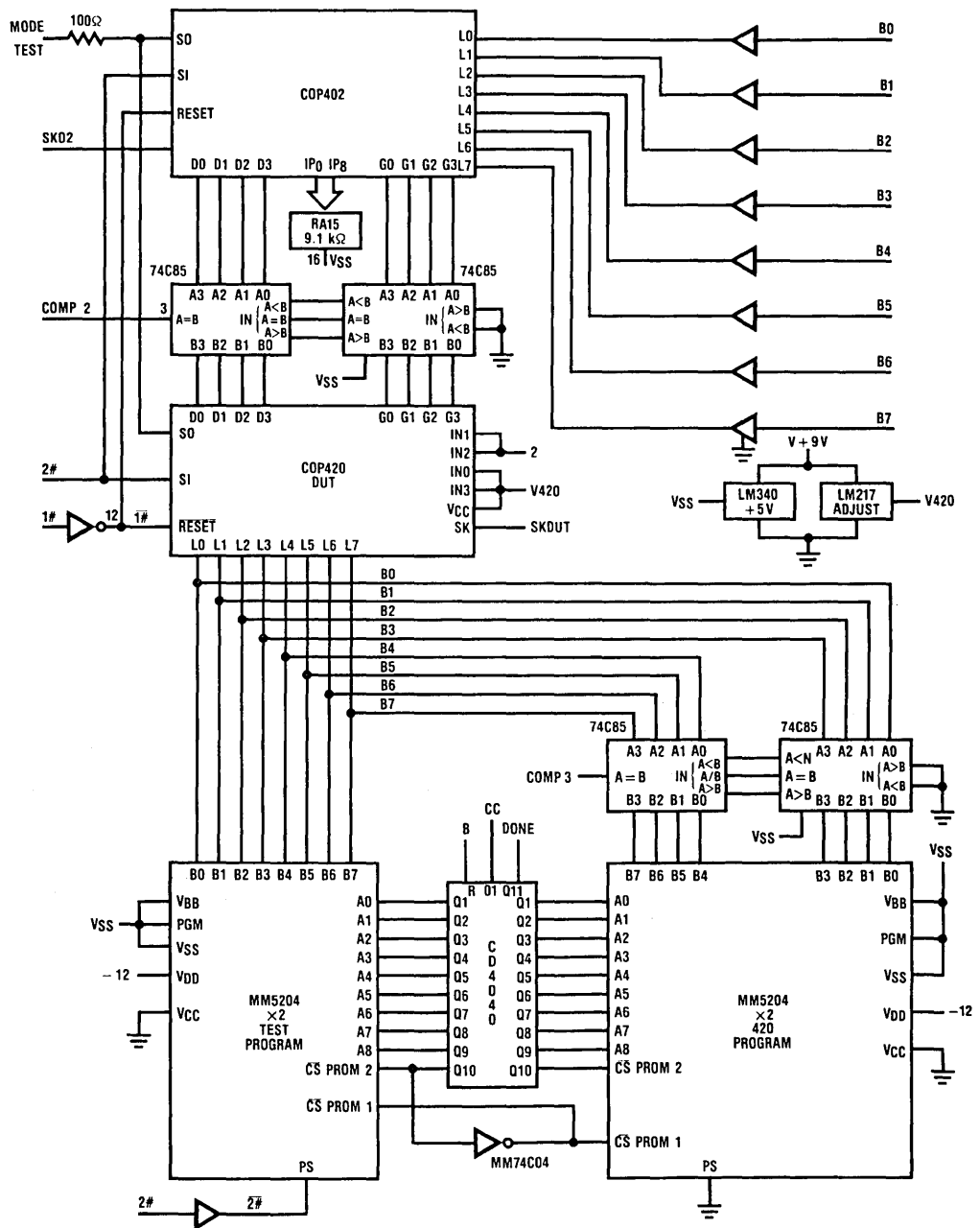


FIGURE 3. Functional Logic and RAM Comparison Circuit

TL/ DD/6940-3

TABLE I. Typical Test Sequence

INSTRUCTION	RESULT	COMMENTS	INSTRUCTION	RESULT	COMMENTS
NOP	NO CHANGE	CHECK NOP & ALLOW TRANSIENT CYCLE FOR MODE	CLRA		
OGI 9	G(0 > 9)	NOT ON 410L/411L	ASC		CHECK ADD WITH CARRY
OGI 6	G(9 > 6)	REVERSE ALL G STATES	SC		CHECK SET CARRY
STII 8		SET UP 0,0 FOR FUTURE	SKC		CHECK SKIP ON CARRY
LBI 3,13		B TO NEW POSITION (3, 13)	LDD 0,0		
OBd	D(0 > 13)	CHECK D	X		STORE A
CLRA		MAKE SURE A = 0	OMG	G = 9	NO CHANGE
XABR		3 > A; 0 > Br	CLRA		
CAB		MOVE 3 to Bd	ASC		
OBd	D(13 > 3)	CHECK XABR CAB & D CHANGE	X		
CLRA		!	OMG	G(9 > 10)	CARRY ADDS ONE TO MEMORY
AISC 2		IFORCE A > 2	CAMQ		STORE A & M IN Q; 10,9
CAB		2 > Bd	XDS		9 > 3,1; 10 > A; Bd > 3,0
OBd	D(3 > 2)	VERIFY 2 FROM A > Bd	X		STORE 9 IN 3,0
STII 7		7 > 0.2 & Bd > 3	OMG	G(10 > 9)	
OBd	D(2 > 3)	STII INCREMENTS Bd	LD 2		9 > A; Bd > 1,0
CAB		SEE THAT A STILL THE SAME			
OMG	G(6 > 7)	OMB & RAM CHECK			
CLRA			OMG	G(9 > 1)	
CAB		B(0,0)	LD 3		1 > A; Bd > 2,0
OMG	G(7 > 8)	TIE IN RAM, A & G OPERATION	OMG	G(1 > 2)	
SMB 0		SMB INST. CHECK	ADD		ADD WITHOUT CARRY
OMG	G(8 > 9)	:	X		STORE 3 IN 2,0
SMB 1		:	SC		
OMG	G(9 > 11)	:	LDD 0,0		7 > A
RMB 0		:	CASC		CHECK CASC
RMB 3		:	SKC		
X		:	X		STORE 12
CAB		:0 > 0,0; 2 > A	OMG	G(2 > 12)	
OMG	G(11 > 7)	A = 2 > B	CLRA		
LD 1		OUTPUT M(0,2)	AISC 3		
XAD 0,0		M(0,2) > A; B > 1,2	X		
AISC 15		A(7) < -> M(0,0) 2	SC		:CHECK
LDD 0,0		AISC CHECK; A = 1	SKC		:SKC/SC
X		CHECK SKIP OF 2 BYTE INST.	X		:
OMB	G(7 > 1)	STORE 1	OMG	G(12 > 3)	
LD 0		VERIFY	RC		:
ADT		COPY 1,2 BACK TO A	SKC		:CHECK
XDS		ADD TEN	X		:RC
XDS		LEAVE 11 IN 1,2; GO 1, 1 WITH 1	OMG	G(3 > 12)	
OBd	D(2 > 0)	LEAVE 1 IN 1,1; GO 1,0 W ?	LBI 0,0		:CHECK
STII 5		CHECK Bd MOVEMENT	LBI 1,15		:SEQUENTIAL LBI'S
CBA		5 > 1,0; Bd TO 1,1	LBI 2,7		ALSO SKIPPED (LBI 2,7 NOT IN 410)
AISC 3		CHECK B > A	OMG	G(2 > 7)	
		AISC CHECK 4 > A	CQMA		LOAD CONSTANTS FROM Q
			OMG	G(7 > 9)	CHECK
INSTRUCTION	RESULT	COMMENTS	X		:
XDS		1 > A; 4 > 1,1	OMG	G(9 > 10)	
OMG	G(1 > 5)	FROM 1,0	LEI 1		
XDS		5 > A; 1 > 1,0; Bd < 15 SKIP	XAS		STORE A -> S(9)
LDD 0,0		SKIPPED !	CLRA		
OBd	D(0 > 15)		AISC 7		:
AISC 4		9 > A	SKGBZ 0		:
X		9 > 15	X		:CHECK
OMG	G(5 > 9)		OMG		:
CLRA			SKGBZ 1		:G BIT
COMP		ONES TO A	X		:
XOR		FLIP MEMORY	OMG	G(10 > 7)	
XIS		6 > 1,15; 9 > A; Bd > 1,0	SKGBZ 2		:
LDD 0,0		SKIP	X		:
SKE			OMG	G(7 > 10)	:TESTS
LB 1,2		SKIP 2 WORD LBI (NOT IN 410)	SKGBZ 3		:
OBd	D(15 > 0)	VERIFY WORD	X		:
SKE		11 NOT = 9	OMG	G(10 > 7)	
LBI 1,0		BACK TO 1,0			
SMB 2		:	INSTRUCTION	RESULT	COMMENTS
SKE		:	SKGZ		
RMB 2		:CHECK BIT	X		:CHECK
SKE		:MANIPULATIONS	OMG	G(7 > 10)	
SMB 3		:	OGI 0	G(10 > 0)	:G TEST
SKE		:	SKGZ		:
LDD 0,0		Bd > 2,0	X		:
X 3		9 > 1,1; 4 > A	OMG	G(0 > 10)	
XAD 1,1		4 > 2,0; Bd > 3,1	SKMBZ 0		
XIS 1		INPUT G PORT	X		CHECK MEMORY BIT TESTS
ING		STORE	OMG		NO CHANGE
X			SKMBZ 1		

TABLE I. Typical Test Sequence (Continued)

INSTRUCTION	RESULT	COMMENTS	INSTRUCTION	RESULT	COMMENTS
X			STII 2		
OMG	G(10 > 7)	NO SKIP	STII 9		
SKMBZ 2			STII 0		
X		WON'T SKIP	LBI 3,0		
OMG	G(7 > 10)		STII 7		
INIL		SEE THAT L LATCHES RESET	STII 14		
ININ		ASSUME G - > I	STII 5		
SKE			STII 12		
X1		Br > 1	STII 3		
OMG		SHOULD BE EQUAL	STII 10		
INIL		:	STII 1		
X		:	STII 8		
SKMBZ 3		:	STII 15		
OBD	D(15 > 0)	:INIL TEST	STII 6		
OGI 1		:	STII 13		
LBI 3,11		:	STII 4		
OGI 0		:	STII 11		
INIL		:	STII 2		
X		:	STII 9		
SKMBZ 0		:	STII 0		
OBD	D(0 > 11)	:			
NOP		:			
XAS		:	INSTRUCTION	RESULT	COMMENTS
X		:XAS TEST	LBI 0,0		CHECK FOR RAM DATA
OMG	G(10 > 9)	:	OMG		OUTPUT DATA
			LD		:
INSTRUCTION	RESULT	COMMENTS	XIS		:MOVE TO NEXT DIGIT
LBI 0,0		LOAD RAM WITH	OMG		OUTPUT DATA
STII 7		CONSTANTS USING	LD		:
STII 14		STII	XIS		:MOVE TO NEXT DIGIT
STII 5			OMG		OUTPUT DATA
STII 12			LD		:
STII 3			XIS		:MOVE TO NEXT DIGIT
STII 10			OMG		OUTPUT DATA
STII 1			LD		:
STII 8			XIS		:MOVE TO NEXT DIGIT
STII 15			OMG		OUTPUT DATA
STII 6			LD		:
STII 13			XIS		:MOVE TO NEXT DIGIT
STII 4			OMG		OUTPUT DATA
STII 11			LD		:
STII 2			XIS		:MOVE TO NEXT DIGIT
STII 9			OMG		OUTPUT DATA
STII 0			LD		:
LBI 1,0			XIS		:MOVE TO NEXT DIGIT
STII 7			OMG		OUTPUT DATA
STII 14			LD		:
STII 5			XIS		:MOVE TO NEXT DIGIT
STII 12			OMG		OUTPUT DATA
STII 3			LD		:
STII 10			XIS		:MOVE TO NEXT DIGIT
STII 1			OMG		OUTPUT DATA
STII 8			LD		:
STII 15			XIS		:MOVE TO NEXT DIGIT
STII 6			OMG		OUTPUT DATA
STII 13			LD		:
STII 4			XIS		:MOVE TO NEXT DIGIT
STII 11			OMG		OUTPUT DATA
STII 2			LD		:
STII 9			XIS		:MOVE TO NEXT DIGIT
STII 0			OMG		OUTPUT DATA
LBI 2,0			LD		:
STII 7			XIS		:MOVE TO NEXT DIGIT
STII 14			OMG		OUTPUT DATA
STII 5			LD		:
STII 12			XIS		:MOVE TO NEXT DIGIT
STII 3			OMG		OUTPUT DATA
STII 10			LD		:
STII 1			XIS		:MOVE TO NEXT DIGIT
STII 8			OMG		OUTPUT DATA
STII 15			LD		:
STII 6			XIS		:MOVE TO NEXT DIGIT
STII 13					
			INSTRUCTION	RESULT	COMMENTS
INSTRUCTION	RESULT	COMMENTS	LBI 1,0		CHECK FOR RAM DATA
STII 4			OMG		OUTPUT DATA
STII 11			LD		:
			XIS		:MOVE TO NEXT DIGIT

TABLE I. Typical Test Sequence (Continued)

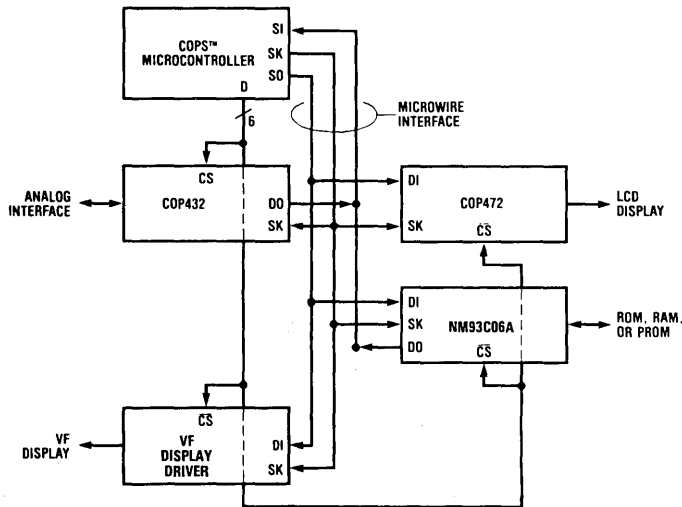
INSTRUCTION	RESULT	COMMENTS
SET TEST MODE		
JP X-2	:	
JSR Y	CHECK JP & JSR	
RELEASE TEST MODE	"Y" SHOULD CHANGE THE OUTPUT	
	CONDITIONS OF "X"	
	IF AT ALL POSSIBLE	
EXECUTE CODE (Y)		
SET TEST MODE		
RET		
RELEASE TEST MODE		
EXECUTE "X" AGAIN	VERIFIES RET	
SET TEST MODE		
JP X-2		
JSRP Z	CHECK JSRP & RETSK	
RELEASE TEST MODE		
EXECUTE CODE	"Z" SHOULD CHANGE "X"	
	OUTPUT CONDITIONS	
SET TEST MODE		
RETSK	DON'T CHANGE Z CONDITIONS —	
	RETSK	
RELEASE TEST MODE		
EXECUTE	" "	
SET TEST MODE		
LOAD A & M TO	FIND VALUE OF ADDRESS IN BLOCK	
	(4 PAGES)	
VALUE OF ADDRESS	AT OR JUST BEFORE AN OUTPUT	
TO GO TO	CHANGE SET A & M TO ADDRESS	
OUTPUT CHANGE	OF "VALUE"	
JID	CHECKS JID	
RELEASE TEST MODE		
EXECUTE OUTPUT		
SET TEST MODE	LOAD A & M WITH A UNIQUE ADDRESS	
LOAD A & M	SUCH THAT CONTENTS OF THAT	
	ADDRESS WILL BE SEEN ON G	
LQID		
X064	;OR USE THIS CAUSE THE DATA COMES	
	;FROM YOUR TESTER ANYWAY	
CQMA		
OMG	LIQUID & CQMA CHECKED	
X		
OMG	"	
INL	:	
OMG	G -> 2 INL TEST (COPY OF 2nd BYTE)	
X		
OMG	G -> E :	

This test sequence is not to be taken as a recommended test routine and is only shown as an example of what might be done to test various COPS parts. It is also advisable to approach measurements in the test mode with some caution. As stated earlier, one can force a large current into the SO node to place the chip in the test mode. Not only can this current do damage if unlimited, but it can also cause local current overloading such that some I/O conditions may be adversely affected. Obviously this will be more pronounced at higher V_{CC} voltages. A specific example is that the L output current sink test should only be tested at a V_{OUT} of 0.4V and 0.36 mA as the more stringent tests can exceed power limits when combined with the SO current.

MICROWIRE™

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and/or may permit the COPS controller to be packaged in a smaller (and even lower cost) package. (MICROWIRE peripherals may also be used with non-COPS controllers). For further applications information, refer to COPS Briefs 8 and 9. MICROWIRE makes sense.

The example below illustrates the power and versatility of MICROWIRE via an extreme example—using one of each type of peripheral with a single controller.



TL/DD/6940-4

COP431 SERIES, 8-BIT A/D CONVERTERS

The COP431 series is an 8-bit successive approximation A/D converter with a serial I/O and configurable input multiplexer with up to 8 channels. The serial I/O is configured to comply with the NSC MICROWIRE serial data exchange standard for easy interface to the COPS family of processors, and can interface with standard shift registers or other μ Ps.

The 2, 4 or 8 channel multiplexers are software configured for single-ended or differential inputs as well as channel assignment.

The differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

COP472-3 LIQUID CRYSTAL DISPLAY CONTROLLER

The COP472-3 Liquid Crystal Display (LCD) Controller drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472-3 contains an on-chip oscillator and generates all the multilevel waveforms for backplanes and segment outputs on a triplex display. One COP472-3 can drive 36 segments multiplexed as 3×12 ($4\frac{1}{2}$ digit display). Two COP472-3 devices can be used together to drive 72 segments (3×24) which could be an $8\frac{1}{2}$ digit display.

NM93C06A 256-BIT SERIAL ELECTRICALLY ERASABLE PROGRAMMABLE MEMORY

The NM93C06A is a 256-bit non-volatile memory. The device contains 256 bits of read/write memory divided into 16 registers of 16 bits each. Each register is serially read or written by the COP400 Family Controller. Written information is stored in a floating gate cell with at least 10 years of retention.

Current Consumption in NMOS COPSTM Microcontrollers

National Semiconductor
Application Brief 3
Len Distaso



Current consumption in the N-channel COPS microcontrollers is a function of manufacturing process variation and three operating condition parameters: temperature, voltage, and frequency. The aforementioned process variation swamps all other variations. Of the operating condition parameters, temperature is by far the most significant. This application brief is intended to provide the user with a guide to approximate the worst-case current consumption of the NMOS COPS microcontroller at a given set of operating conditions and to approximate the current variation with respect to temperature, voltage, and frequency.

Note that this is a guide only. Some approximations in the equations have been made. Only the current values found in the various device data sheets are guaranteed. Values derived by the techniques described here are neither guaranteed nor tested.

PROCESS VARIATION

If a user were to measure the current in two identical COPS microcontrollers under identical operating conditions (i.e., same temperature, voltage, and frequency), the results would probably be different. The reason for this difference is variation in the manufacturing process within its valid range. This variation can be quite substantial; a range of about 3 to 1 can be expected. This variation is essentially a device-to-device variation and basically not related to the operating conditions of the device. The three operating condition parameters (temperature, voltage, and frequency) affect current in the manner described below.

The values for current consumption in the various device data sheets are worst-case maximum values and assume that the processing parameters are at the end of the valid range which will produce maximum current consumption in the device.

THE EFFECT OF FREQUENCY

The frequency effect on current consumption is primarily a device design consideration. The higher the intended operating frequency, the higher the maximum current. However, once the device is designed in this process for a given maximum frequency, there is little variation with operating frequency. To be sure, there is some variation. As might be expected, current consumption is greater at higher frequencies. The variation is, however, slight—typically less than 5%.

THE EFFECT OF VOLTAGE

The operating voltage of the microcontroller has a slightly greater effect on current consumption than the operating current. Current consumption increases with increasing operating voltage. On examining the MOS device equations, one finds that the device current is proportional to the square of a voltage term:

$$I \propto (V_{GS} - V_T)^2$$

where:

I = device current

V_{GS} = device gate to source voltage

V_T = device threshold voltage.

In the N-channel COPS devices, current is consumed primarily by the load devices. Most of these devices, though not all, are depletion mode devices with the gate and source tied together. Thus, V_{GS} is 0. Therefore, the primary mechanism for current consumption as related to voltage is variation in V_T . The depletion mode load devices in the COPS NMOS microcontrollers have geometries (length is much greater than width) which tend to minimize variations in threshold voltage. There are additional second order effects related to operating voltage, such as effective channel lengths shortening due to increased voltage, which affect current consumption. These effects, however, do not have a major impact on current consumption. Note also that the threshold voltage is affected by process variation. This is one of the areas where the process variation contributes to the device-to-device variation in current consumption. The user can typically expect to see a 5% to 10% variation in current due to operating voltage with the maximum current consumption occurring at maximum operating voltage.

THE EFFECT OF TEMPERATURE

Of the three operating parameters affecting current consumption in the NMOS COPS microcontrollers, temperature has by far the greatest impact. The relationship is given by the following simplified, empirical equation:

$$I(T) = I_0(T/T_0)^{-3/2}$$

where:

T_0 = reference junction temperature in °K

T = device junction temperature in °K

I_0 = device current at temperature T_0

$I(T)$ = device current at temperature T .

Although this equation is for a single transistor, it can be applied to the entire microcontroller since all the devices are made with the same process and will exhibit the same

characteristics. It should also be noted that the temperatures involved are device junction temperatures. The junction temperature is essentially a function of two items:

$$T_j = F(T_A, \theta_{jA})$$

where:

T_j = junction temperature

T_A = ambient temperature

θ_{jA} = package thermal characteristic.

The preceding relationship indicates that the package for the device will affect current because the package affects junction temperature. This should not come as a surprise. One need only consider the differences between ceramic and plastic packages to find support for this claim.

For purposes of discussion, it will be assumed that junction temperature is given by the following:

$$T_j = T_A + 25^\circ\text{K}$$

where T_j and T_A are as defined previously. Note that this is an approximation. It is not necessarily true for all packages, or any package. The relationship between junction temperature and ambient temperature is also not necessarily linear. However, the approximation is reasonable and provides a workable framework.

Substituting the junction temperature relationship into the current equation, the following equation results:

$$I(T_A) \cong I_0 \left(\frac{T_A + 25}{T_{AO} + 25} \right)^{-3/2}$$

where:

T_{AO} = reference ambient temperature, °K

T_A = ambient temperature, °K

I_0 = current at ambient temperature T_{AO}

$I(T_A)$ = current at ambient temperature T_A .

AN EXAMPLE

The COP320L has a specified maximum current of 10 mA. In this process, maximum current occurs at minimum temperature, which is -40°C in this case. It is desired to find the maximum current at 25°C . Therefore,

$$T_{AO} = -40^\circ\text{C} = 233^\circ\text{K}$$

$$T_A = 25^\circ\text{C} = 298^\circ\text{K}$$

$$I_0 = 10 \text{ mA}$$

$I(T_A)$ to be determined

$$I(T_A) \cong I_0 \left(\frac{T_A + 25}{T_{AO} + 25} \right)^{-3/2}$$

$$\cong 10 \text{ mA} (323/258)$$

$$\cong 7.14 \text{ mA.}$$

Thus the maximum current for the COP320L at 25°C is approximately 7 mA.

CONCLUSION

A means is provided to the user to approximate the current variation of the NMOS COPS microcontroller over its valid operating range. A given device will consume its maximum current at maximum operating voltage, maximum operating frequency, and minimum operating ambient temperature. Conversely, minimum current will be consumed at minimum operating voltage, minimum operating frequency, and maximum operating ambient temperature.

The user should remember that this document is intended as a guide only. The values produced here are reasonable but they are approximations and are not guaranteed values. The user should also remember that the equations and methods discussed here do not involve process variation. The numbers calculated approximate the worst-case maximum current values at a given set of operating conditions. The user should be prepared to see a wide range of values over the course of volume production.

Further Information on Testing of COPS™ Microcontrollers

National Semiconductor
Application Brief 4
Len Distaso



COP Note 7 describes the basic approach and philosophy for testing COPS microcontrollers. This application brief is intended to complement and expand COP Note 7. It is assumed that the reader is familiar with and has access to COP Note 7.

TEST MODE

On COPS microcontrollers, test mode is entered by forcing the SO output to a logic "1" when it should otherwise be a logic "0". The easiest way to do this is to hold the COPS device in reset, hold the $\overline{\text{RESET}}$ pin low, and pull SO up to a logic "1" level. **WARNING: Do not force more than 3.0V on SO, as damage to the device may occur.** SO should be forced to approximately 2.5V to guarantee entry into test mode and to protect the device from damage.

Once the device is in test mode, the state of the SI input controls the type of test. SI at a logic "1" (high level) conditions the device to accept instructions from an external source via the L port. In test mode, when SI is high, the internal ROM is disabled. SI at a logic "0" (low level) forces the device to dump the internal ROM to the L port where the user can read and verify the ROM contents.

INSTRUCTION INPUT

With the device in test mode and SI at a logic "1", the microcontroller will read the data at the L port as instructions. The instructions must be presented at the beginning of each cycle time and must remain valid during the whole cycle time. The chip SK output is the instruction cycle clock in test mode and can be used as the timing reference. *Figure 1* indicates the timing for instruction input using the chip's SK output as the reference. A new instruction must be valid at the L inputs within approximately 200 ns of the rising edge of SK. The user should make every effort to make this time (t_2 in *Figure 1*) as short as possible.

It is possible to create an external SK signal which more closely duplicates the internal SK. This requires building a divider from CKI and synchronizing the resultant signal with the device under test. This is significant because it is the internal version of the SK signal which is the master timing signal for the microcontroller. The short time from the rising edge of the SK output to instruction valid is necessary because the actual objective is to provide new instructions at the rising edge, or close to it, of the internal timing signal. If the user creates the external timing signal, the 200 ns time is not applicable. A new instruction, or ROM word, would be presented at each rising edge of the external signal. A method for generating and using this external SK is described in COP Note 7.

ROM DUMP

With SI at logic "0" in test mode, the microcontroller will dump the ROM to the L port. ROM will be dumped sequentially, one word at a time, starting at whatever value the

program counter contains. A new ROM word appears at the L lines every falling edge of the chip SK signal. The output timing (t_1 in *Figure 1*) is the L output timing as found in the various device data sheets. The device will remain in ROM dump mode as long as SI is at logic "0" in test mode. The program counter will wrap around from the maximum address to 000 and ROM dump will continue.

To get a ROM dump, the user cannot simply enter test mode and force SI to logic "0". Some conditioning of the device is necessary. This requires that the user first go into instruction input mode and set up the device. The suggested sequence is as follows:

1. Enter test mode—pull $\overline{\text{RESET}}$ low, force SO to about 2.5V.
2. Force SI to logic "1" and force 0s on L lines—RESET still low.
3. Force RESET high and input the following sequence to the device:

```
CLRA
JMP 3FC (modify for ROM size)
LQID
O44H
LEI 4
NOP
```

4. During the NOP, change SI from high to low as shown in *Figure 2*. The ROM dump should start at address 000H at the time shown in *Figure 2*.

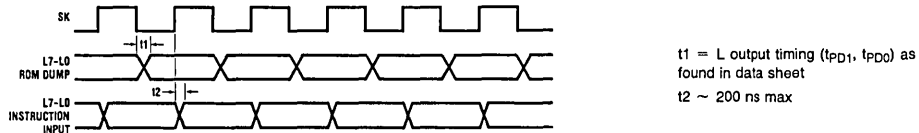
Figure 3 presents a general timing diagram for the entire sequence above. The jump instruction (JMP 3FC) in the sequence is used merely to position the program counter so that the ROM dump will begin at a specified location. That jump will be modified to reflect different ROM sizes or different desired starting locations for the ROM dump.

CHANGING BETWEEN INSTRUCTION INPUT AND ROM DUMP

The change from instruction input to ROM dump is accomplished according to the timing in *Figure 2*. It is necessary to do this to perform a valid ROM dump. However, it is not recommended to go the other direction, from ROM dump to instruction input, "on the fly". The instruction input mode should only be entered while the device is reset, RESET line low, to guarantee proper timing.

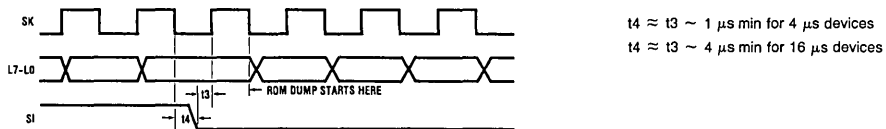
CONCLUSION

With COP Note 7 and this application brief, the user should be able to create a workable functional test for his COPS microcontroller. The relative timing is presented here and general techniques and sequences are provided in COP Note 7.



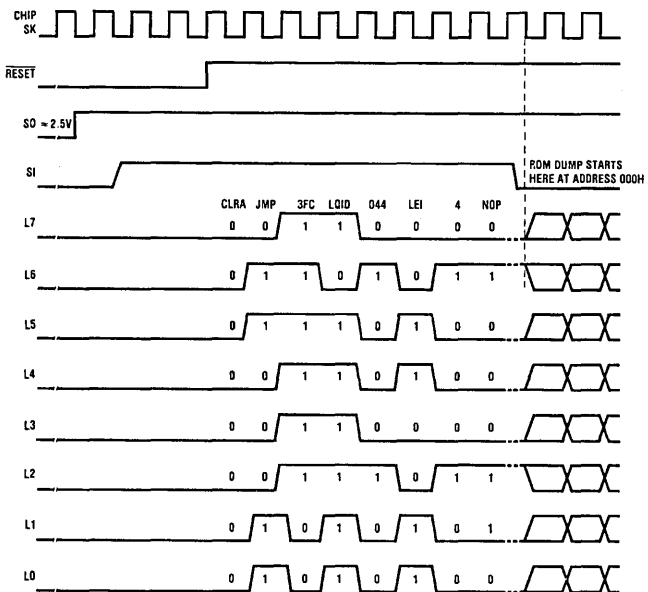
TL/DD/5146-1

FIGURE 1. Basic Test Mode Timing



TL/DD/5146-2

FIGURE 2. Timing for Changing from Instruction Input to ROM Dump—Test Mode



TL/DD/5146-3

FIGURE 3. Relative Timing for Suggested Sequence to Generate ROM Dump

COPS™ Interrupts

National Semiconductor
 Application Brief 6
 Jim Murashige



This brief describes in detail the timing requirements pertinent to COPS interrupts. *Figure 1* shows a typical enable-interrupt sequence in relation to the SK (Instruction Cycle) Clock. The SK clock is actually derived from the $\phi 1$ clock which is 180° out of phase with the $\phi 2$ clock. It is the $\phi 1$ and $\phi 2$ clocks to which all operation is referenced but for our purposes the SK will suffice. Program instructions are read on a rising $\phi 1$ edge and executed during the $\phi 1, \phi 2$ cycle time. Here we see the EN register interrupt enable bit EN2 being set with an LEI instruction. Interrupts are actually enabled on the $\phi 2$ leading edge of the second byte of the instruction point ②. Timing for an INTERRUPT DISABLE is essentially the same.

The interrupt line is sampled on the leading edge of $\phi 1$ as shown and interrupts are recognized if the minimum setup and hold times shown are satisfied. Note that the guaranteed times are longer than the typicals. The interrupt signal conditioning circuitry contains a falling edge detection circuit (a one shot) which requires that in addition to meeting the setup and hold times, the enable interrupt bit EN1 must have been turned on sometime before the end of the WINDOW OF OPPORTUNITY shown. If not, the interrupt will be missed and another high to low IN1 transition will be required. EN1 is automatically disabled upon interrupt recognition at point ⑤. Note that although the interrupt is recog-

nized at point ⑤ it will not be acted upon until all successive transfer of control instructions are executed as defined in the data sheets.

Because of gate delays it is doubtful that if an interrupt had been generated in time to meet the leading $\phi 1$ edge at point ② that the EN1 enable bit would have been on in time to meet the WINDOW of OPPORTUNITY.

By doing a worst case analysis one can see that in order to guarantee reception of an asynchronous interrupt IN1 must remain low for at least 2 instruction cycles. The analysis is as follows. Assuming that interrupts had been enabled prior to point ①, if the interrupt arrives a little after point ① it will not satisfy the minimum setup requirements bringing us up to a point ③ our total elapsed time becomes ③ - ① = 2 t_{CYC}.

In a dual COPS the interrupt sequence is the same except that now an instruction cycle time is made up of both a Processor X and a Processor Y instruction execution cycle. With one $\phi 1$ and $\phi 2$ clock per processor execution cycle the instruction cycle time is made up of 2 $\phi 1$'s and $\phi 2$'s. Therefore 1 instruction cycle time in a dual COPS is equivalent to 2 instruction cycle times in a single COPS as far as $\phi 1$'s, $\phi 2$'s and interrupts are concerned.

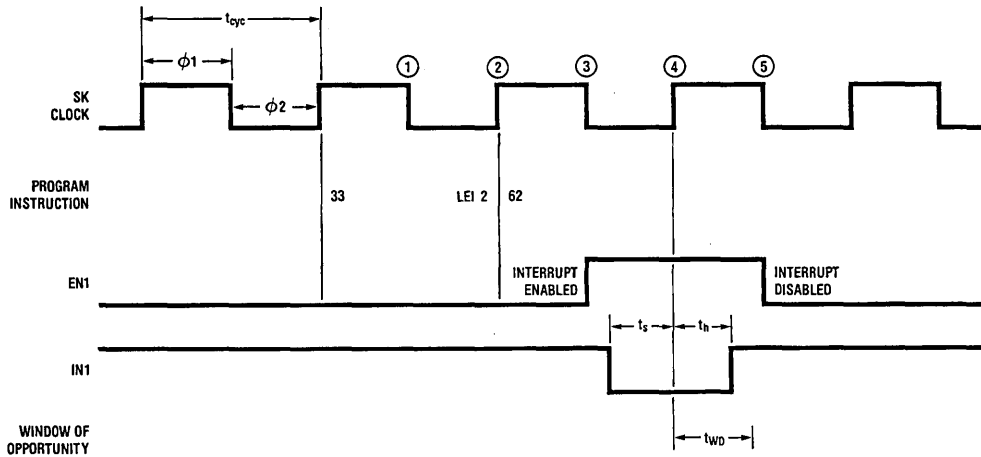


FIGURE 1. COP Interrupt Diagram

TL/DD/5180-1

Parameter	Min	Typ	Max
t_s	$\frac{1}{2} t_{cyc}$	200 ns	
t_h	$\frac{1}{2} t_{cyc}$	200 ns	
t_{wo}	$-\infty$	$\frac{1}{2} t_{cyc} - 600 \text{ ns}$	0

Protecting Data in Serial EEPROMs

National Semiconductor
Application Brief 15
Paul Lubeck



National offers a broad line of serial interface EEPROMs which share a common set of features:

- Low cost
- Single supply in all modes (+5V ± 10%)
- TTL compatible interface
- MICROWIRE™ compatible interface
- Read-Only mode or read-write mode

This Application Brief will address protecting data in any of National's Serial Interface EEPROMs by using read-only mode.

Whereas EEPROM is non-volatile and does not require V_{CC} to retain data, the problem exists that stored data can be destroyed during power transitions. This is due to either uncontrolled interface signals during power transitions or noise on the power supply lines. There are various hardware design considerations which can help eliminate the problem although the simplest most effective method may be the following programming method.

All National Serial EEPROMs, when initially powered up are in the Program Disable Mode*. In this mode, the EEPROM will abort any requested Erase or Write cycles. Prior to Eras-

ing or Writing it is necessary to place the device in the Program Enable Mode†. Following placing the device in the Program Enable Mode, Erase and Write will remain enabled until either executing the Disable instruction or removing V_{CC}. Having V_{CC} unexpectedly removed often results in uncontrolled interface signals which could result in the EEPROM interpreting a programming instruction causing data to be destroyed.

Upon power up the EEPROM will automatically enter the Program Disable Mode. Subsequently the design should incorporate the following to achieve protection of stored data.

- 1) The device powers up in the read-only mode. However, as a backup, the EWDS instruction should be executed as soon as possible after V_{CC} to the EEPROM is powered up to ensure that it is in the read-only mode.
- 2) Immediately preceding a programming instruction (ERASE, WRITE, ERAL or WRAL), the EWEN instruction should be executed to enable the device for programming; the EWDS instruction should be executed immediately following the programming instruction to return

*EWDS or WDS, depending on exact device.

†EWEN or WEN, depending on exact device.

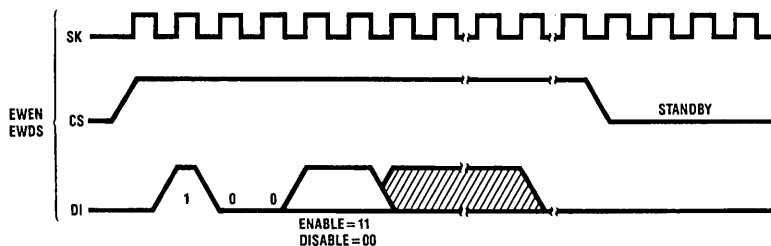
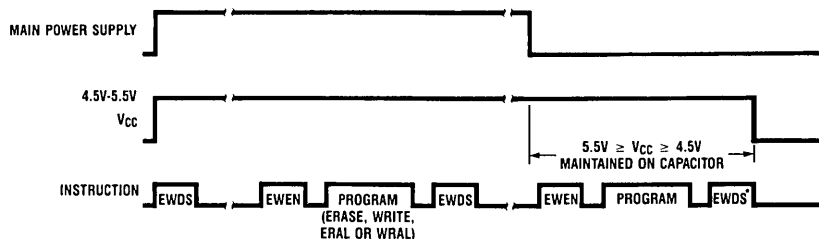


FIGURE 1. EWEN, EWDS Instruction Timing

TL/D/7085-1



*EWDS must be executed before V_{CC} drops below 4.5V to prevent accidental data loss during subsequent power down and/or power up transients.

FIGURE 2. Typical Instruction Flow for Maximum Data Protection

TL/D/7085-2

the device to the read-only mode and protect the stored data from accidental disturb during subsequent power transients or noise.

- 3) Special care must be taken in designs in which programming instructions are initiated to store data in the EEPROM after the main power supply has gone down. This is usually accomplished by maintaining V_{CC} for the EEPROM and its controller on a capacitor for a sufficient amount of time (approximately 50 ms, depending on the clock rate) to complete these operations. This capacitor

must be large enough to maintain V_{CC} between 4.5 and 5.5 volts for the total duration of the store operation, INCLUDING the execution of the EWDS instruction immediately following the last programming instruction. FAILURE TO EXECUTE THE LAST EWDS INSTRUCTION BEFORE V_{CC} DROPS BELOW 4.5 VOLTS MAY CAUSE INADVERTENT DATA DISTURB DURING SUBSEQUENT POWER DOWN AND/OR POWER UP TRANSIENTS.

A Users Guide to COPSTM Oscillator Operation

National Semiconductor
Application Note 326
Jim Murashige



AN-326

The following discussion is an overview of the COPS oscillator circuits meant to give the reader a working knowledge of the circuits. Although the descriptions are very general and light on detail; a background in complex frequency analysis is necessary. For additional information the references cited should be consulted as well as the many works on oscillator theory.

There are 2 basic circuits from which all of the COPS oscillator options are provided. (See option lists in individual data sheets.) The first and simplest in description is the astable one shot of *Figure 1* which gives us our RC oscillator option. A1 and A2 are inverters with A1 possessing a Schmitt trigger input. T1 is a large N channel enhancement MOS FET. Operation with the external R-C shown is as follows. Assuming C is initially discharged the CKI pin is low forcing T1 off. As C charges through R the trigger point of A1 is eventually reached at which time T1 is turned on discharging C and beginning a new cycle. Although almost any combination of R-C could be chosen, we would ideally like to have as short a discharge time as possible thereby eliminating the high variability in T1 drain current from device to device as a timing factor. For this reason R is chosen very large and C very small. This choice also leads to minimum R-C power dissipation. For the CKI Schmitt trigger clock input option the T1 MOS FET is merely mask disabled from the oscillator circuit.

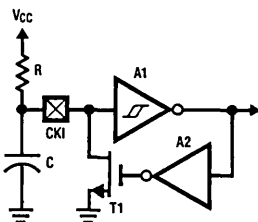


FIGURE 1. R-C Oscillator

TL/DD/5139-1

The second oscillator circuit is the classic phase shift oscillator depicted in *Figure 2*. Found not only on COPS but on most other microprocessor circuits it is the simplest oscillator in terms of component complexity but the most difficult to analyze.

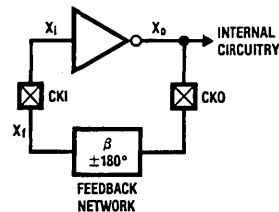


FIGURE 2. Phase Shift Oscillator

TL/DD/5139-2

The conditions under which the circuit will oscillate are described by the Barkhausen Criterion which states that oscillation will occur at the frequency for which the total loop phase shift from x_i to x_f is 0° or a multiple of 360° (i. e., x_f is identical to x_i). In addition the total loop gain must be > 1 to insure self propagation. The inverting amplifier shown between x_i and x_o provides 180° of phase shift thus leaving the feedback network to supply the other $\pm 180^\circ$. The feedback network can be comprised of active or passive components but highly effective oscillators are possible using only passive reactive components and the general configuration of *Figure 3*.

If you work out the feedback loop equations for *Figure 3* it can be shown that in order to achieve $\pm 180^\circ$ phase shift:

$$X_1 + X_2 + X_3 = 0 \quad (1)$$

X_1 and X_2 must both be inductors or capacitors (2)

therefore X_3 is inductive if X_1 is capacitive and vice versa if X_1 and X_2 are capacitors it is a Colpitts Oscillator

X_1 and X_2 are inductors it is a Hartley Oscillator

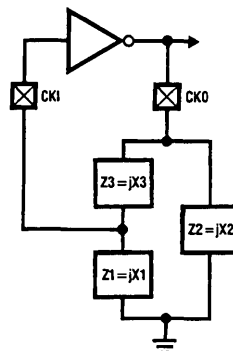


FIGURE 3. Typical Feedback Configuration

TL/DD/5139-3

3

The Colpitts configuration is commonly shown in microprocessor oscillator circuits (Figure 5) with the inductive X3 replaced by a crystal for reasons we shall soon see. The equivalent electrical model of a crystal is shown in Figure 4b and a plot of its Reactance versus Frequency shown in Figure 4c. R-L-C represent the electro-mechanical properties of the crystal and C₀ the electrode capacitance. There are 2 important points on the reactance curve labeled f_a and f_b.

$$\text{At } f_a = \frac{1}{2\pi} \sqrt{\frac{1}{LC}}$$

the crystal is at series resonance with L and C canceling each other out leaving only a nonreactive R for 0 phase shift. This mode of operation is important in oscillator circuits where a non-inverting amplifier is used and 0° phase shift must be preserved.

$$\text{At } f_b = \frac{1}{2\pi} \sqrt{\frac{1}{LC} + \frac{1}{LC_0}}$$

which is just a little higher than f_a the crystal is at parallel resonance and appears very inductive or capacitive. Note that the crystal will only appear inductive between f_a and f_b and that it becomes highly inductive very quickly. In addition f_b is only a fraction of a percent higher than f_a. Therefore the only time that the crystal will satisfy the X3 = -(X1 + X2) condition in the Colpitts configuration of Figure 5 is when the circuit is oscillating between f_a and f_b. The exact frequency will be the one which gives an inductive reactance large enough to cancel out:

$$X1 + X2 = \frac{1}{\omega C1} + \frac{1}{\omega C2} = \frac{1}{\omega} \left[\frac{1}{C1} + \frac{1}{C2} \right] = \frac{1}{2\pi f} \left[\frac{1}{C_L} \right]$$

Therefore by varying C1 or C2 we can trim slightly the oscillator frequency.

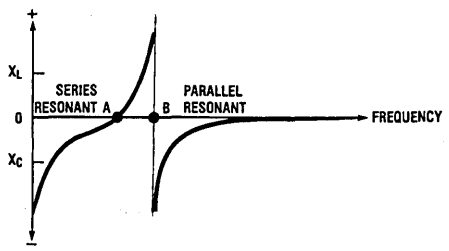
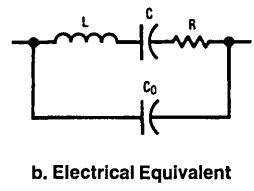
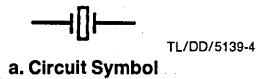


FIGURE 4. Quartz Crystal

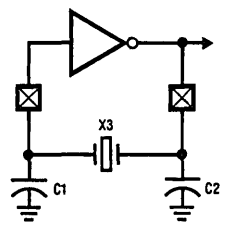


FIGURE 5. Colpitts Oscillator

The Q of a circuit is often bounced around in comparing different circuits and can be viewed graphically here as the slope of the reactance curve between f_a and f_b . Obviously the steeper the curve the smaller the variation in f necessary to restore the Barkhausen Phase Shift Criterion. In addition a lower Q (more R) means that the reactance curve won't peak as high at f_b , necessitating a smaller X1 + X2. When selecting crystals the user should be aware that the frequency stamped on the cans are for either parallel or series resonance, which, although very close, may matter significantly in the particular application.

An actual MOS circuit implementation of *Figure 5* is shown in *Figure 6*. It consists of a MOS inverter with depletion load and the crystal π network just presented. External to the COPS chips are the R_f and R_g resistors. R_f provides bias to the MOS inverter gate $V_g = V_0$. Since the gate draws no current R_f can be very large (M Ω) and should be, since we do not wish it to interact with the crystal network. R_g increases the output resistance of the inverter and keeps the crystal from being over driven.

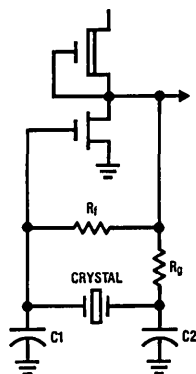


FIGURE 6. MOS Oscillator

TL/DD/5139-8

Of course the feedback network doesn't have to have the configuration of *Figure 3* and can be anything so long as the Barkhausen Phase Shift Criterion is satisfied. One popular configuration is shown in *Figure 7* where the phase shift will be 180°

$$\text{at } f = \frac{1}{(2\pi RC\sqrt{6})}$$

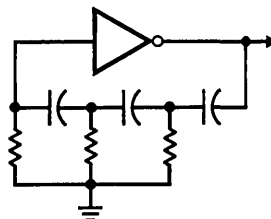
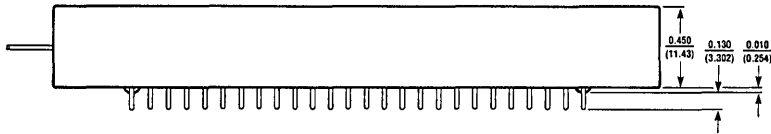
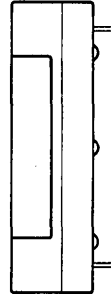
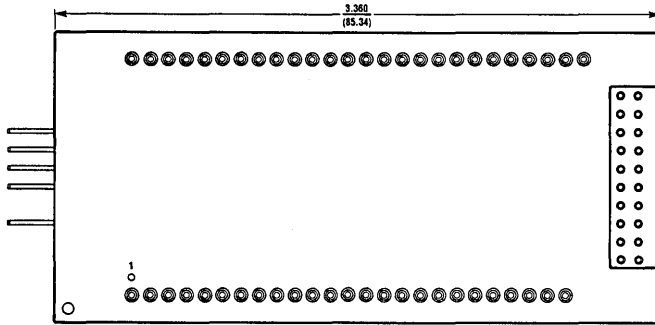


FIGURE 7. R-C Phase Shift Oscillator

TL/DD/5139-9

REFERENCES

1. Crystal/INS8048 Oscillator, AN-296, March 1982, National Semiconductor
2. Oscillator Characteristics of COPS Microcontrollers, CN-5, Feb. 1981, National Semiconductor
3. Integrated Electronics, Chapter 14, Millman and Halkias 1972
4. Handbook of Electronics Calculations, Chapter 9, Kaufman and Seidman 1979
5. 1982 COPS Microcontroller Databook, National Semiconductor



TL/DD/5139-10

Implementing an 8-Bit Buffer in COPS™

National Semiconductor
Application Note 329
David Pointer



AN-329

Sometimes a COP microcontroller must input and/or output 8-bit data; for instance, when handling ASCII data. In some applications, the processor must also provide temporary storage for 8-bit data before it is output. The COP instruction set and RAM structure lend themselves very nicely to providing a 32 digit, 8-bit buffer for a solution to these applications.

Such a large buffer is possible using a COP440 or a COP444L. The other members of the COP400 family with half as much RAM as these two would provide a 16 digit 8-bit buffer using the techniques described in this example.

Four adjacent RAM registers (16 digits each) are required. Referring to *Figure 1*, registers 4, 5, 6, and 7 are used for the buffer. Each RAM location contains 4 bits, so 2 locations will be used to store a byte of data. But these RAM locations are not adjacent to each other. You will note that the MSD of digit number 0A hex is in RAM location (4, A) while the LSD of the same digit is in RAM location (6, A).

The 2 RAM locations CHARM and CHARL are used for temporary storage of an 8-bit value.

In addition, 4 RAM locations are used for buffer pointers: those labelled IPM and IPL are the MSD and LSD of the

input pointer, and those labelled OPM and OPL are the MSD and LSD of the output pointer. Each pointer's function is to store an 8-bit counter whose value ranges from 00 hex thru 1F hex. The input pointer's value is used for storing the temporary storage buffer contents into the digit with the same number. For example, if the input pointer equals 14 hex, then the contents of CHARM would be stored in RAM location (5, 4) and the contents of CHARL would be stored in RAM location (7, 4). The output pointer's value is used for retrieving a digit from the buffer and putting it in CHARM and CHARL. For instance, if the output pointer equals 05 hex, then the contents of RAM location (4, 5) would be transferred to CHARM and the contents of RAM location (6, 5) would be transferred to CHARL.

A simple example of one possible application of the buffer is flowcharted in *Figure 2*. In this example, data is input to CHARM and CHARL, then stored in the buffer. An output device (a printer) is checked to see if it is ready to receive data. If it is, data is brought out of the buffer and put in CHARM and CHARL for output to the printer.

Pages 3 and 4 contain a listing of the subroutines needed to perform the data transfers in the 32-digit, 8-bit buffer.

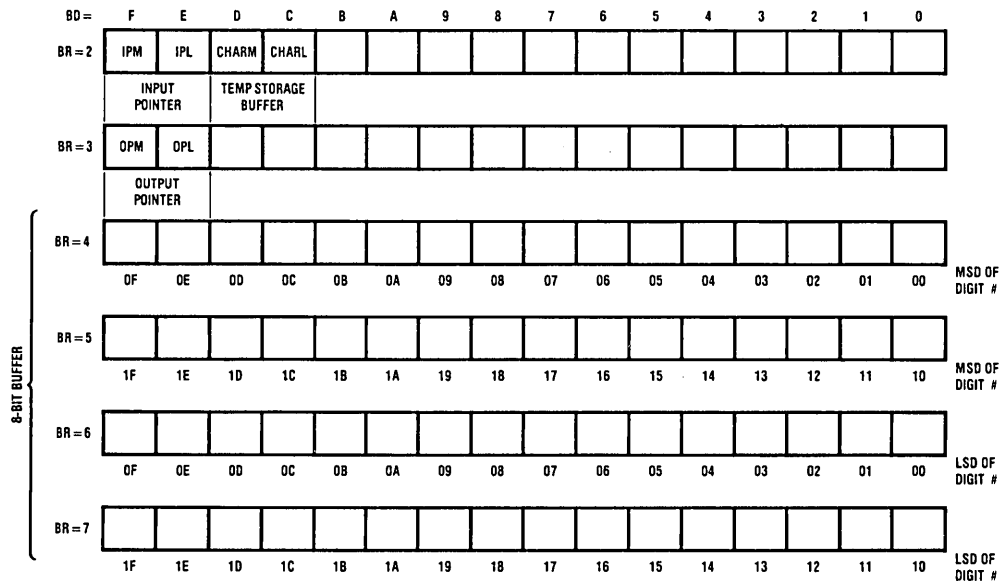


FIGURE 1. 8-Bit Buffer RAM Map

TL/DD/5181-1

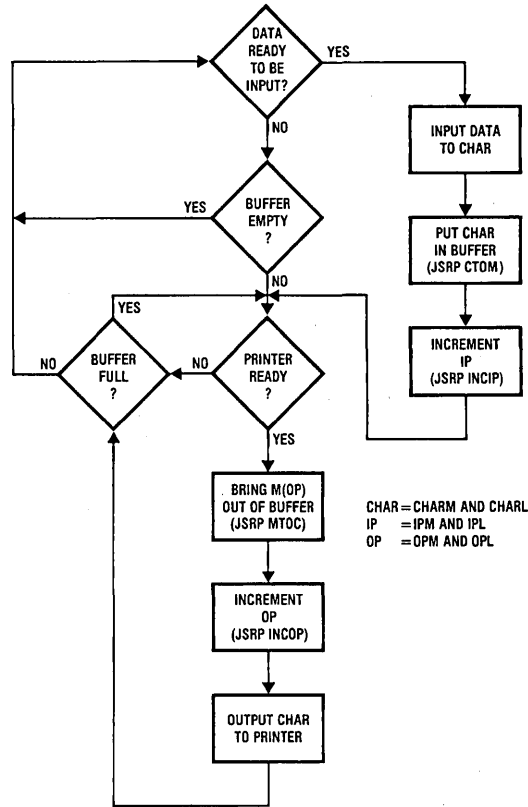


FIGURE 2. Buffer Example Flowchart

TL/DD/5181-2

COP CROSS ASSEMBLER PAGE: 1
BUFFER

```

1          ;*****
2          ;***                ***
3          ;*** 8-BIT RAM BUFFER SUBROUTINES ***
4          ;***                ***
5          ;*****
6          ;THESE ARE SUBROUTINES FOR IMPLEMENTING A 32 BYTE
7          ;BUFFER IN A COP440 OR COP444L RAM 9/3/82
8 01BC     .CHIP 444
9          .TITLE BUFFER
10 002D    CHARM = 2,13          ;TEMPORARY STORAGE BUFFER MSD
11 002C    CHARL = 2,12        ;TEMPORARY STORAGE BUFFER LSD
12 002F    IPM = 2,15          ;INPUT POINTER MSD
13 002E    IPL = 2,14         ;INPUT POINTER LSD
14 003F    OPM = 3,15         ;OUTPUT POINTER MSD
15 003E    OPL = 3,14         ;OUTPUT POINTER LSD
16 000 00    CLRA
17 0080     .PAGE 2
18          ;MTOC IS A SUBROUTINE THAT TRANSFERS M(OPM) AND M(OPL) TO
19          ;CHARM AND CHARL
20 080 233E  MTOC: LDD OPL          ;LOAD LSD OUTPUT POINTER
21 082 50    CAB                ;WHICH IS BD
22 083 233F  LDD OPM            ;LOAD MSB OUTPUT POINTER FOR B
23 085 54    AISC 4            ;MAKE BR EQUAL 4 OR 5
24 086 12    XABR
25 087 25    LD 2              ;LOAD M(OPM), MAKE BR = 6 OR 7
26 088 23AD  XAD CHARM         ;M(OPM) TO CHARM
27 08A 05    LD                ;LOAD M(OPL)
28 08B 23AC  XAD CHARL        ;M(OPL) TO CHARL
29 08D 48    RET
30          ;
31          ;
32          ;CTOM IS A SUBROUTINE THAT TRANSFERS CHARM AND CHARL TO
33          ;M(IPM) AND M(IPL)
34 08E 232E  CTOM: LDD IPL          ;LOAD LSD INPUT POINTER
35 090 50    CAB                ;WHICH IS BD
36 091 232F  LDD IPM           ;LOAD MSD INPUT POINTER FOR BR
37 093 54    AISC 4            ;MAKE BR = 4 OR 5
38 094 12    XABR
39 095 232D  LDD CHARM         ;LOAD MSD TEMP STORAGE
40 097 26    X 2              ;TO M(OPM), MAKE BR = 6 OR 7
41 098 232C  LDD CHARL        ;LOAD LSD TEMP STORAGE
42 09A 06    X                ;TO M(OPL)
43 09B 48    RET
44          ;
45          ;

```

COP CROSS ASSEMBLER PAGE: 2

BUFFER

```

46      .FORM
47      ;INCREMENTS INPUT POINT OR OUTPUT POINTER, ROLLS OVER
48      ;AT 1F HEX
49 09C 2D  INCIP: LBI   IPL           ;POINT TO LSD OF POINTER
50 09D 3D  INCOP: LBI   OPL           ;C=1 FOR INCREMENT
51 09E 22          SC           ;C=1 FOR INCREMENT
52 09F 00          CLRA
53 0A0 30          ASC           ;INCREMENT RAM VALUE
54 0A1 44          NOP           ;NEGATES SKIP CONDITION
55 0A2 04          XIS           ;STORE AND POINT TO (X,F)
56 0A3 00          CLRA
57 0A4 30          ASC           ;PROPAGATE CARRY, IF ANY, TO MS
58 0A5 44          NOP
59 0A6 06          X             ;STORE
60 0A7 45          RMB   1        ;ROLL OVER AT X'1F
61 0A8 48          RET
62      ;
63      ;
64      .END

```

COP CROSS ASSEMBLER PAGE: 3

BUFFER

```

CHARL 002C   CHARM 002D   CTOM  008E *   INCIP 009C *
INCOP 009D *  IPL   002E   IPM   002F   MTOC  0080 *
OPL   003E   OPM   003F

```

NO ERROR LINES

42 ROM WORDS USED

COP 444 ASSEMBLY

SOURCE CHECKSUM = C6A5

INPUT FILE 6:RBUFC. SRC VN: 5

Designing with the NM93C06 A Versatile Simple to Use E² PROM

National Semiconductor
Application Note 338
Masood Alavi



This application note outlines various methods of interfacing an NM93C06 with the COPST[™] family of microcontrollers and other microprocessors. Figures 1-6 show pin connections involved in such interfaces. Figure 7 shows how parallel data can be converted into a serial format to be inputted to the NM93C06; as well as how serial data outputted from an NM93C06 can be converted to a parallel-format.

The second part of the application note summarizes the key points covering the critical electrical specifications to be kept in mind when using the NM93C06.

The third part of the application note shows a list of various applications that can use a NM93C06.

GENERIC CONSIDERATIONS

A typical application should meet the following generic criteria:

1. Allow for no more than 10,000 E/W cycles for optimum and reliable performance.
2. Allow for any number of read cycles.
3. Allow for an erase or write cycle that operates in the 10-30 ms range, and not in the tens or hundreds of ns range as used in writing RAMs. (Read vs write speeds are distinctly different by orders of magnitude in E²PROM, not so in RAMs.)

4. No battery back-up required for data-retention, which is fully non-volatile for at least 10 years at room-ambient.

SYSTEM CONSIDERATIONS

When the control processor is turned on and off, power supply transitions between ground and operating voltage may cause undesired pulses to occur on data, address and control lines. By using WEEN and WEDS instructions in conjunction with a LO-HI transition on CS, accidental erasing or writing into the memory is prevented.

The duty cycle in conjunction with the maximum frequency translates into having a minimum HI-time on the SK clock. If the minimum SK clock high time is greater than 1 μ s, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max. On the low side no limit exists on the minimum frequency. This makes it superior to the COP499 CMOS-RAM. The rise and fall times on the SK clock can also be slow enough not to require termination up to reasonable cable-lengths.

Since the device operates off of a simple 5V supply, the signal levels on the inputs are non-critical and may be operated anywhere within the specified input range.

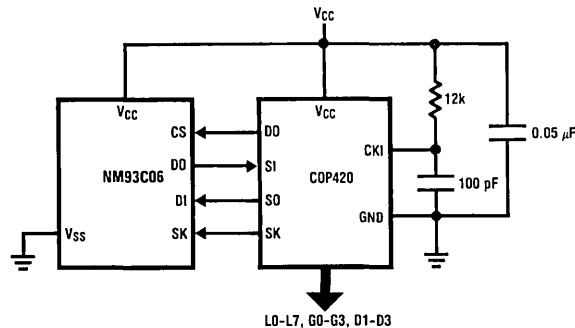


FIGURE 1. NM93C06—COP420 Interface

TL/D/5286-1

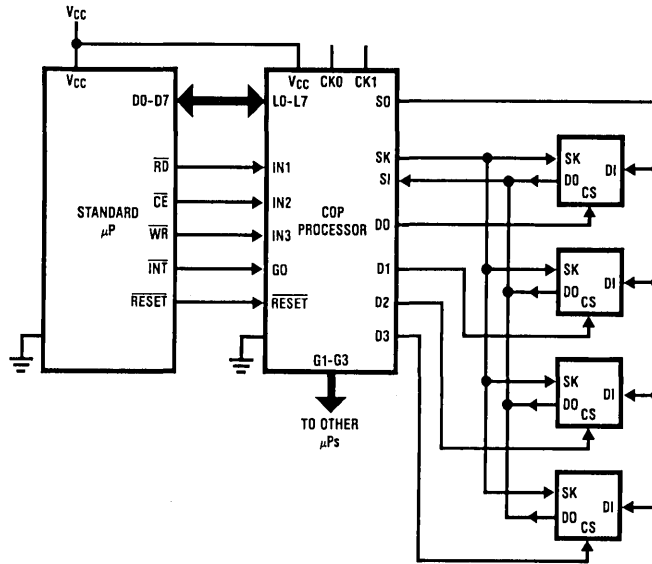
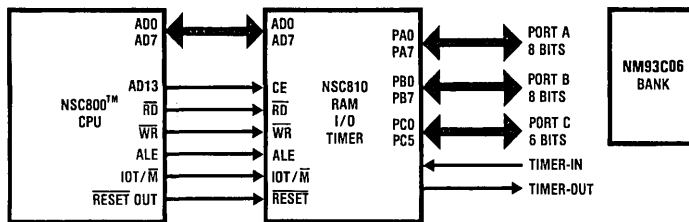


FIGURE 2. NM93C06—Standard μ P Interface Via COP Processor

TL/D/5286-2

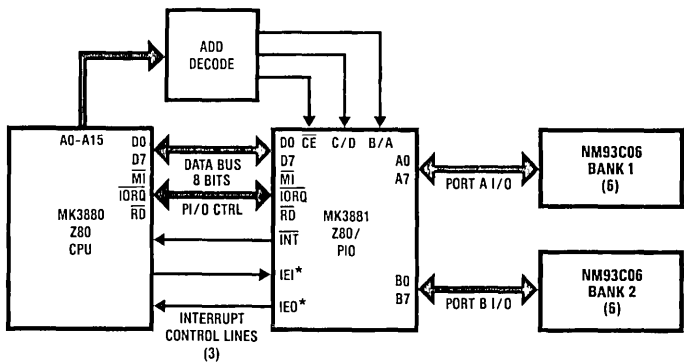


TL/D/5286-3

PA0 → SK
 PA1 → DI/DO } Common to all 9306's
 PA2-7 → 6CS for 6-9306's

- * SK is generated on port pins by bit-set and bit-clear operations in software. A symmetrical duty cycle is not critical.
- * CS is set in software. To generate 10-30 ms write/erase the timer/counter is used. During write/erase, SK may be turned off.

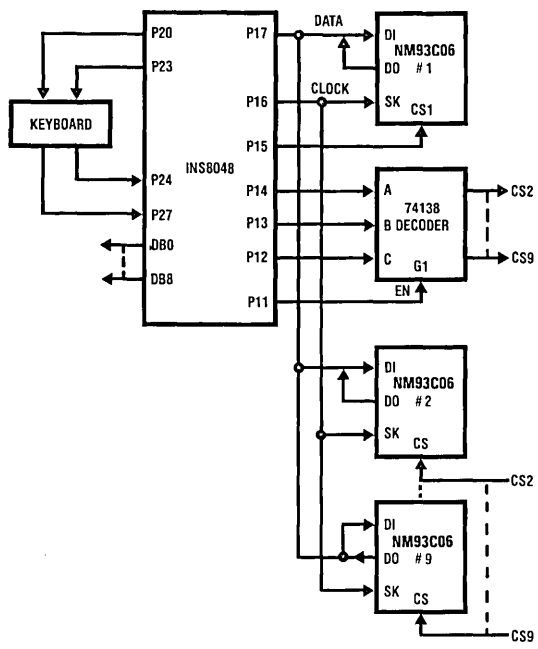
FIGURE 3. NSC800™ to NM93C06 Interface (also Valid for 8085/8085A and 8156)



TL/D/5286-4

Z80-PIO 9306
 A0 SK
 A1 DI/DO } Common to all 9306's (Bank 1)
 A2-A7 CS1-CS6
 * Only used if priority interrupt daisy chain is desired
 * Identical connection for Port B

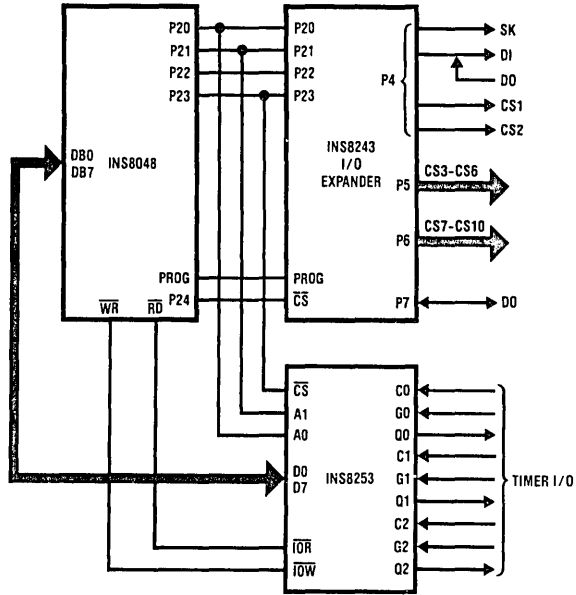
FIGURE 4. Z80—NM93C06 Interface Using Z80-PIO Chip



TL/D/5286-5

* SK and DI are generated by software. It should be noted that at 2.72 μ s/instruction. The minimum SK period achievable will be 10.88 μ s or 92 kHz, well within the NM93C06 frequency range.
 * DO may be brought out on a separate port pin if desired.

FIGURE 5. 48 Series μ P—NM93C06 Interface

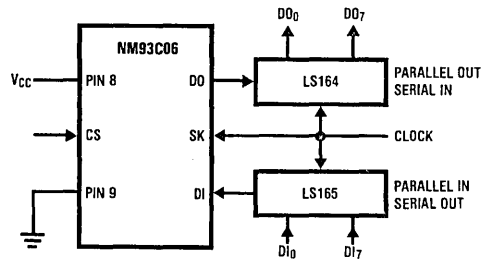


TL/D/5286-6

Expander outputs

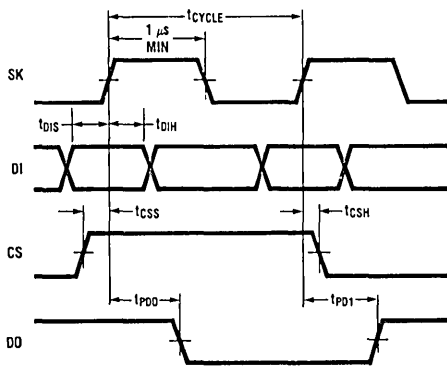
- DI } (COMMON)
- SK } (COMMON)
- Port 4 CS1
- CS2
- Port 5-6 CS3-CS10
- Port 7 DO (COMMON)

FIGURE 6. 8048 I/O Expansion



TL/D/5286-7

FIGURE 7. Converting Parallel Data into Serial Input for NM93C06



TL/D/5286-8

FIGURE 8. NM93C06 Timing

	Min	Max
t_{CYCLE} 0		250 kHz
t_{DIS} 400		ns
t_{DIH} 400		ns
t_{CSS} 200		ns
t_{CSH} 0		ns
t_{PD0}		2 μ s
t_{PD1}		2 μ s

THE NM93C06A

Extremely simple to interface with any μ P or hardware logic. The device has six pins for the following functions:

Pin 1	CS*	HI enabled
Pin 2	SK	Serial Clock input
Pin 3	DI	For instruction or data input
Pin 4	DO**	For data read, TRI-STATE® otherwise
Pin 5	GND	
Pin 8	V_{CC}	For 5V power
Pins 6-7	No Connect	No termination required

*Following an E/W instruction feed, CS is also toggled low for 10 ms (typical) for an E/W operation. This internally turns the VPP generator on (HI-LO on CS) and off (LO-HI on CS).

**DI and DO can be on a common line since DO is TRI-STATE[®] when unselected DO is only on in the read mode.

USING THE NM93C06

The following points are worth noting:

- SK clock frequency should be in the 0-250 kHz range. With most μ Ps this is easily achieved when implemented in software by bit-set and bit-clear instructions, which take 4 instructions to execute a clock or a frequency in the 100 kHz range for standard μ P speeds. Symmetrical duty cycle is irrelevant if SK HI time is $\geq 2 \mu$ s.
- CS low period following an E/W instruction must not exceed the 30 ms max. It should best be set at typical or minimum spec of 10 ms. This is easily done by timer or a software connect. The reason is that it minimizes the 'on time' for the high V_{PP} internal voltage, and so maximizes endurance. SK-clock during this period may be turned off if desired.
- All E/W instructions must be preceded by EWEN and should be followed by an EWDS. This is to secure the stored data and avoid inadvertent erase or write.
- A continuously 'on' SK clock does not hurt the stored data. Proper sequencing of instructions and data on DI is essential to proper operation.

- Stored data is fully non-volatile for a minimum of ten years independent of V_{CC} , which may be on or off. Read cycles have no adverse effects on data retention.
- Up to 10,000 E/W cycles/register are possible. Under typical conditions, this number may actually approach 1 million. For applications requiring a large number of cycles, redundant use of internal registers beyond 10,000 cycles is recommended.
- Data shows a fairly constant E/W Programming behavior over temperature. In this sense E²PROMs supersede EPROMs which are restricted to room temperature programming.
- As shown in the timing diagrams, the start bit on DI must be set by a ZERO - ONE transition following a CS enable (ZERO - ONE), when executing any instruction. ONE CS enable transition can only execute ONE instruction.
- In the read mode, following an instruction and data train, the DI can be a don't care, while the data is being outputted i.e., for next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
- The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI till another CS LO-HI transition starts a new instruction cycle.
- When a common line is used for DI and DO, a probable overlap occurs between the last bit on DI and start bit on DO.
- After a read cycle, the CS must be brought low for 1 SK clock cycle before another instruction cycle can start.

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ — After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE — Write command shifted in followed by data in (16 bits) then CS pulsed low for 10 ms minimum.

INSTRUCTION SET

Instruction	SB	Opcode	Address	Data	Comments
READ	01	10xx	A3A2A1A0		Read Register A3A2A1A0
WRITE	01	01xx	A3A2A1A0	D15-D0	Write Register A3A2A1A0
ERASE	01	11xx	A3A2A1A0		Erase Register A3A2A1A0
EWEN	01	0011	XXXX		Erase/Write Enable
EWDS	01	0000	XXXX		Erase/Write Disable
ERAL	01	0010	XXXX		Erase All Registers
WRAL	01	0001	XXXX	D15-D0	Write All Registers

NM93C06 has 7 instructions as shown. Note that MSB of any given instruction is a "1" and is viewed as a start bit in the interface sequence. The next 8 bits carry the op code and the 4-bit address for 1 of 16, 16-bit registers.

X is a don't care state.

The following is a list of various systems that could use a NM93C06

- | | |
|--|---|
| <p>A. Airline terminal
Alarm system
Analog switch network
Auto calibration system
Automobile odometer
Auto engine control
Avionics fire control</p> <p>B. Bathroom scale
Blood analyzer
Bus interface</p> <p>C. Cable T.V. tuner
CAD graphics
Calibration device
Calculator—user programmable
Camera system
Code identifier
Communications controller
Computer terminal
Control panel
Crystal oscillator</p> <p>D. Data acquisition system
Data terminal</p> <p>E. Electronic circuit breaker
Electronic DIP switch
Electronic potentiometer
Emissions analyzer
Encryption system
Energy management system</p> <p>F. Flow computer
Frequency synthesizer
Fuel computer</p> <p>G. Gas analyzer
Gasoline pump</p> <p>H. Home energy management
Hotel lock</p> <p>I. Industrial control
Instrumentation</p> <p>J. Joulemeter</p> <p>K. Keyboard -softkey</p> <p>L. Laser machine tool</p> <p>M. Machine control
Machine process control
Medical imaging
Memory bank selection
Message center control
Mobile telephone</p> | <p>Modem
Motion picture projector</p> <p>N. Navigation receiver
Network system
Number comparison</p> <p>O. Oilfield equipment</p> <p>P. PABX
Patient monitoring
Plasma display driver
Postal scale
Process control
Programmable communications
Protocol converter</p> <p>Q. Quiescent current meter</p> <p>R. Radio tuner
Radar detector
Refinery controller
Repeater
Repertory dialer</p> <p>S. Secure communications system
Self diagnostic test equipment
Sona-Bouy
Spectral scanner
Spectrum analyzer</p> <p>T. Telecommunications switching system
Teleconferencing system
Telephone dialing system
T.V. tuner
Terminal
Test equipment
Test system
TouchTone dialers
Traffic signal controller</p> <p>U. Ultrasound diagnostics
Utility telemetering</p> <p>V. Video games
Video tape system
Voice/data phone switch</p> <p>W. Winchester disk controller</p> <p>X. X-ray machine
Xenon lamp system</p> <p>Y. YAG—laser controller</p> <p>Z. Zone/perimeter alarm system</p> |
|--|---|

A Study of the Crystal Oscillator for CMOS-COPS™

National Semiconductor
Application Note 400
Abdul Aleaf



INTRODUCTION

The most important characteristic of CMOS-COPS is its low power consumption. This low power feature does not exist in TTL and NMOS systems which require the selection of low power IC's and external components to reduce power consumption.

The optimization of external components helps decrease the power consumption of CMOS-COPS based systems even more.

A major contributor to power consumption is the crystal oscillator circuitry.

Table I presents experimentally observed data which compares the current drain of a crystal oscillator vs. an external squarewave clock source.

The main purpose of this application note is to provide experimentally observed phenomena and discuss the selection of suitable oscillator circuits that cover the frequency range of the CMOS-COPS.

Table I clearly shows that an unoptimized crystal oscillator draws more current than an external squarewave clock. An RC oscillator draws even more current because of the slow rising signal at the CKI input.

Although there are few components involved in the design of the oscillator, several effects must be considered. If the requirement is only for a circuit at a standard frequency which starts up reliably regardless of precise frequency stability, power dissipation and etc., then the user could directly consult the data book and select a suitable circuit with proper components. If power consumption is a major requirement, then reading this application note might be helpful.

WHICH IS THE BEST OSCILLATOR CIRCUIT?

The Pierce Oscillator has many desirable characteristics. It provides a large output signal and drives the crystal at a low power level. The low power level leads to low power dissipation, especially at higher frequencies. The circuit has good short-term stability, good waveforms at the crystal, a frequency which is independent of power supply and temperature changes, low cost and usable at any frequency. As compared with other oscillator circuits, this circuit is not disturbed very much by connecting a scope probe at any point in the circuit, because it is a stable circuit and has low impedance. This makes it easier to monitor the circuit without any major disturbance. The Pierce oscillator has one disadvantage. The amplifier used in the circuit must have high gain to compensate for high gain losses in the circuitry surrounding the crystal.

TABLE I

A. Crystal oscillator vs. external squarewave COP410C change in current consumption as a function of frequency and voltage, chip held in reset, CKI is ÷ 4.

I = total power supply current drain (at V_{CC}).

Crystal

V_{CC}	f_{ckl}	Inst. cyc. time	$I_{\mu A}$
2.4V	32 kHz	125 μs	8.5
5.0V	32 kHz	125 μs	83
2.4V	1 MHz	4 μs	199
5.0V	1 MHz	4 μs	360

External Squarewave

V_{CC}	f_{ckl}	Inst. cyc. time	I
2.4V	32 kHz	125 μs	4.4 μA
5.0V	32 kHz	125 μs	10 μA
2.4V	1 MHz	4 μs	127 μA
5.0V	1 MHz	4 μs	283 μA

WHAT IS A PIERCE OSCILLATOR?

The Pierce is a series resonant circuit, and its basic configuration is shown below.

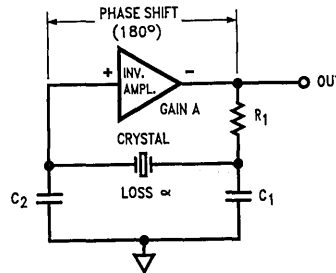


FIGURE 1

TL/DD/8439-1

For oscillation to occur, the Barkhausen criteria must be met: (1) The loop gain must be greater than one. (2) The phase shift around the loop must be 360° .

Ideally, the inverting amplifier provides 180° , the R_1C_1 integration network provides a 90° phase lag, and the crystal's impedance which is a pure resistance at series resonance together with C_2 acts as a second integration network which provides another 90° phase lag. The time constants of the two RC phase shifting networks should be made as big as possible. This makes their phase shifts independent of any changes in resistance or capacitance values. However, big RC values introduce large gain losses and the selected amplifier should provide sufficient gain to satisfy gain requirement. CMOS inverters or discrete transistors can be used as amplifiers. An experimental evaluation of crystal oscillators using either type of amplifier is given within this report.

CRYSTAL OSCILLATORS USING CMOS-IC

The use of CMOS-IC's in crystal oscillators is quite popular. However, they are not perfect and could cause problems. The input characteristics of such IC's are good, but they are limited in their output drive capability.

The other disadvantage is the longer time delay in a CMOS-inverter as compared to a discrete transistor. The longer this time delay the more power will be dissipated. This time delay is also different among different manufacturers.

As a characteristic of most CMOS-IC's the frequency sensitivity to power supply voltage changes is high. As a group, IC's do not perform very well when compared with discrete transistor circuits.

But let us not be discouraged. Low component count which leads to low cost is one good feature of IC oscillators.

As a rule, IC's work best at the low end of their frequency range and poorest at the high end.

Several types of crystal oscillators using CMOS-IC's have been found to work satisfactorily in some applications.

CMOS—TWO INVERTER OSCILLATOR

The two inverter circuit shown in *Figure 2* is a popular one. The circuit is series resonant and uses two cascaded inverters for an amplifier.

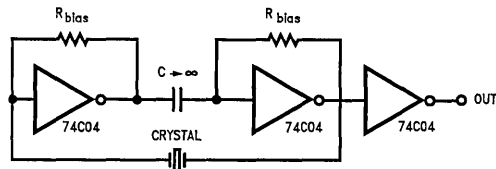


FIGURE 2

TL/DD/8439-2

Each inverter has a DC biasing resistor which biases the inverter halfway between the logic "1" and "0" states. This will help the inverters to amplify when the power is applied and the crystal will start oscillation.

The 74C family works better as compared with other CMOS-IC's. Will oscillate at a higher frequency and is less sensitive to temperature changes. The CMOS-COPS data sheet states that a crystal oscillator will typically draw $100 \mu\text{A}$ more than an external clock source. However, the crystal oscillator described above will draw approximately as much

current as an external squarewave clock. The experimental data presented below shows the comparison:

Chip held in Reset, $V_{CC} = +5.0\text{V}$

$f = 455 \text{ kHz}$, COP444C, CKI is $\div 8$

Instruction cycle time = $17.5 \mu\text{s}$

I = total power supply (V_{CC}) current drain

Oscillator Type	I (current drain)
Crystal Osc. (data sheet)	$950 \mu\text{A}$
Crystal Osc. (two inverter)	$810 \mu\text{A}$
Ext. Clock	$790 \mu\text{A}$

PIERCE IC OSCILLATOR

Figure 3 shows a Pierce oscillator using CMOS inverter as an amplifier.

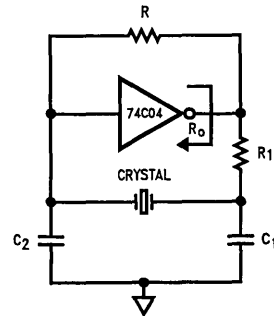


FIGURE 3

TL/DD/8439-3

The gain of CMOS inverter is low, so the resistor R_1 should be made small. This reduces gain losses. The output resistance of the inverter (R_o) can be the integrating resistor for the R_oC_1 phase lag network.

Omitting R_1 or with a small value of R_1 , the crystal will be driven at a much higher voltage level. This will increase power dissipation.

For lower frequencies (i.e., 32 kHz), R_1 must be large enough so that the inverter won't overdrive the crystal. Also, if R_1 is too large we won't get an adequate signal back at the inverter's input to maintain oscillation. With large values of R_1 the inverter will remain in its linear region longer and will cause more power dissipation. Typically for 32 kHz, R_1 should be constrained by the relation.

$$\frac{1}{2\pi R_1 C_1} \ll 32 \text{ kHz}$$

At higher frequencies, selection of R_1 is again critical. In order to drive a heavy load at high frequency, the amplifier output impedance must be low. In order to isolate the oscillator output from C_1 so it can drive the following logic stages, then R_1 should be large. But again, R_1 must not be too large, otherwise it will reduce the loop gain.

The value of R_1 is chosen to be roughly equal to the capacitive reactance of C_1 at the frequency of operation, or the value of load impedance Z_L .

$$\text{Where } Z_L = \frac{X_{C1}}{R_L}$$

$R_L = R_S =$ series resistance of crystal

The small values of C_1 and C_2 will help minimize the gain reduction they introduce.

typically: $C_1 = C_2 = 220$ pF at 1 MHz
 $C_1 = C_2 = 330$ pF at 2 MHz

DISCRETE TRANSISTOR OSCILLATOR

As mentioned earlier, a discrete transistor circuit performs better than an IC circuit. The reason for this is that in a discrete transistor circuit it is easier to control the crystal's source and load resistances, the gain and signal amplitude. A discrete transistor circuit has shorter time delay, because it uses one or two transistors. This time delay should always be minimized, since it causes more power dissipation and shifts frequency with temperature changes. *Figure 4* shows a basic Pierce oscillator using a transistor as an amplifier.

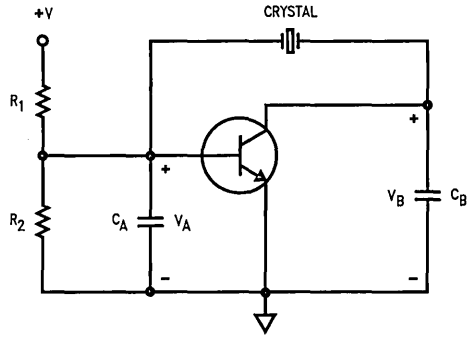


FIGURE 4

TL/DD/8439-4

The basic phase shift network consists of C_{A1} , C_{B2} and the crystal which looks inductive and is series resonant with C_{A1} and C_{B1} . The phase shift through the transistor is 180° and the total phase shift around the loop is 360° . The condition of a unity loop gain must also be satisfied.

$$\frac{V_A}{V_B} = - \left(\frac{C_B}{C_A} \right)$$

$$\frac{V_A}{V_B} = - \left(\frac{X_{CA}}{X_{CB}} \right)$$

For oscillation to occur, the transistor gain must satisfy the relation

$$G \left(\frac{V_A}{V_B} \right) \geq 1$$

where $G = -g_{fe}Z_L$

g_{fe} is the transconductance of the transistor

Z_L is the load seen by the collector

$$Z_L = \frac{X_B^2}{R_e}, \quad X_B = - \frac{1}{\omega C_B}$$

R_e is the crystal's effective series resistance.

The crystal's drive level

$$P_d = \frac{V_B^2 R_e}{X_B^2}$$

This drive level should not exceed the manufacturer's spec. Certain biasing conditions might cause collector saturation. Collector saturation increases oscillator's dependence on the supply voltage and should be avoided.

The circuit of *Figure 5* has been tested and has a very good performance.

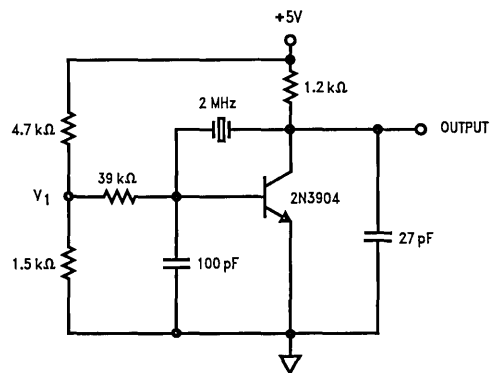


FIGURE 5

TL/DD/8439-5

This circuit will oscillate over a wide range of frequencies 2–20 MHz.

$$\text{Voltage } (V_1) = \frac{(5)(1.5)}{1.5 + 4.7} = 1.21V$$

$$\text{Base Current} = \frac{1.21 - V_{BE}}{39k} = 15.6 \mu A$$

At Saturation ($V_{CE} = 0$)

$$I_C (\text{SAT}) = \frac{5}{1.2} = 4.2 \text{ mA}$$

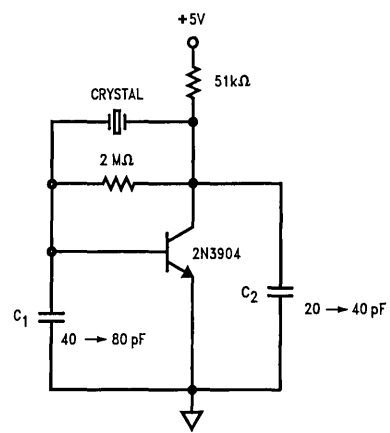


FIGURE 6

TL/DD/8439-6

Having $15.6 \mu\text{A}$ of base current, for saturation to occur

$$h_{FE} = \frac{4.2 \text{ mA}}{15.6 \mu\text{A}} = 269$$

The DC beta for 3904 at 1 mA is 70 to 210, so no problem with saturation, even at lower supply voltages.

The current consumption (power supply V_{CC} current drain) of COP444C using the above oscillation circuit is around $267 \mu\text{A}$.

The circuit of *Figure 6* is another configuration of discrete transistor oscillator.

The performance of above circuit is also good. The only drawback is that it does not provide larger output signal.

CONCLUSION

As discussed within this report, a discrete transistor circuit gives better performance than an IC circuit. However, oscillators using discrete transistors are more expensive than those using IC's when assembly labor costs are included. So, the selection of either circuit is a trade-off between better performance and cost.

The data and circuits presented here are intended to be used only as a guide for the designer. The networks described are generally simple and inexpensive and have all been observed to be functional. They only provide greater flexibility in the oscillator selection for CMOS-COPS systems.

Selecting Input/Output Options On COPS™ Microcontrollers

National Semiconductor
Application Note 401
Abdul Aleaf



AN-401

INTRODUCTION

There are a variety of user selectable input and output options available on COPS when the ROM is masked. These options are available to help the user tailor the I/O characteristics of the Microcontroller to the application. This application note is intended to provide the user a guide to the options: What are they? When and how to use which ones? The paper is generally written without reference to a specific device except when examples are given. It must be remembered that any given generic COPS Microcontroller has a subset of all the possible options available and that a given pin might not have all possible options. A reference to the device data sheet will determine which options are available for a specific device and a specific pin of that device.

INPUT/OUTPUT OPTIONS

Table I summarizes the I/O capability of NMOS-COPS, in general. However, some of the options have different configuration in CMOS-COPS. Data sheets provide information on the I/O options associated with the CMOS-COPS.

I. OUTPUTS

The following discussion provides detailed information on the capabilities of the mask-programmable output options available on COPS.

A. STANDARD OUTPUT

This option is a simple, straightforward, logic compatible output used for simple logic interface. It is available on SO, SK and all D and G outputs. It is recommended to be used as a default option for all but SO, SK outputs.

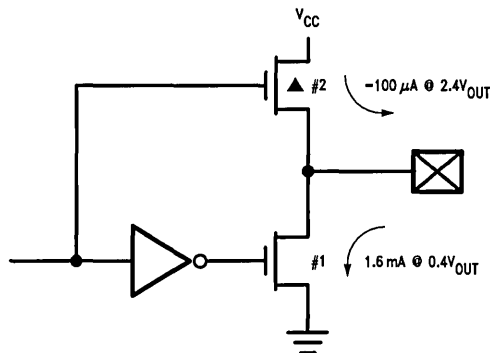


FIGURE 1. Standard Output

Figure 1 shows the standard output configuration. The enhancement mode device to ground is good at sinking current (sinks 1–2 mA) and is compatible with the

sinking requirement of 1 TTL load (1.6 mA at 0.4V). It will meet the "low" voltage requirement of CMOS logic. All output options use this device (device #1) for current sinking. On the other hand, the relatively high impedance depletion-mode device (device #2) to V_{CC} provides low current sourcing capability (100 μA at 2.4V). This pullup is sufficient to provide the source current for a TTL high level and will go to V_{CC} to meet the "high" voltage requirements of CMOS logic. An external resistor to V_{CC} may be required to interface to other external devices requiring higher sourcing capability.

An interface example to a common emitter NPN transistor is given below:

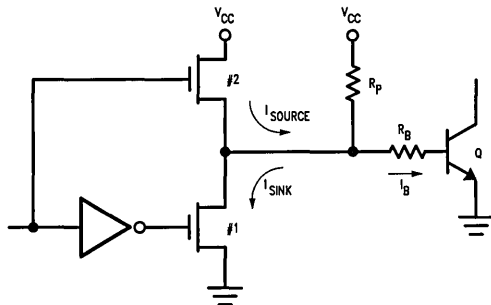


FIGURE 2

TL/DD/8440-2

R_B is needed to limit transistor's base current if I_{source} > I_{B(max)}.

R_P helps generate base drive if the I_{source} is not sufficient. The disadvantage of R_P is the introduction of more power dissipation. The temperature effects on the reverse saturation current I_{CB0} causes I_C to shift. I_{CB0} approximately doubles for every 10°C temperature rise. The effect of changes in I_{CB0} reduces off state margin and increases power dissipation in the off state.

However, in a typical device, the current supplied by R_P will swamp out any effects on I_{CB0}. Another parameter found to be decreasing linearly with temperature is V_{BE}:

$$\Delta V_{BE} = V_{BE2} - V_{BE1} = -k(T_2 - T_1)$$

where k ≈ 2 mV/°C, T in °C.

Now let's consider a practical example:

LOW SOURCE CURRENT OUTPUT:

Standard output, COP420, device #2.

The selected transistor is 2N3904.

DESIGN CONSIDERATIONS:

- Q is in saturation during ON-state.
- Q's collector current I_C = 100 mA

TABLE I

	Default	Standard	Push-Pull	High Sink	Very High Sink	LED	Hi-Current LED	TRI-STATE® Push-Pull	Hi Current TRI-STATE Push-Pull	Open Drain
SO	Push-Pull	Logic Compatible; Non MICROWIRE™	MICROWIRE Higher Drive, Faster X'sition							External Pull Up
SK	Push-Pull	Logic Compatible; Non MICROWIRE	MICROWIRE Higher Drive Faster Transition							External Pull Up
D	Standard	Logic Compatible		L Parts Only 15 mA	L Parts Only 30 mA					External Pull Up, Standard, Hi Sink or V.H.S. Pull Down
G	Standard	Logic Compatible; Inputs		L Parts Only 15 mA	L Parts Only 30 mA					External Pull-Up, Standard, Hi Sink or V.H.S. Pull Down
L	Standard	Logic Compatible; Inputs, TRI-LEVEL				Hi Source 1.5 mA TRI-LEVEL	L Parts Only Higher Source 3 mA TRI-LEVEL	MICROBUS™ Meets TRI-STATE Spec. TRI-LEVEL	L Parts Only Meets TRI-STATE Spec. TRI-LEVEL	External Pull Up TRI-LEVEL
H	Standard	Logic Compatible Inputs								External Pull Up
R	Standard	Logic Compatible; Inputs, TRI-LEVEL	Higher Drive Faster Transition TRI-LEVEL					Meets TRI-STATE Spec TRI-LEVEL		External Pull Up TRI-LEVEL

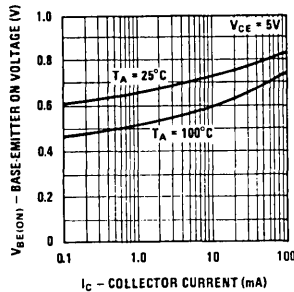
c. Assuming a "forced" β of 10 for Q. This is a standard value for β to insure saturation.

For an $I_C = 100$ mA, $\beta = 10$, we have $I_B \geq 10$ mA. The low current standard output certainly cannot provide $I_B \geq 10$ mA. Therefore, a pullup resistor (R_P) is required.

d. Now we need to select the minimum allowed value for R_P . The sinking ability of COPS output will determine R_P . We must sink the pullup current to a $V_{OUT} < V_{BE}$ in order to hold Q off. Also, note that

$$\frac{\Delta V_{BE}}{\Delta T} = -2 \text{ mV}/^\circ\text{C}.$$

e. Assuming the worst case is at V_{CC} (max) and High-temperature (let $\Delta T = 20^\circ\text{C} \Rightarrow \Delta V_{BE} = -40$ mV). From $V_{BE(ON)}$ Vs. I_C curve, Figure 3:



TL/DD/8440-3

FIGURE 3. 2N3904 I/V

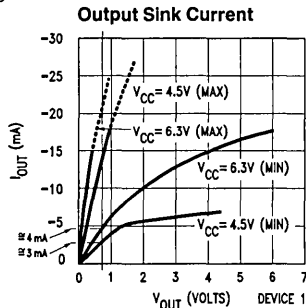
at 100 mA, 25°C , $V_{BE} \approx 0.85$ V.

So, our $V_{BE(45^\circ\text{C})} = 0.85 - 0.04 \approx 0.81$ V.

There is not margin here for process V_{BE} variations so we can allow 200 mV of slope,

$$V_{BE} = 0.61 \text{ V (worst case)}$$

f. Having $V_{BE} = 0.61$ V, we go to COPS sink graph and draw a vertical line at $V_{OUT} = V_{BE} = 0.61$ V. Figure 4 below:



TL/DD/8440-4

FIGURE 4

This will tell us, at $V_{OUT} = V_{BE}$, how much current can be sunk to keep Q "OFF". The intersection of $V_{CC} = 6.3$ (MIN) and $V_{BE} = 0.61$ V gives us $I_{sink} = 4$ mA.

g. Now calculate R_P .

$$R_P \geq \frac{6.3 - 0.61}{4} \text{ k} \geq 1.42 \text{ k}$$

$$\text{the actual standard } R_P (\pm 10\%) = \frac{1.42}{0.9} = 1.6 \text{ k} \pm 10\%$$

h. Using the value of R_P , let's calculate the current through R_P at $V_{CC} = 4.5$ V (MIN).

$$I_{R_P} = \frac{4.5 - 0.61}{1.42} \text{ mA} = 2.74 \text{ mA}$$

Which is less than sink ability of device (3 mA from Figure 4) at $V_{CC} = 4.5$ V, $V_{OUT} = 0.61$ V.

i. Now calculate the available source current. Here we use $V_{BE(max)}$ which is the worst case, and low temperature.

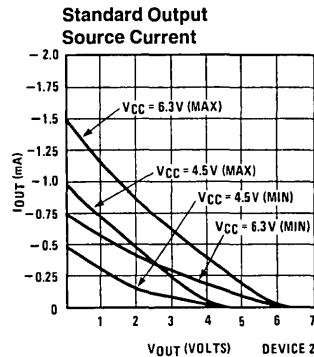
Let T (ambient) = 10°C .

From V_{BE} vs. I_C curve, Figure 3:

$$V_{BE} \approx 0.83 \text{ V at } 25^\circ\text{C}$$

$$V_{BE} \approx 0.83 + 2 \text{ mV}/^\circ\text{C} \times 15 = 0.86 \text{ V at } 10^\circ\text{C}.$$

Using this value of V_{BE} , we go to COP420 Standard Output source current curve (Figure 5), and draw a vertical line at $V_{BE} = 0.86$ V. The intersection of this line and $V_{CC} = 4.5$ (MIN) gives an $I_{source} = 325 \mu\text{A}$.



TL/DD/8440-5

FIGURE 5

This is low but typical of N-channel low current standard output.

Contribution of R_P

$$I_{R_P} = \frac{4.5 - 0.86}{(1.6)(1.1)} = 2.07 \text{ mA}$$

$$I_B(\text{min}) \approx 2.07 + 0.325 = 2.3 \text{ mA}$$

This is our worst case base drive, but we needed 10 mA.

What can we do to get the base drive we need?

1. We can use above design and allow Q to come out of saturation. The disadvantage is that Q's power dissipation increases.
2. Or use a Darlington configuration (Process 05). In such a configuration only first stage of Darlington can be saturated (not output stage). This will introduce a slightly higher power dissipation. Note that for a process 05 transistor, the forced β is 1000.
3. Use a high source type output such as TRI-STATE output. If we draw a vertical line at $V_{BE} = 0.86$, we get a source current of ≈ 6 mA at $V_{CC} = 4.5(\text{MIN})$ Figure 6, which gives us a worst case

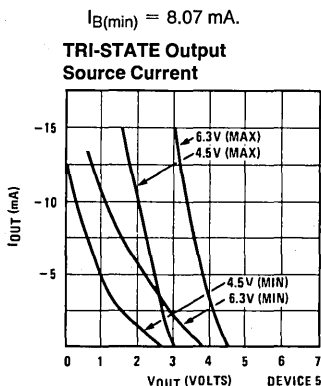


FIGURE 6

CAUTION On TRI-STATE graph the intersection of $V_{\text{out}} = V_{BE} = 0.86\text{V}$ and $V_{CC} = 6.3\text{V}(\text{MAX})$ curve (Figure 6) would result in an $I_{B(\text{MAX})} = 50\text{--}60 \text{ mA}$, which is way too much to handle. In this case there is a need for a series current limiting R_B to kill some of the worst case $I_{B(\text{max})}$.

4. There is a high current Standard-L option on some COPS (i.e., COP4XL, L-port) which provides sufficient source current.
5. N-channel output can generally sink better than source. PNP transistor can be used instead of NPN. The same analysis applies and in general will show better overdrive capabilities.

As shown in Figure 7, the D_0 output option, is driving the base of the PNP transistor. Assuming $V_{CC} = 4.5\text{V}$ (for COP402), $V_{BE} = 1.0\text{V}$, and a worst case base drive requirement of 3.0 mA . We see that we must supply $200 \mu\text{A}$ to the base-emitter resistor to turn the transistor on:

$$1.0\text{V}/5.1\text{k} = 200 \mu\text{A}$$

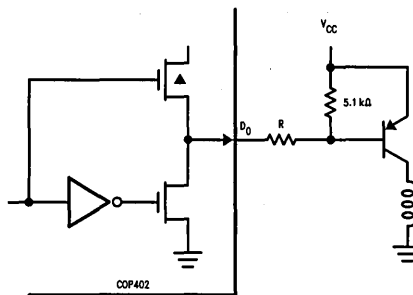


FIGURE 7. PNP Drive

From the output sink current curve on the COP402 data sheet, we find that, at 1.0V the D-line can sink 3.2 mA . To calculate the value of the current limiting resistor,

$$R = (V_{CC} - V_{BE} - V_{D0})/I$$

When $V_{CC} = 6.3\text{V}$, the D_0 output can sink more than enough current at 0.3V , and if the $V_{BE} = 0.7\text{V}$, we can calculate the maximum D_0 output current:

$$I = (V_{CC} - V_{BE} - V_D)/R \\ = (6.3 - 0.7 - 0.3)/780 = 6.3 \text{ mA.}$$

Using the Standard Output Option for Bidirectional I/O (G-port)

The standard output is good at sinking current, but rather weak at sourcing it. Therefore, by using the Standard Drive configuration and outputting 1's to the port, an external source may easily overdrive the port drivers with the added bonus of a built-in pullup. While the depletion-mode device provides sufficient current for a TTL high level, yet can be pulled low by an external source, thus allowing the same pin to be used as an input and output. Data written to the ports is statically latched and remains unchanged until rewritten. As inputs the lines are non-latching (Figure 8).

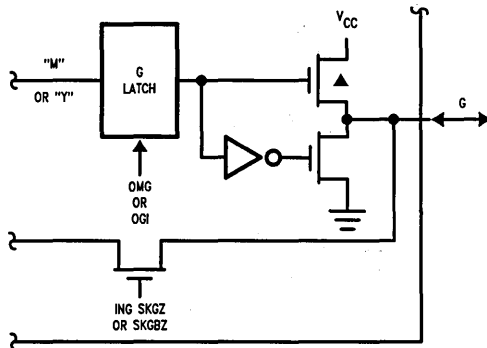


FIGURE 8. G Port Characteristics

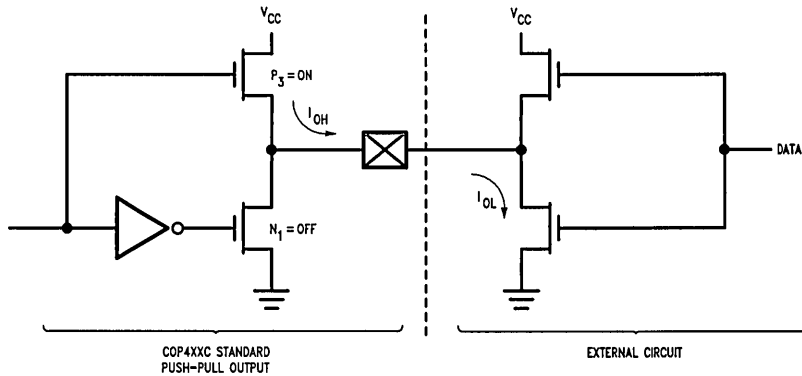


FIGURE 9

TL/DD/8440-9

When writing a "0" to the port, the enhancement-mode device to ground overcomes the high pullup and provides TTL current sinking capability. While writing a "1" the depletion-mode device behaves as internal pullup maintaining the "1" level indefinitely. In this situation, an input device capable of overriding the small amount of current supplied by the pull-up device can be read. This feature provides maximum user flexibility in selecting input/output lines with minimum external components.

In CMOS-COPS the low current push-pull output has even much weaker source current capability and this make it easier to be overridden.

Referring to *Figure 9*.

Note that $I_{OL} > I_{OH}$, otherwise transistors or buffers must be used.

For COP424C/444C, standard push-pull

$$\begin{aligned} @ V_{CC} = 4.5V, V_{out} = 0V, I_{OH(min)} &= 30 \mu A \\ I_{OH(max)} &= 330 \mu A \end{aligned}$$

$$\begin{aligned} @ V_{CC} = 2.4V, V_{out} = 0V, I_{OH(min)} &= 6 \mu A \\ I_{OH(max)} &= 80 \mu A \end{aligned}$$

While in NMOS (COP420L), Standard output:

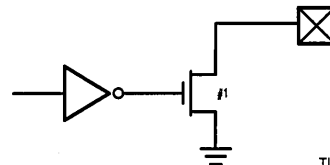
$$\begin{aligned} @ V_{CC} = 4.5V, V_{OH} = 2.0V, I_{OH(min)} &= 30 \mu A \\ I_{OH(max)} &= 250 \mu A \end{aligned}$$

$$\begin{aligned} @ V_{CC} = 6.3V, V_{OH} = 2.0V, I_{OH(min)} &= 75 \mu A \\ I_{OH(max)} &= 480 \mu A \end{aligned}$$

As we see, both in CMOS and NMOS it is easier to override I_{OH} . Note that the standard output option is available with standard, high, or very high sink current capability ("L" parts only). The pull-down device is bigger for the high/very high current standard output. The sourcing current is the same. These three choices provide some control over current capability.

B. OPEN-DRAIN OUTPUT

This option uses the same enhancement-mode device to ground as the standard output with the same current sinking capability. It does not contain a load device to V_{CC} , allowing external pullup as required by the user's application. The sinking ability of device # 1 determines the minimum allowed external pullup. The analysis discussed earlier for Standard Output options equally applies here. Available on SO, SK, and all D, G, and L outputs.



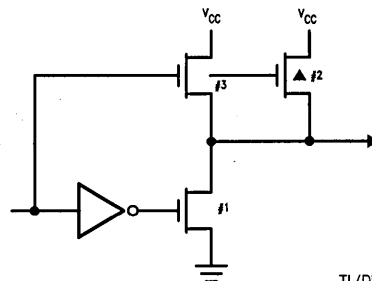
TL/DD/8440-10

FIGURE 10. Open-Drain Output

The open-drain option makes the ports G and L very easy to drive when they are used as inputs. This option is commonly used for high noise margin inputs, unusual logic level inputs as from a diode isolated keyboard, analog channel expansion, and direct capacitive touch-panel interface. Available with standard, high or very high sink capability ("L" parts only).

C. PUSH-PULL OUTPUT

The push-pull output differs from the standard output configuration in having an enhancement-mode device in parallel with the depletion-load device to V_{CC} , providing greater current sourcing capability (better drive) and faster rise and fall times when driving capacitive loads.



TL/DD/8440-11

FIGURE 11. Push-Pull Output

If a push-pull output is interfaced to an external transistor, a current-limiting resistor must be placed in series with the base of the transistor to avoid excessive source current flow out of the push-pull output. This option is generally for MICROWIRE Serial Data exchange.

It is available on SO, SK only and is recommended to be used as a default option for these outputs. A few points must be kept in mind when using SO, SK for MICROWIRE interface.

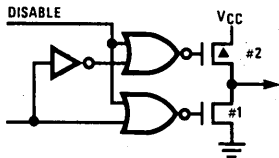
The data sheet specifies the propagation delay for a certain test condition (i.e., $V_{CC} = 5V$, $V_{OH} = 0.4V$, Loading = 50 pF, etc.).

In practice, actual delay varies according to actual input capacitive loading (typical 7–10 pF per IC input), total wire capacitance and PCB stray capacitance connected to the SI input. Thus, if actual total capacitive loading is too large to satisfy the delay time relationships ($t_d = t_{SK} - t_r$; t_d = actual delay time, t_{SK} = the instruction cycle time, t_r = the finite SK rise time), either slow down SK cycle time or add a pullup resistor to speed up SK "0" to "1" transition or use an external buffer to drive the large load. Besides the timing requirement, system supply and fan-out/fan-in requirements have to be considered, too.

If devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Briefly, for devices that have incompatible input levels or source/sink requirements to exchange data, external pullups or buffers are necessary to provide level shifting or driving. Unreliable operation might occur during data transfer, otherwise. For a 100 pF load, a standard COPS Microcontroller may use a 4.7k external resistor, with the output "low" level increased by less than 0.2V. For the same load the low power COPS may use a 22k resistor; with the SO, SK output "low" level increased by less than 0.1V.

D. STANDARD L OUTPUT

Same as Standard Output, but may be disabled. Available on L-outputs only.



TL/DD/8440-12

FIGURE 12. Standard L Output

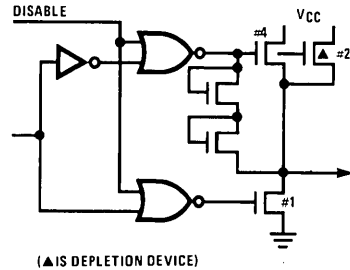
When this option is implemented on the L-port and the L-drivers are disabled to use the L lines as inputs, the disabled depletion-mode device cannot be relied on to source sufficient current to pull an input to a logic high. There are two ways to use L lines as inputs (having standard L option):

The first method requires that the drivers be disabled. In this case the lines are floating in an undefined state. The external circuitry must provide good logic levels both high and low to the input pins. The inputs are then read by the INL instruction. The second method is similar to the technique used for the G-port. The drivers are enabled and a "1" must be written to the Q register.

The external circuitry will then be required only to pull the lines low to a logic "0". The line will pull up to a "1" itself. The INL instruction is used as before to read the lines.

E. LED DIRECT DRIVE OUTPUT

In this configuration, the depletion-load device to V_{CC} is paralleled by an enhancement-mode device to V_{CC} to allow for the greater current sourcing capacity required by the segments of an LED display. Source current is clamped to prevent excessive source current flow.



TL/DD/8440-13

FIGURE 13. LED (L output) NMOS-COPS

This configuration can be disabled under program control by resetting bit 2 (EN^2) of the enable register to provide simplified display segment blanking.

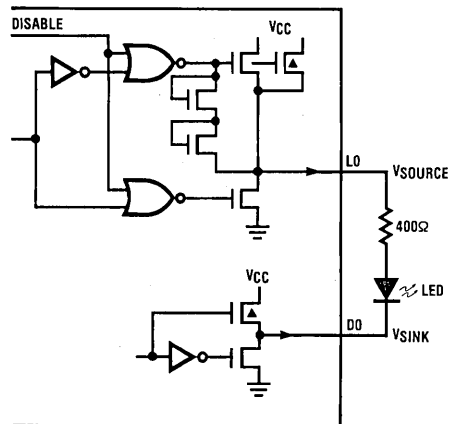
However, while both enhancement-mode devices are turned off in the disabled mode, the depletion-load device to V_{CC} will still source up to 0.125 mA. As in the case of Standard L output, again this current is not sufficient to pull an input to a logic "1".

The drivers must be disabled and the lines must be pulled high and low externally, whenever they are used as inputs.

Example #1:

When COPS outputs are used to drive loads directly, the power consumed in the outputs must be considered in the maximum power dissipation of the package.

Figure 14 shows an LED segment obtaining its source current from L_0 output and D_0 sinking the current. In this configuration all the power required to drive the LED with the exception of the portion consumed by the LED itself, is consumed within the chip. Assuming COP404L is the driving device:



TL/DD/8440-14

FIGURE 14. LED Drive

If we assume the V_{source} is not inserted, the device has a V_{CC} of 9.5V, and that the voltage drop across the LED is 2.0V.

We can calculate the power dissipation in these outputs. The minimum current that D_0 can sink at 1.0V is 35 mA (COP404L data sheet). L_0 can source up to 35 mA at 3.0V. Therefore, the power dissipation for the L_0 output could be: $(9.5 - 3.0)(0.035) = 227$ mW. The power in the D_0 output is $(1)(0.035) = 35$ mW.

Now let us calculate the current limiting resistor. Referring to COP404L L_0-L_7 output source current curves, at $V_{CC} = 9.5V$ the minimum current curve peaks at $I = 6.0$ mA and $V_{source} = 4.8V$. The current curve is actually very flat between 4.0 and 5.0 volts. For maximum current, we need to set the voltage on the L pin equal to 4.8V at 6.0 mA. The D line will sink this current at 0.4V. Therefore, the resistor and LED must make up the difference:

$$V_I = V_D + IR + V_{LED}$$

$$4.8 = 0.4 + 0.006R + 2.0$$

$$R = 400\Omega$$

At the other end of the curve, when the L line sources the maximum current, assume the LED and the D line will have the same voltage drop.

$$V_I = 0.4 + IR + 2.0$$

$$V_I = 2.4 + IR$$

From the current curve, we see that at 6.4V the L line will source 10 mA. Therefore: $V_I = 2.4 + (0.01)(400) = 6.4V$.

Example #2:

Let's consider a different configuration.

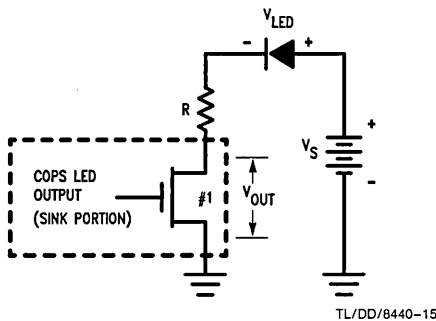


FIGURE 15. LED DRIVE

Now we calculate the series current limiting resistor R . The circuit has two non-linear devices to be considered; the output device and the LED.

The LED in this example is NSC5050. Looking at I/V curve, the device has a threshold 1.6V. Also, note that for $V_{LED} > 1.6V$ the I/V curve is very linear (Figure 17). Because of this, the LED characteristic can be modeled as a sharp threshold device with a non-zero source resistance (normally I/V curve is LOG looking).

From ON part of curve,

$$R_S = \frac{1.9 - 1.7}{0.05} = 4\Omega$$

We can neglect R_S as well (only $R_S \ll R$). Our model is simply a voltage source for the LED when

$$I = 0 \text{ for } V_{LED} < V_{TH}$$

$$I = \infty \text{ for } V_{LED} > V_{TH}$$

Design Procedure:

$$1. I_{LED(min)} = \frac{V_S(min) - (V_{LED(max)} + V_{OUT(max)})}{R(max)}$$

We need endpoints of the load line.

$$a. @V_{out} = 0 \Rightarrow I_{LED(min)} = \frac{V_S(min) - V_{LED(max)}}{R(max)}$$

$$b. @V_{out} + V_{LED(max)} = V_S \Rightarrow I = 0$$

$$(V_{LED(max)} = 2V)$$

2. Plot a and b

Assuming an $I_{min} = 7$ mA, $V_S(min) = 4.5V$ from 1 $R(max) = 357\Omega$

Draw the load line with slope $-1/357$ crossing

$$V_{out} = V_S - V_{LED(max)} = 4.5 - 2 = 2.5V.$$

(Figure 16).

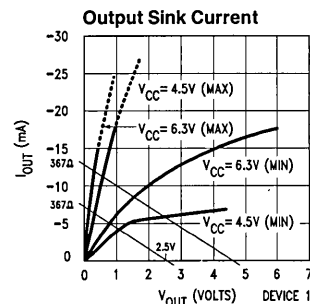


FIGURE 16. COP420

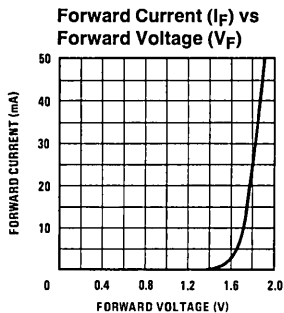


FIGURE 17. LED I/V Characteristic

The intersection of this load line and $V_{CC} = 4.5V$ (min) curve, we find an actual value of $I_{(min)} = 4.25$ mA. To determine I_{max} (at $R = 357\Omega$) we draw a parallel load line intersecting $V_{out} = 6.3 - 2.0 = 4.3V$ and find that @ $V_{CC} = 6.3V$, $I_{(max)} = 13$ mA.

3. From above calculations we observe that our $I_{(min)}$ (actual) is way off. Let's try to rotate our first load line around $V_{out} = 2.5V$ to increase I_{min} and then check I_{max} and R . (Figure 18).

Let's go for an I_{min} (actual) = 6 mA. This will give us $R = 89\Omega$ and the max. plot goes off the graph to = 36 mA.

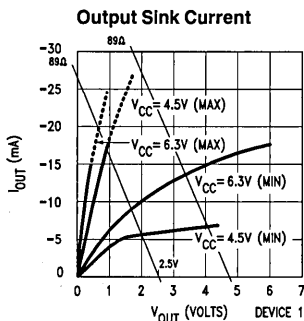


FIGURE 18. COP420

Comments:

1. The design must be a compromise between the two extremes (battery life should also be considered).
2. The lower the LED threshold the better. (The load line moves further up the device curve.)

F. TRI-STATE PUSH-PULL OUTPUT

This option is specifically available to meet the specifications of National's MICROBUS, outputting data over the data bus to a host CPU. It has two enhancement-mode devices to ground and V_{CC} .

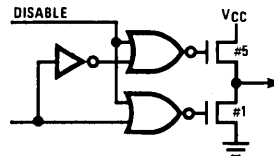


FIGURE 19. TRI-STATE Push Pull (L output)

The TRI-STATE logic can disable both enhancement-mode devices to free the MICROBUS data lines for input operation.

CAUTION Never try to pull against the TRI-STATE Output (too much source current) with the drivers enabled and Q register previously loaded with "1". The choices we have are mentioned earlier. Either TRI-STATE L-port or use Standard L output option.

II. INPUTS

COPS inputs may be programmed either with a depletion load device to V_{CC} or floating (Hi-Z input). All inputs are TTL/CMOS compatible. Hi-Z inputs should not be left floating; they should be connected to the output of a "high" and "low" driving device if active or to V_{CC} and ground if unused. Especially when using CMOS COPS (very high impedance inputs), the open inputs can float to any voltage. This will cause incorrect logic function and more power dissipation. Also, the CMOS inputs are more susceptible to static charge which causes gate oxide rupture and destroys the device. Unlike inputs, the outputs should be left open to allow the output switch without drawing any DC current. Another precaution is powering up the device. Never apply power to inputs or TRI-STATE outputs before both V_{CC} and ground are connected. This will forward bias input protection diodes, causing excessive diode currents. It will also power the device.

Special care must be practiced when interfacing a CMOS-COPS input to an analog IC, powered by different supply voltages. Avoid overvoltage conditions resulting

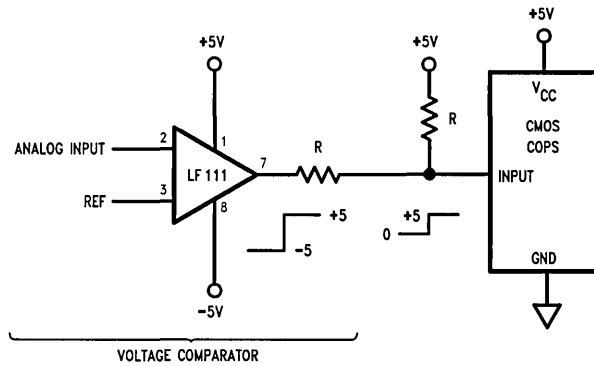


FIGURE 20

TL/DD/8440-20

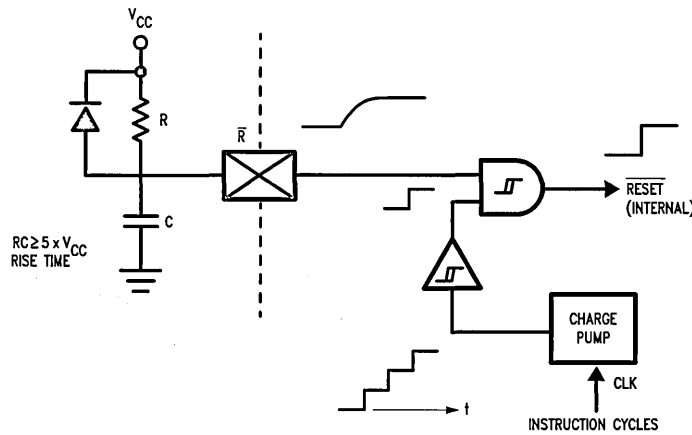


FIGURE 21

TL/DD/8440-21

from such situations. As an example, consider the interface of a CMOS-COPS with the LF111 voltage comparator:

When the low level “-5V” appears on the comparator’s output, the COPS input is pulled low below “logic low” of “0V”. This will cause damage if the comparator sinks enough current. The use of a current-limiting resistor in series with the input is helpful. A better solution is to use a voltage divider as shown in *Figure 20*. Any time a low level appears on the comparator’s output, a total voltage drop of 10V will appear across both resistors each dropping 5V, causing the input to sit at 0V. Whenever the output goes high, the resistors will not drop any voltage (no current through the resistors) and a logic high of 5V will appear on the input. To reduce power dissipation introduced by resistors, the resistor value must be high (> 100k), because the CMOS inputs have very high input impedance.

RESET INPUT

All COPS Microcontroller have internal reset circuitry. Internally there is an AND gate with one input coming from the RESET input, and the second input connected to a charge pump circuit. In the Charge pump circuit, a tiny capacitor is being charged upon execution of each internal instruction cycle. When the voltage across this inter-

nal capacitor reaches a high logic level, the second input of the AND gate is released.

The Reset logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μ s. With a slowly rising power supply, the part may start running before V_{CC} is within the guaranteed range. In this case, the user must provide an external RC network and diode shown in *Figure 21* above. The external RC network is there to hold the RESET pin below V_{IL} until V_{CC} reaches at least $V_{CC(min)}$. The desired response is shown in *Figure 22*.

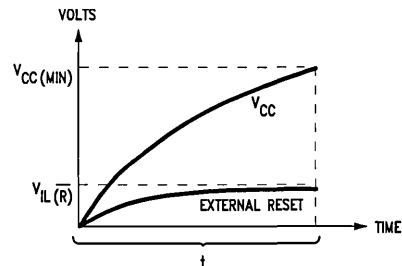


FIGURE 22

TL/DD/8440-22

t = 500-600 instruction cycles (8 msec)
for COPxxxL
t = 900-1000 instruction cycles (4 msec)
for COPxxxC

The diode is included in the reset circuitry to cause a "forced Reset" when the power supply goes away and recovers quickly. In such a situation the diode helps discharge the capacitor quickly. Otherwise, if the power failure occurs for a short time, the capacitor will not be fully discharged and the chip will continue operation with incorrect data.

Note that on the CMOS COPS, the internal charge pump circuitry can be disabled when using a very slow clock (<32 kHz) [option 23 = 1]. This is necessary, because one can run from DC to 4 μs instruction cycle time (fully static). In such a situation external RC network discussed earlier must be used.

INPUT PROTECTION DEVICES

All inputs and I/O pins have input protection circuitry. This circuitry is there regardless of any option selected. It is the first circuitry encountered at the pin.

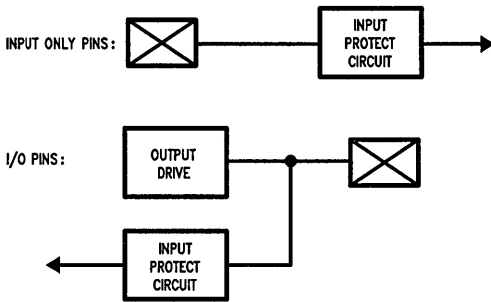
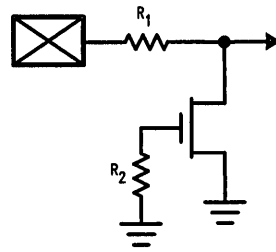


FIGURE 23

TL/DD/8440-23

For NMOS and XMOS devices, the circuits are of the form:

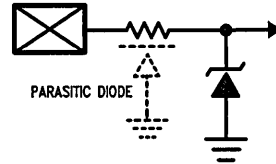


TL/DD/8440-24

FIGURE 24

This is a standard circuit defined for the process. R₁ is on the order of 200Ω. R₂ is around 300Ω (note that the R values are not precise).

This circuit is functionally equivalent to:



TL/DD/8440-25

FIGURE 25

The zener breakdown is around 10-15V; the gate breakdown is 50V.

CONCLUSION

All COPS Microcontrollers have a number of I/O options necessary to implement dedicated control functions in a wide variety of applications. The flexibility to select different options allows the user to tailor within limits, the I/O characteristics of the Microcontroller to the system. Thus, the user can optimize COPS for the system, thereby achieving maximum capability and minimum cost. This application note deals with the basic functionality of COPS I/O characteristics and does not address electrical differences among the various COPS devices.

New CMOS Vacuum Fluorescent Drivers Enable Three Chip System to Provide Intelligent Control of Dot Matrix VF Display

National Semiconductor
Application Note 440
Tom Markman



INTRODUCTION

Vacuum Fluorescent (VF) displays are becoming more and more common in a variety of applications. Manufacturers of everything from Automobiles to Video Recorders have taken advantage of these easy to read displays. VF displays are available in a wide variety of configurations; clock displays, calculator displays, multi-segment, and dot matrix displays are readily available at a low cost. This application note develops and covers in some detail a small CMOS system consisting of a single chip microcontroller and two display drivers which control a 20 character, 5 x 7 dot matrix VF display.

Figure 1 shows the schematic of the system. The microcontroller, a COPSM 424C, receives a character in ASCII form from the host system, stores the ASCII value of the character in its onboard RAM, converts the ASCII value to a 5 byte data word suitable for the display drivers and displays it on the VF display. The COPS also refreshes the display continuously while performing character update, much like a dumb terminal. Not including the address decoding logic, this application requires only the on-board RAM and ROM of the COPS424C, and National's MM58341 and MM58348 VF display drivers. If a steady message or a scrolling sentence is desired, only small changes in the COPS software are re-

quired. In this case the messages could be stored in the ROM of the COPS and the need for a host system would be eliminated.

VF DISPLAY AND VF DISPLAY DRIVER REQUIREMENTS

The display used in this application was an Itron #DC205G2. This 20 segment, 5 x 7 dot matrix, multiplexed display required a filament voltage of 5.7 Vac and a filament current of 37 mAac. The anode and grid voltages were supplied by the display drivers. The voltage and current requirements vary considerably for different displays depending on the size and number of characters, and the configuration (dot matrix, 7 segment, 14 segment, etc.). To determine the voltage requirements for a particular display, a simple calculation can be made. If maximum possible brightness of the display is desired, the following equation must be true:

$$E_t \geq E_b + E_k + (I_b)(R_{on}) \text{ where:}$$

E_t is the total Voltage of the display driver or $|V_{dis}| + V_{dd}$

E_k is the display Cathode Bias Voltage

$E_b = E_c$ is the typical Anode or Grid Voltage (V_{p-p})

I_b is the typical anode current (mA-p-p)

R_{on} is the display driver output impedance (Ω)

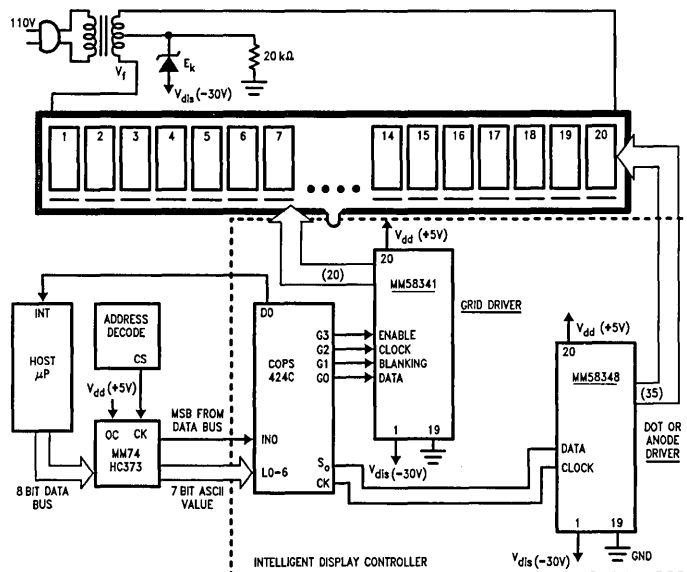


FIGURE 1. System Diagram Showing the Basic 3-Chip Display Controller and the Interface to a Microprocessor System

TL/F/8683-1

If the maximum brightness is not desired, the following equation can be used: $(E_t)(1.2) \geq E_b + E_k + (I_b)(R_{on})$. In this application, the calculated E_t was 42.25V, however, the display was legible under normal lighting conditions, with an E_t as low as 25V. If your display requires more than the 35V output of the MM58341 and MM58348, pin for pin compatible 60V VF Display Drivers (MM58241, MM58248) are available.

Figure 2 shows the relationship between the required VF display voltages. The cut-off voltage (E_k) is set by the Zener diode on the center tap of the filament transformer. This value is given in the VF display data sheet.

Avoiding Flicker and Pulsing

There are two different conditions which may cause the display to appear to flicker. The first is the refresh rate. This is particularly a problem on displays where the micro-controller must up-date more than 25 characters. Since the human

eye begins to notice flicker at about 40 Hz, a display with a refresh rate less than that will appear to be flashing on and off.

The second type of flicker occurs when the refresh rate is between 40 Hz and 90 Hz. In this case, the display will appear to be rolling rather than flashing. This condition occurs when the refresh rate and the filament frequency are close together. If a character is only on during the time when the filament voltage is negative, it will appear to be slightly brighter than the character next to it which may only be on during the positive cycle of the filament voltage. If this is the case, as it was in this application, the simplest solution is to increase the frequency of the filament. A DC oscillator circuit, such as the one shown in Figure 3, can be used to replace the AC voltage source. The filament frequency can be easily adjusted to eliminate this condition.

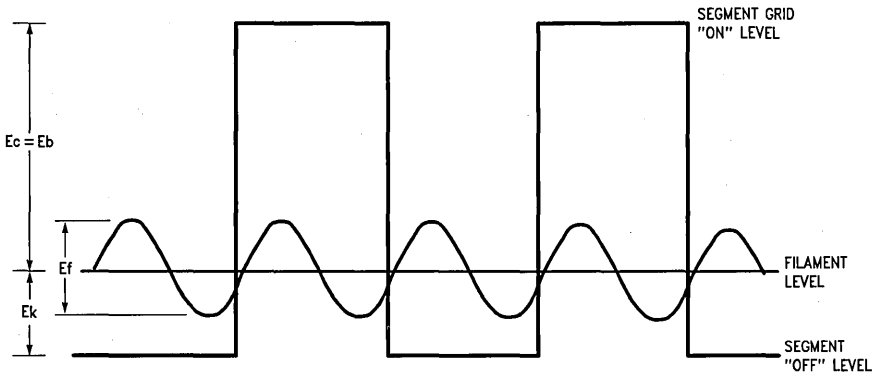


FIGURE 2. Voltage Levels for VF Display

TL/F/8683-2

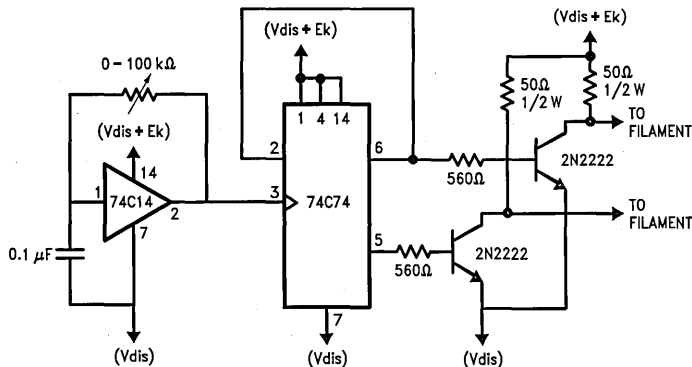


FIGURE 3. Filament Oscillator Circuit

TL/F/8683-3

VF Display Drivers

Two high voltage display drivers were needed to control the VF display. A MM58341, was used to control the grids and a MM58348 was used to control the individual pixels or anodes. Both of these drivers receive serial information and output 32 and 35 segments of data respectively.

The MM58341 has three control pins which make it ideal for controlling the grids of a VF display. The blanking control pin will turn off all segments of the display when a logic '1' is applied to this pin. This is particularly important for reducing ghosting, and controlling brightness. Ghosting is a condition where the last characters shadow appears behind the character being displayed. The enable pin acts as an envelope for the input signal. Only while it is at a logic '1' level will the circuit accept clock inputs. When the pin goes low, all the data is latched and displayed. A data out pin is also provided for cascading. If the display has more than 32 grids, a second grid driver can be cascaded by connecting the data out pin to the input data for the second grid driver.

The MM58348 is a 35 bit shift register and latch which is used to control each pixel or dot. When a leading 1, fol-

lowed by 35 bits of data, is received, the data is latched and displayed. The chip is automatically reset upon power up.

MULTIPLEXED DISPLAY REFRESH TIMING

Considering first the digit driver (MM58341), it becomes clear that the digits must be enabled or refreshed sequentially and that this process must be continuous regardless if the display data has changed. The data for the MM58341 is simply a 1 followed by 19 zeroes where the 1 is shifted through the internal registers of the MM58341. As each digit is enabled, the corresponding segment data is displayed. To insure that no ghosting effects are seen during the transition between digits, the blanking control is activated just before the data is latched into the dot or anode driver and deactivated just after the data has been latched. During this time when the blanking control is activated, the grid driver is clocked shifting the 1 to the next location. *Figure 4* shows the micro-controller waveforms and the resultant display waveforms for the 20 character display.

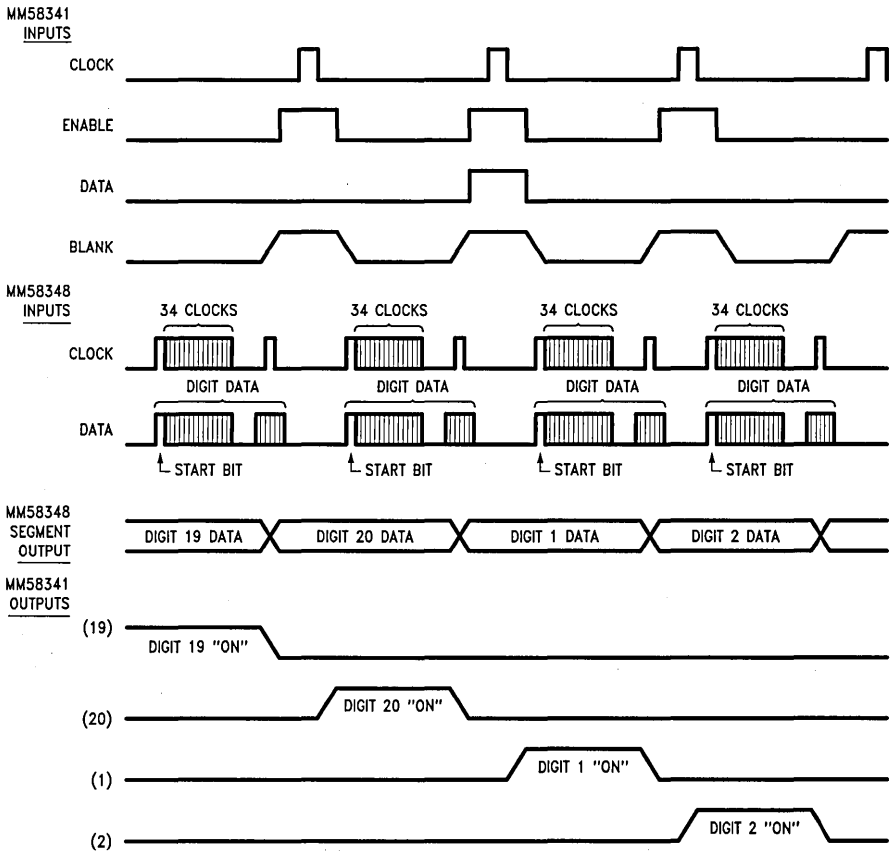


FIGURE 4. Timing Diagram

TL/F/8683-4

In between digit strobes, the segment data is updated. The first 34 bits of segment data are set up in the dot driver and the blanking signal is activated to disable all 20 digits. The 35th bit of data is clocked in, updating the segments. Since the MM58348 resets its internal shift register each time the data is latched, it can accept all but the final data bit while still displaying the previous digit. The digit driver is then clocked, shifting the digit strobe to the next position. The enable is then brought low, enabling the next digit. Finally blanking control is deactivated and the data displayed.

During the time which the blanking control is high, the order in which the segments or the digits are updated is not critical. Since this occurs while the display is blank. The digit driver may be clocked first, or the segments could be changed first. In general, the philosophy for driving this VF multiplexed display is outlined in *Figure 5*.

HOST INTERFACE AND PROGRAMMING

With a minimal amount of address decoding and an eight bit latch, COPS can be interfaced with a common microprocessor bus. When a character has been input into the host to be displayed, the ASCII value of that character is latched into the eight bit latch (MM74HC373) and is read on the L port (L0-6) of the COPS. The MSB of the ASCII value must be a logic 1. This MSB is the signal to the COPS that a new character is being presented. Once the character has been stored, an interrupt is sent from the COPS to the host through the D-0 port. The COPS checks for a new character being input every 200 μ s. If a character is being sent, 1 ms is required to store that character in the RAM of the COPS. With the COPS controlling the display, the host microprocessor is not being tied down with character look-up and display refresh. A simple flowchart of the host requirements is shown in *Figure 6*.

COPS SOFTWARE

There are four main sections of the COPS software. The first section, the initialization of the RAM, sets up the RAM as shown in *Figure 7*. A '0' is stored in all of the LSB positions and a '2' is stored in all of the MSB positions. Since the COPS is in a constant display loop, this is necessary to insure a blank display. 20H is the ASCII value of a space. With the RAM set up in this way, a maximum of 28 characters can be stored in RAM. Since the display in this application is only 20 characters long, RAM locations M1,4 to M1,11 and M3,4 to M3,11 are not used. RAM locations 1,12 to 1,15 and 3,12 to 3,15 are used as temporary storage throughout the program and cannot be used for character storage.

The second part of the program, stores the new characters sent by the host CPU in RAM. Once a character has been sent, this section of the program checks the ASCII value of that character to see if it is a control character or a display character. If it is a display character, the character is stored in RAM and an interrupt is sent to the host. There are three control characters which the COPS program will recognize. Cursor forward (ASCII value 08H) moves the cursor forward without destroying the data, cursor backwards (ASCII value 0CH) moves the cursor backwards without destroying the data, and return (ASCII value 0DH) will clear the display and put the cursor at the beginning of the display. To recognize and store a character, 1 ms is required.

The third part of the program, the display loop, is the heart of the program. Unless a new character has been detected, the program is always in this loop. This section does the

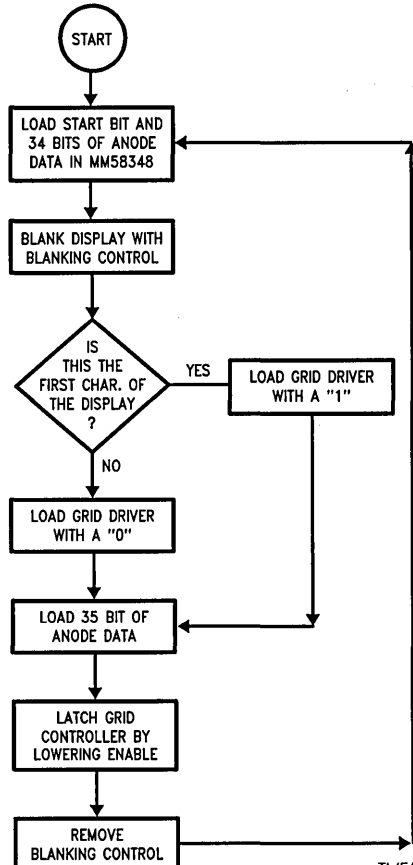


FIGURE 5. Flowchart for Display Drivers

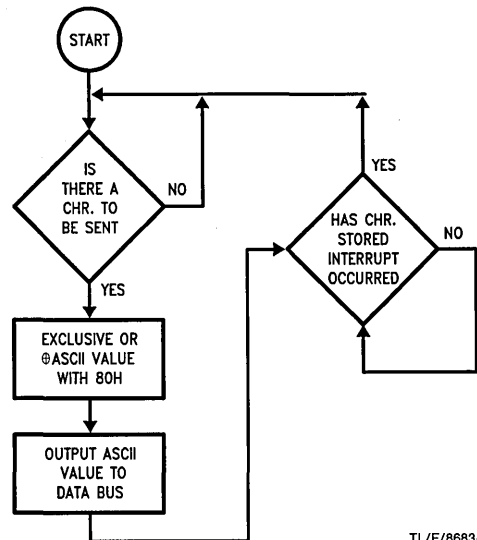


FIGURE 6. Host System Flowchart

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
LSB Chr 1	LSB Chr 2	LSB Chr 3	LSB Chr 4	LSB Chr 5	LSB Chr 6	LSB Chr 7	LSB Chr 8	LSB Chr 9	LSB Chr 10	LSB Chr 11	LSB Chr 12	LSB Chr 13	LSB Chr 14	LSB Chr 15	LSB Chr 16	M0		
MSB Pointer	LSB Pointer	Temp. ASCII STORAGE										LSB Chr 17	LSB Chr 18	LSB Chr 19	LSB Chr 20	M1		
MSB Chr 1	MSB Chr 2	MSB Chr 3	MSB Chr 4	MSB Chr 5	MSB Chr 6	MSB Chr 7	MSB Chr 8	MSB Chr 9	MSB Chr 10	MSB Chr 11	MSB Chr 12	MSB Chr 13	MSB Chr 14	MSB Chr 15	MSB Chr 16	M2		
Temp. Storage of Pointer														MSB Chr 17	MSB Chr 18	MSB Chr 19	MSB Chr 20	M3

FIGURE 7. COPS RAM Map

Matrix	PAD	Column 1	Column 2	Column 3	Column 4	Column 5	PAD			
Binary	0 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 0									
Hex.	13		EA		24		28		3E	

FIGURE 8

character font look-up, shifts the character data out the COPS serial port to the MM58348, and controls the MM58341 through the four bit parallel port (G0-4). Because the most significant nibble of the program counter is used as part of some COPS instructions, it is important that parts of the program are located at specific locations in ROM.

The final part of the program is the data. Each character is represented by a 5 byte data word. Each byte of the data word is stored at a different location in ROM. Fonts for characters with the ASCII values from 20H-5AH have already been stored in ROM. These characters can be changed or more characters can be added. The only limitation to the number of characters is the amount of available ROM.

CREATING THE 5 BYTE DATA WORD

Any number or combination of pixels or dots can be turned on at a time. To create a new character, it is easiest to first create a binary string which represents the character. A '1' in the binary string will turn on the pixel, a '0' will turn it off. To create this string, start in the upper left corner of the matrix and go down the columns.

The letter 'A' (Figure 9) would have a binary string shown in Figure 8. The data must be padded to make it an even 5 bytes in length. The pad at the beginning of the data (0001) is used as the leading 1 for the MM58348. The one bit pad at the end of the binary string must be a 0. If a 1 were sent as the pad, it would be used as the start bit for the next character.

The 5 byte data word that would be stored in ROM and represent the letter 'A' would then be 13EA24283E.

STORING THE DATA IN ROM

The 5 bytes of data are stored in 5 different locations in ROM. The first byte of data will be stored, LSB first, at location 200H plus the ASCII value of the character. For example, the ASCII value of the letter 'A' is 41H. The first byte of data for the letter 'A' would be stored, least significant bit first, at 241H. The second byte of data is stored at the location of the first data byte plus 60H or in this case at 2A1H. The location of the third byte is 40H plus the location of the

second byte. In this case, the third byte of data would be stored at 2E1H. The fourth byte of data is stored at 300H plus the ASCII value of the character or at 341H for the letter 'A'. The final byte of data is stored 40H from the fourth byte or at 381H. Remember the LSB of each byte is stored first. Table 1 shows the locations in ROM and the values stored in them for the letter 'A'.

This application shows a VF display controller designed with a minimum number of IC's. If additional information about VF displays or VF display drivers is required, refer to Application Note AN-371 (The MM58348/342/341/248/242/241 direct drive Vacuum Fluorescent (VF) Displays.

TABLE 1. Character Data of 'A' and Its Locations in ROM

Address In ROM	Data Stored
0241H	31
02A1H	AE
02E1H	42
0341H	82
0381H	E3

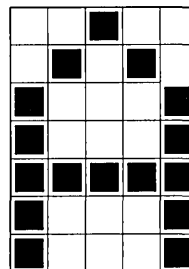


FIGURE 9. 5 x 7 Character as Stored in ROM

TL/F/8683-7

Section 1 of COPS Software

```

.CHIP 424C           ;DEFINES COPS CHIP
;THIS SECTION INITIALIZES THE RAM IN THE COPS BY LOADING A
;2 IN THE MSB AND A 0 IN THE LSB LOCATIONS OF EACH CHARACTER.
;IT ALSO STOPS THE CLOCK AND SETS THE POINTER AT THE FIRST
;CHARACTER OF THE DISPLAY.

```

```

RESET:              CLRA
                   LBI 3,15      ;LOADS A 2 IN ALL
                   JSR CLEAR2     ;MSB LOCATIONS
                   LBI 2,15      ;LOADS A 2 IN ALL
                   JSR CLEAR2     ;MSB LOCATIONS
                   LBI 1,15      ;LOADS A 0 IN ALL
                   JSR CLEAR      ;LSB LOCATIONS
                   LBI 0,15      ;LOADS A 0 IN ALL
                   JSR CLEAR      ;LSB LOCATIONS

                   CLRA          ;LOADS POINTER IN RAM
                   XAD 1,15      ;MSB IN 1,0F
                   CLRA
                   AISC 15       ;LSB IN 1,0E
                   XAD 1,14

                   RC            ;RESETS CARRY TO
                   XAS           ; STOP CLOCK
                   JMP START

```

```

CLEAR:              CLRA          ;CLEARS REGISTERS
                   XDS 0
                   JMP CLEAR
                   RET

```

```

CLEAR2:            CLRA          ;PUTS A 2 IN REGISTERS
                   AISC 02
                   XDS 0
                   JMP CLEAR2
                   RET

```

Section 2 of COPS Software

```

;THIS SECTION OF CODE IS ONLY EXECUTED WHEN A NEW
;CHARACTER HAS BEEN ENTERED. IF THE CHARACTER IS
;A CONTROL CHARACTER, THE CURSOR IS MOVED ACCORDINGLY,
;OTHERWISE THE CHARACTER IS STORED IN THE RAM OF THE COPS.

```

```

;NEW CHARACTER HAS BEEN ENTERED
NEW:               LBI 1,0C      ;DUMMY POINTER
                   INL          ;READS ASCII FROM
                   XIS 0        ;DATA BUS
                   X 0
                   LDD 1,0D
                   RC           ;CHAR. MSB=0 THEN YES
                   AISC 15      ;MSB<>0 THEN NO
                   JMP SPECIAL
                   AISC 01
                   LDD 1,0E      ;STORE ASCII IN RAM
                   CAB
                   LDD 1,0F
                   XABR
                   LDD 1,0C      ;MSB IN 1,0C
                   X 2
                   LDD 1,0D      ;LSB IN 1, 0D
                   X 0

```

Section 2 of COPS Software (Continued)

AN-440

```

JSR CURFOR
LBI 0,01      ;SENDS INTERRUPT TO
OBD          ; HOST. CHAR. IS
LBI 0,0      ; STORED IN RAM
OBD
JMP START
    
```

;SPECIAL CHARS. (CR, LF, CLEAR DISPLAY)

```

CURFOR:      LDD 1,0E      ;MOVES CURSOR FORWARD ONE
              COMP        ;SPACE. IF CURSOR IS
              AISC 01     ;MOVED BEYOND THE END OF
              JMP OK      ;DISPLAY, IT WRAPS AROUND
              AISC 0F     ;TO THE OTHER END. DATA IS
              XAD 1,0E    ;NOT DESTROYED BY MOVING
              CLRA        ;CURSOR
              AISC 01
              LBI 1,0F
              XOR
              JMP SKIP
OK:          COMP
SKIP:       LBI 1,0E
            X 0
            RET

CURBAC:     LDD 1,0F      ;MOVES CURSOR BACK ONE
            AISC 01      ;CHARACTER. DOES NOT
            JMP GOOD     ;DESTROY DATA AS IT IS MOVED
            LBI 1,0E    ;IF MOVED BEYOND THE
            CLRA        ;END OF THE DISPLAY IT
            AISC 01     ;WRAPS AROUND TO THE OTHER
            XOR         ;END
            X 0
            JMP START
GOOD:      XAD 1,0F
            JMP START

SPECIAL:   LDD 1,0C      ;CONTROL CHAR. HAS BEEN
            AISC 03      ;DETECTED
            JMP NOTRET
            JMP RESET    ;RETURN CLEARS DISPLAY, STARTS
                        ;PROGRAM OVER
NOTRET:    AISC 01      ;NOT RETURN, CHECK FOR CURSOR
            JMP CFOR     ;FORWARD
            JMP CURBAC   ;BY DEFAULT, CURSOR BACKWARDS

CFOR:      JSR CURFOR
            JMP START
    
```

;DISPLAY LOOP

3

Section 3 of COPS Software

;THIS IS THE DISPLAY LOOP OF THE PROGRAM. UNLESS A NEW CHARACTER
 ;HAS BEEN ENTERED AND IS BEING STORED, THE PROGRAM IS ALWAYS IN
 ;THIS DISPLAY LOOP. IT LOOKS UP THE CHARACTER FONT, SHIFTS THE
 ;CHARACTER DATA OUT THE SERIAL PORT AND CONTROLS THE GRID DRIVER.

```

START:      LBI 2,15      ;DISPLAY LOOP POINTER
            JSR HERE     ;GOTO DISPLAY LOOP
            LBI 3,03     ;SECOND DISPLAY LOOP POINTER
            JSR HERE     ;GOTO DISPLAY LOOP
            OGI 09      ;LOADS A 1 IN GRID DRIVER
            OGI 0D
            OGI 09

            JMP START

            ;CHECKS FOR NEW CHAR

HERE:       RC
            ININ
            AISC 15
            JMP OLDCHR
            JMP NEW

            ;DISPLAY LOOP FOR OLD CHAR AND
            ; LOOK UP

OLDCHR:     LD 2         ;LOOKS UP FIRST BYTE OF CHR.FONT
            JSR DATA4  ; 200H+ASCII VALUE
            AISC 06     ;ADDS 06H TO MSB OF ASCII
            JSR DATA2  ;LOOKS UP SECOND BYTE OF CHR FONT
            AISC 0A     ;ADDS 0AH TO MSB OF ASCII
            JSR DATA2  ;LOOKS UP THIRD BYTE OF CHR. FONT
            JSR DATA3  ;LOOKS UP THIRD BYTE OF CHR. FONT
                    ; AT 300H+ASCII VALUE
            AISC 06     ;ADDS 06H TO MSB OF ASCII VALUE
            OGI 02     ;TURNS ON BLANKING CONTROL
            JSR DATA3  ;LOOKS UP LAST BYTE OF CHR. FONT
            ;CLOCKS A 0 IN GRID DRIVER

            OGI 0A     ;ENABLE,BLANKING CONTROL
            OGI 0E     ;ENABLE,BLANKING CONTROL,CLOCK
            OGI 0A     ;ENABLE,BLANKING CONTROL
            OGI 00     ;A 0 SHIFTED IN

            LD 0
            XDS 2
            JMP HERE
            RET

RIGHT:      LBI 3,15
            CQMA
            JSR SHIFT   ;OUTPUTS A
            X 0         ;NEW DATA
            JSR SHIFT   ;OUTPUTS A
            LEI 01      ;COUNTER MODE
            LDD 3,14    ;1,0 IN A
            XABR        ;A IN BR
            LDD 3,13    ;1,1 IN A
            CAB         ;A IN BD
            LD 2
            RET
  
```

Section 3 of COPS Software (Continued)

```

POINTER:      LEI 01          ;COUNTER MODE
               XAS           ;A IN SIO
               XABR          ;BR IN A
               AISC 02       ;ADD 2
               XAD 3,14      ;A IN 1,0
               CBA           ;BD IN A
               XAD 3,13      ;A IN 1,1
               LBI 3,15
               XAS           ;SIO IN A
               LEI 08        ;SERIAL MODE
               JMP RIGHT
;SHIFTS OUT SERIAL PORT

SHIFT:        LEI 08        ;THIS ROUTINE SHIFTS THE DATA
               SC           ;FROM THE SI/O REGISTER OUT
               XAS          ;THE SERIAL PORT WITH EACH
               NOP          ;CLOCK CYCLE
               NOP
               RC
               XAS
               RET

.=0200
DATA3:        LQID
               JMP RIGHT
DATA4:        LQID
               JMP POINTER

.=0300
DATA3:        LQID
               JMP RIGHT

```

Section 4 of COPS Software

;THE CHARACTER FONTS FOR THE CHARACTERS WITH ASCII VALUES BETWEEN 20H AND 5AH HAVE BEEN STORED IN THIS SECTION OF THE PROGRAM.

```

;DATA FOR FIRST 2 BYTES OF EACH
; CHAR.

```

```

.=0220
.WORD 001, 001, 001, 021, 021, 0C1, 061, 001
.WORD 031, 001, 041, 011, 001, 011, 001, 001
.WORD 071, 001, 041, 081, 011, 0E1, 031, 081
.WORD 061, 061, 001, 001, 001, 021, 001, 041
.WORD 071, 031, 081, 071, 081, 0F1, 0F1, 071
.WORD 0F1, 081, 081, 0F1, 0F1, 0F1, 0F1, 071
.WORD 0F1, 071, 0F1, 061, 081, 0F1, 0F1, 0F1
.WORD 0C1, 0C1, 081

```

Section 4 of COPS Software (Continued)

```
;DATA FOR SECOND 2 BYTES OF EACH
; CHAR.
```

```
.=0280
```

```
.WORD 000, 000, 0C1, 0F9, 0A4, 095, 02D, 000
.WORD 088, 000, 054, 020, 000, 020, 000, 014
.WORD 01D, 082, 003, 005, 058, 045, 0AC, 001
.WORD 02D, 023, 000, 000, 020, 058, 001, 001
.WORD 00D, 0AE, 0F3, 00D, 0F3, 02F, 02F, 00D
.WORD 02E, 003, 00D, 02E, 00E, 08E, 08E, 00D
.WORD 02F, 00D, 02F, 025, 001, 00C, 008, 00C
.WORD 056, 040, 017
```

```
;THIRD 2 BYTES OF DATA FOR EACH CHAR.
```

```
.=02C0
```

```
.WORD 000, 0E3, 000, 0AC, 0FB, 040, 0A5, 083
.WORD 00A, 002, 0F3, 0F1, 034, 040, 008, 040
.WORD 046, 0F7, 02E, 046, 021, 086, 046, 02E
.WORD 046, 046, 0A0, 0B4, 0A0, 0A0, 015, 022
.WORD 0E6, 042, 04E, 006, 00E, 046, 042, 046
.WORD 040, 0F7, 006, 0A0, 004, 080, 0E0, 006
.WORD 042, 026, 062, 046, 0F3, 004, 008, 034
.WORD 040, 070, 046
```

```
;FOURTH TWO BYTES OF DATA FOR EACH CHAR.
```

```
.=0320
```

```
.WORD 000, 008, 007, 0F7, 0AA, 031, 028, 000
.WORD 008, 02A, 049, 080, 000, 080, 000, 001
.WORD 01D, 018, 09C, 09D, 0F7, 01D, 09C, 084
.WORD 09C, 0AC, 000, 000, 022, 041, 041, 08C
.WORD 0DC, 082, 09C, 01C, 01C, 09C, 084, 09C
.WORD 080, 01C, 0EF, 022, 018, 002, 020, 01C
.WORD 084, 02C, 0A4, 09C, 00C, 018, 028, 010
.WORD 041, 009, 01D
```

```
;LAST BYTES OF DATA FOR EACH CHAR.
```

```
.=0380
```

```
.WORD 000, 000, 000, 082, 084, 064, 0A0, 000
.WORD 000, 083, 044, 001, 000, 001, 000, 004
.WORD 0C7, 020, 026, 0CC, 080, 0C9, 0C8, 00E
.WORD 0C6, 087, 000, 000, 028, 082, 001, 006
.WORD 027, 0E3, 0C6, 044, 0C7, 028, 008, 0C5
.WORD 0EF, 028, 008, 028, 020, 0EF, 0EF, 0C7
.WORD 006, 0A7, 026, 0C4, 008, 0CF, 08F, 0CF
.WORD 06C, 00C, 02C
```

```
.END
```

MICROWIRE™ Serial Interface

National Semiconductor
Application Note 452
Abdul Aleaf



AN-452

INTRODUCTION

MICROWIRE is a simple three-wire serial communications interface. Built into COPS™, this standardized protocol handles serial communications between controller and peripheral devices. In this application note are some clarifications of MICROWIRE logical operation and of hardware and software considerations.

LOGICAL OPERATION

The MICROWIRE interface is essentially the serial I/O port on COPS microcontrollers, the SIO register in the shift register mode. SI is the shift register input, the serial input line to the microcontroller. SO is the shift register output, the serial output line to the peripherals. SK is the serial clock; data is clocked into or out of peripheral devices with this clock.

It is important to examine the logical diagram of the SIO and SK circuitry to fully understand the operation of this I/O port (Figure 1).

The output at SK is a function of SYNC, EN0, CARRY, and the XAS instruction. If CARRY had been set and propagated to the SKL latch by the execution of an XAS instruction, SYNC is enabled to SK and can only be overridden by EN0 (Figure 2). Trouble could arise if the user changes the state of EN0 without paying close attention to the state of the latch in the SK circuit.

If the latch is set to a logical high and the SIO register enabled as a binary counter, SK is driven high. From this state, if the SIO register is enabled as a serial shift register, SK will output the SYNC pulse immediately, without any intervening XAS instruction.

The SK clock (SYNC pulse) can be terminated by issuing an XAS instruction with CARRY = 0 (Figure 3).

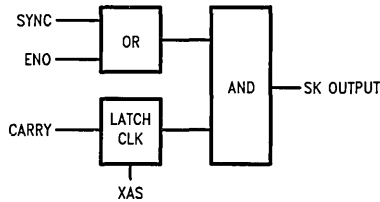


FIGURE 1. Logical Diagram of SK Circuit

TL/DD/8796-1

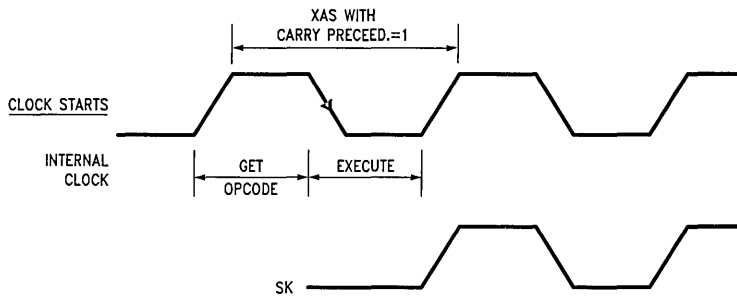


FIGURE 2. SK Clock Starts

TL/DD/8796-2

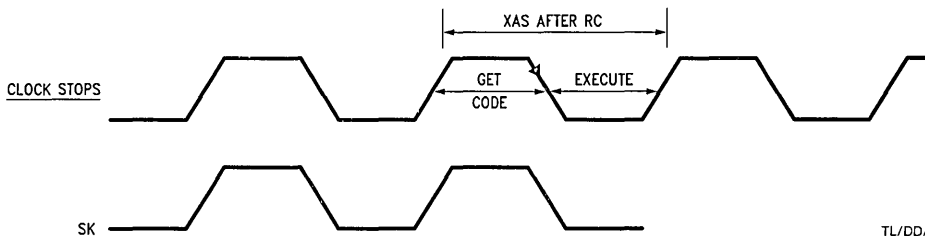


FIGURE 3. SK Clock Stops

TL/DD/8796-3

The SIO register can be compared to four master-slave flip-flops shown in Figure 4. The masters are clocked by the rising edges of the internal clock. The slaves are clocked by the falling edges of the internal clock. Upon execution of an XAS, the outputs of the masters are exchanged with the contents of the accumulator (read and overdrive) in such a way that the new data are present at the inputs of the four slaves when the falling edge of the internal clock occurs. The content of the accumulator is, therefore, latched respectively in the four slave flip-flops and bit 3 appears directly on SO.

This means that:

- a) SO will be shifted out upon the falling edges of SK and will be stable during rising edges of SK.

- b) SI will be shifted in upon the rising edge of SK, and will be stable when executing, i.e., an XAS instruction.

The shifting function is automatically performed on each of the four instruction cycles that follow an XAS instruction (Figure 5).

When the SIO register is in the shift register mode (EN0 = 0), it left shifts its contents once each instruction cycle. The data present on the SI input is shifted into the least significant bit (bit 0) of the serial shift register. SO will output the most significant bit of the SIO register (bit 3) if EN3 = 1. Otherwise, SO is held low. The SK is a logic controlled clock which issues a pulse each instruction cycle. To ensure that the serial data stream is continuous, an XAS instruction must be executed every fourth time. Serial I/O timing is related to instruction cycle timing in the following way:

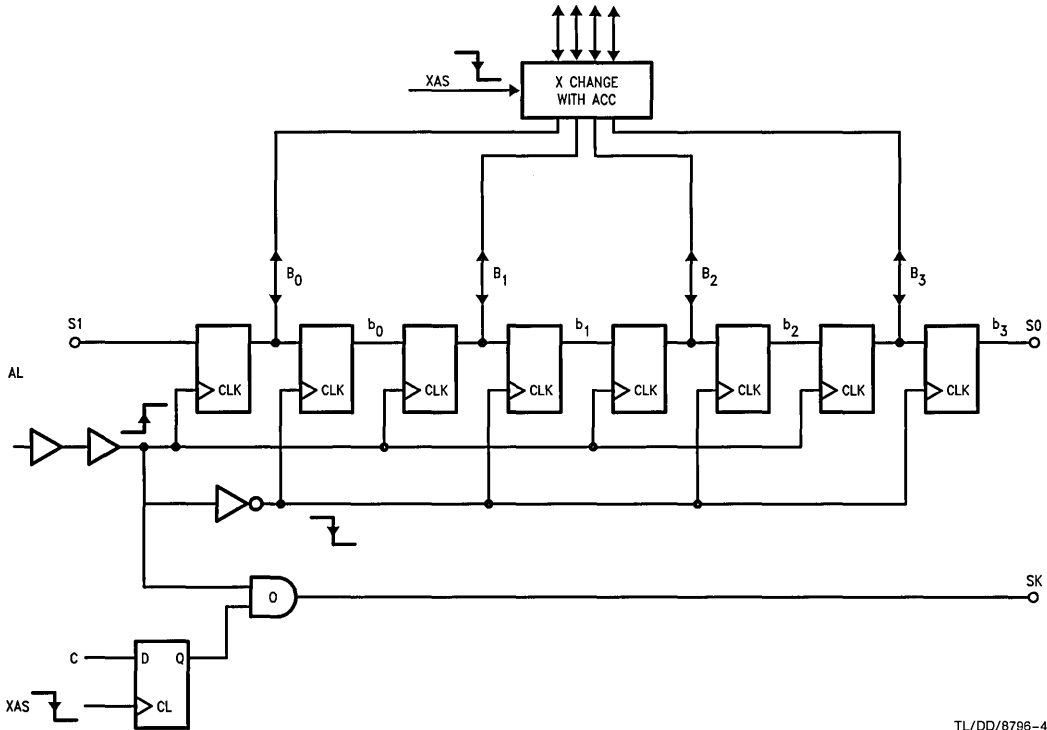


FIGURE 4

TL/DD/8798-4

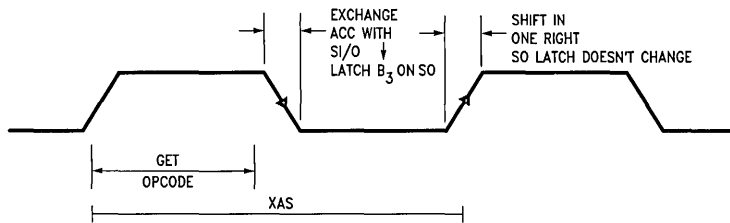


FIGURE 5. XAS Sequence

TL/DD/8796-5

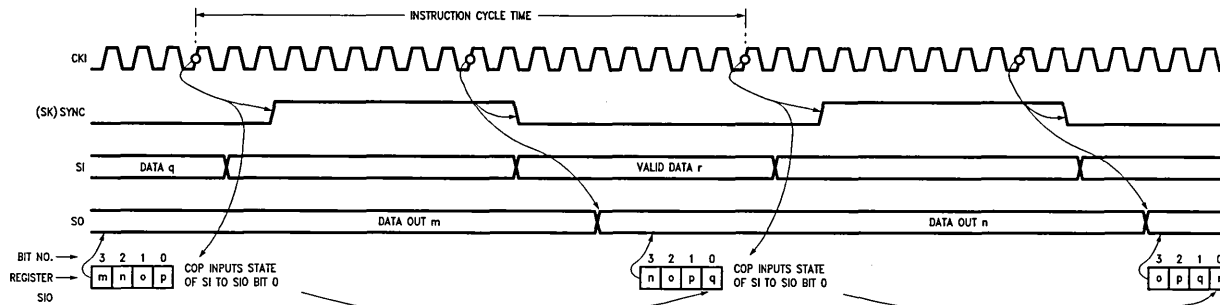


FIGURE 6. Serial I/O Timing

TL/DD/8796-6

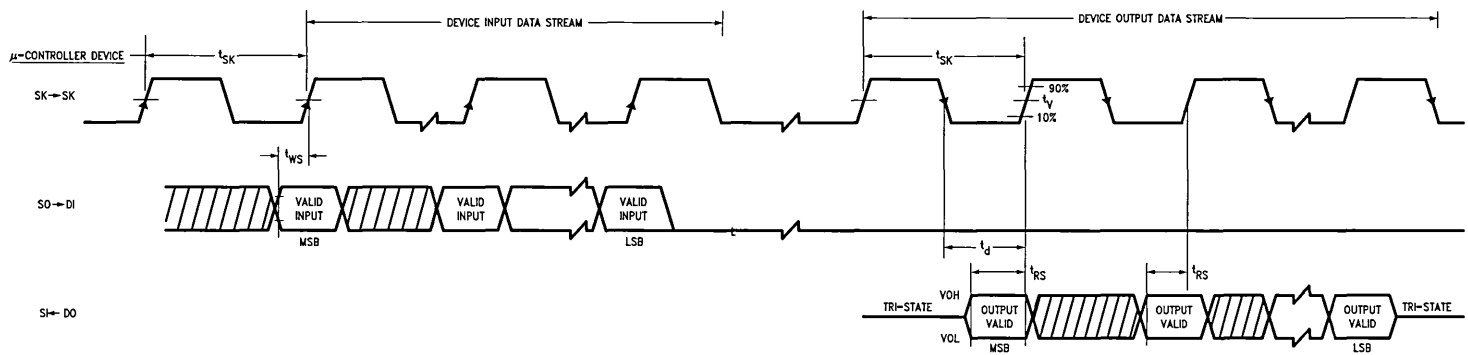


FIGURE 7. MICROWIRE Serial Data Exchange Timing

TL/DD/8796-7

To write to device: $t_{ns} > t_{setup}$

To read from device: $t_d \ll t_{SK} - t_r$; $t_{RS} > t_{SK}/4$

Where: t_{ws} is MICROWIRE write data-in (DI) setup time,

t_{setup} is device data sheet min data setup time to latch in valid data,

t_{SK} is system clock (SK) cycle time (Recommended 50% duty cycle),

t_r is rise time (10% to 70% bout) of system clock (SK),

t_d is device actual delay time before data-out (DO) valid and

t_{RS} is minimum data setup time for controller to shift-in valid data

The first clock rising edge of the instruction cycle triggers the low-to-high transition of SYNC output via SK. At this time, the processor reads the state of SI into SIO bit 0, shifting the current bits 0–2 left. Halfway through the cycle (shown in *Figure 6* as the eight clock rising edge), SK is reset low and the new SIO bit 3 is outputted via SO.

INTERFACING CONSIDERATIONS

To ensure data exchange, two aspects of interfacing have to be considered: 1) serial data exchange timing; 2) fan-out/fan-in requirements. Theoretically, infinite devices can access the same interface and be uniquely enabled sequentially in time. In practice, however, the actual number of devices that can access the same serial interface depends on the following: system data transfer rate, system supply requirement capacitive loading on SK and SO outputs, the fan-in requirements of the logic families or discrete devices to be interfaced.

HARDWARE INTERFACE

Provided an output can switch between a HIGH level and a LOW level, it must do so in a predetermined amount of time for the data transfer to occur. Since the transfer is strictly synchronous, the timing is related to the system clock (SK) (*Figure 7*). For example, if a COPS controller outputs a value at the falling edge of the clock and is latched in by the peripheral device at the rising edge, then the following relationship has to be satisfied:

$$t_{\text{DELAY}} + t_{\text{SETUP}} \leq t_{\text{CK}}$$

where t_{CK} is the time from data output starts to switch to data being latched into the peripheral chip, t_{SETUP} is the setup time for the peripheral device where the data has to be at a valid level, and t_{DELAY} is the time for the output to read the valid level. t_{CK} is related to the system clock provided by the SK pin of the COPS controller and can be increased by increasing the COPS instruction cycle time.

The maximum t_{SETUP} is specified in the peripheral chip data sheets. The maximum t_{DELAY} allowed may then be derived from the above relationship.

Most of the delay time before the output becomes valid comes from charging the capacitive load connected to the output. Each integrated circuit pin has a maximum load of 7 pF. Other sources come from connecting wires and connection from PC boards. The total capacitive load may then be estimated. The propagation delay values given in data sheets assume particular capacitive loads (e.g. $V_{\text{CC}} = 5\text{V}$, $V_{\text{OH}} = 0.4\text{V}$, loading = 50 pF, etc.).

If the calculated load is less than the given load, those values should be used. Otherwise, a conservative estimate is to assume that the delay time is proportional to the capacitive load.

If the capacitive load is too large to satisfy the delay time criterion, then three choices are available. An external buffer may be used to drive the large load. The COPS instruction cycle may be slowed down. An external pullup resistor may be added to speed up the LOW level to HIGH level transition. The resistor will also increase the output LOW level and increase the HIGH level to LOW level transition time, but the increased time is negligible as long as the output LOW level changes by less than 0.3V. For a 100 pF load, the standard COPS controller may use a 4.7k external resistor, with the output LOW level increased by less than

0.2V. For the same load, the low power COPS controller may use a 22k resistor, with the SO and SK LOW levels increased by less than 0.1V.

Besides the timing requirements, system supply and fan-out/fan-in requirements also have to be considered when interfacing with MICROWIRE. For the following discussion, we assume single supply push-pull outputs for system clock (SK) and serial output (SO), high-impedance input for serial input (SI).

To drive multi-devices on the same MICROWIRE, the output drivers of the controller need to source and sink the total maximum leakage current of all the inputs connected to it and keep the signal level within the valid logic "1" or logic "0". However, in general, different logic families have different valid "1" and "0" input voltage levels. Thus, if devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Similarly, devices with TRI-STATE® outputs, when connected to the SI input, must satisfy the minimum valid input level of the controller and the maximum TRI-STATE leakage current of all outputs.

So, for devices that have incompatible input levels or source/sink requirements, external pull-up resistors or buffers are necessary to provide level-shifting or driving.

SOFTWARE INTERFACE

The existing MICROWIRE protocol is very flexible, basically divided into two groups:

1) 1AAA.....ADD.....D

where leading 1 is the start bit and leading zeroes are ignored.

AAA.....A is device variable instruction/address word.

DDD.....D is variable data stream between controller and device.

2) No start bit, just bit stream, i.e., bbb.....b

where b is a variable bit stream. Thus, device has to decode various fields within the bit stream by counting exact bit position.

SERIAL I/O ROUTINES

Routines for handling serial I/O are provided below. The routines are written for 16-bit transmissions, but are trivially expandable up to 64-bit transmissions by merely changing the initial LBI instruction. The routines arbitrarily select register 0 as the I/O register. It is assumed that the external device requires a logic low chip select. It is further assumed that chip select is high and SK and SO are low on entry to the routines. The routines exit with chip select high, SK and SO low. GO is arbitrarily chosen as the chip select for the external device.

SERIAL DATA OUTPUT

This routine outputs the data under the conditions specified above. The transmitted data is preserved in the microcontroller.

```
OUT2: LBI 0,12 ; point to start of
          data word
          SC
          OGI 14 ; select the external
          device
```

TABLE I. MICROWIRE Standard Family

Features	Part Number					
	DS8908	MM545X	COP472-3	COP430 (ADC83X)	NM93C06A	
GENERAL						
Chip Function	AM/PM PLL	LED Display Driver	LCD Display Driver	A/D	E ² PROM	
Process	ECL	NMOS	CMOS	CMOS	NMOS	
V _{CC} Range	4.75V–5.25V	4.5V–11V	3.0V–5.5V	4.5V–0.3V	4.5V–5.5V	
Pinout	20	40	20	8/14/20	14	
HARDWARE INTERFACE						
Min V _{IH} /Max V _{IL}	2.1V/0.7V	2.2V/0.8V	0.7V _{CC} /0.8V	2.0V/0.8V	2.0V/0.8V	
SK Clock Range	0–625 kHz	0–500 kHz	4–250 kHz	10–200 kHz	0–250 kHz	
Write Data DI	Setup Min	0.3 μs	0.3 μs	1 μs	0.2 μs	0.4 μs
	Hold Min	0.8 μs	(3)	100 ns	0.2 μs	0.4 μs
Read Data Prop Delay		(Note 4)	(Note 3)	(Note 3)	(Note 3)	
Chip Enable	Setup	0.3 μs	0.4 μs	1 μs (Note 1)	0.2 μs	0.2 μs
	HOLD	0.8 μs	(Note 3)	1 μs (Note 2)	0.2 μs	0
Max Frequency Range	AM	8 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)
	FM	120 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)
SOFT						
Serial I/O Protocol		11D1...D20	1D1...D35	b1...b40	1xxx	1AA...DD
Instruction/Address Word		None	None	None	(Note 4)	(Note 4)

Note 1: Reference to SK rising edge.

Note 2: Reference to SK falling edge.

Note 3: Not defined.

Note 4: See data sheet for different modes of operation.

```

LEI 8 ; enable shift
      register mode
JP SEND2
SEND1: XAS
SEND2: LD ; data output loop
XIS
JP SEND1
XAS ; send last data
RC
CLRA
NOP
XAS ; turn SK clock off
OGI 15 ; deselect the device
LEI 0 ; turn S0 low
RET
    
```

The code for reading serial data is almost the same as the serial output code. This should be expected because of the nature of the SIO register and the XAS instruction.

MICROWIRE STANDARD FAMILY

A whole family of off-the-shelf devices exists that is directly compatible with MICROWIRE serial data exchange standard. This allows direct interface with the COPS family of microcontrollers.

Table I provides a summary of the existing devices and their functions and specifications.

TYPICAL APPLICATION

Figure 8 shows pin connection involved in interfacing an E²PROM with the COP420 microcontroller.

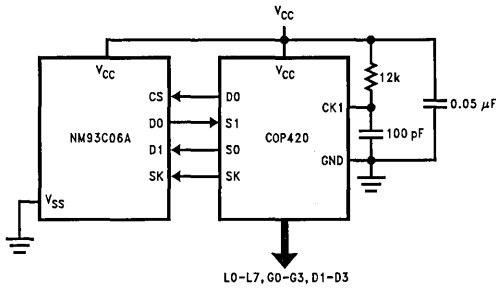


FIGURE 8. NM93C06A COP420 Interface

The following points have to be considered:

1. For NM93C06A the SK clock frequency should be in the 0 kHz–250 kHz range. This is easily achieved with COP420 running at 4 μs–10 μs instruction cycle time (SK period is the COP420 instruction cycle time). Since the minimum SK clock high time is greater than 1 μs, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max.
2. CS low period following an E/W instruction must not exceed 30 ms maximum. It should be set at typical or minimum spec of 10 ms. This is easily done in software using the SKT timer on COP420.

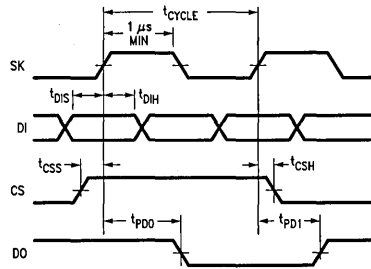


FIGURE 9. NM93C06A Timing

3. As shown in WRITE timing diagram, the start bit on DI must be set by a “0” to “1” transition following a CS enable (“0” to “1”) when executing any instruction. One CS enable transition can only execute one instruction.
4. In the read mode, following an instruction and data train, the DI can be a “don’t care,” while the data is being outputted, i.e., for the next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
5. The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI until another CS LO to HI transition starts a new instruction cycle.
6. After a read cycle, the CS must be brought low for one SK clock cycle before another instruction cycle starts.

INSTRUCTION SET

Commands	Opcode	Comments
READ	1000A3A2A1A0	Read Register 0–15
WRITE	1100A3A2A1A0	Write Register 0–15
ERASE	1010A3A2A1A0	Erase Register 0–15
EWEN	111000 0 0 1	Write/Erase Enable
ENDS	111000 0 1 0	Write/Erase Disable
***WRAL	111000 1 0 0	Write All Registers
ERAL	111000 1 0 1	Erase All Registers

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ— After read command is shifted in DI becomes don’t care and data can be read out on data out, starting with dummy bit zero.

WRITE— Write command shifted in followed by data in (16 bits) then CS pulsed low for 10 ms minimum.

ERASE

ERASE ALL—Command shifted in followed by CS low.

WRITE ALL—Pulsing CS low for 10 ms.

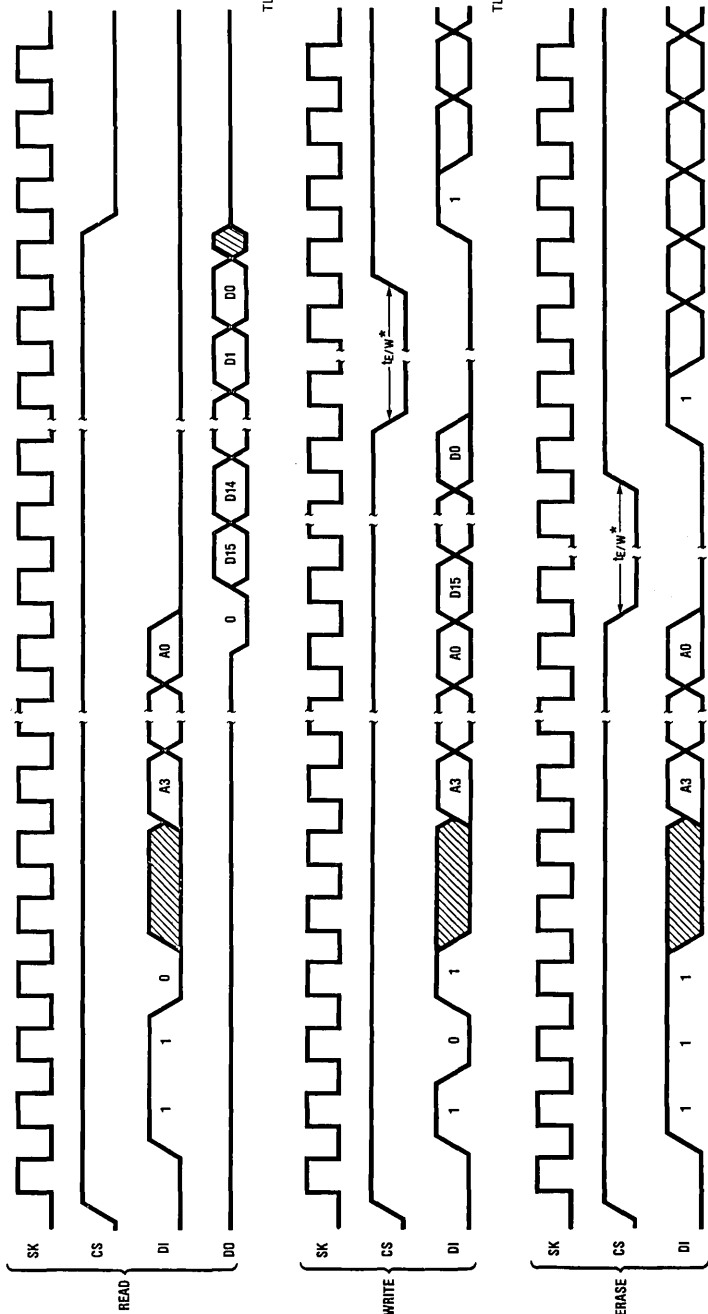
WRITE

ENABLE/DISABLE—Command shifted in.

*** (This instruction is not speeded on Data sheet.)

TIMING WAVEFORMS

Instruction Timing

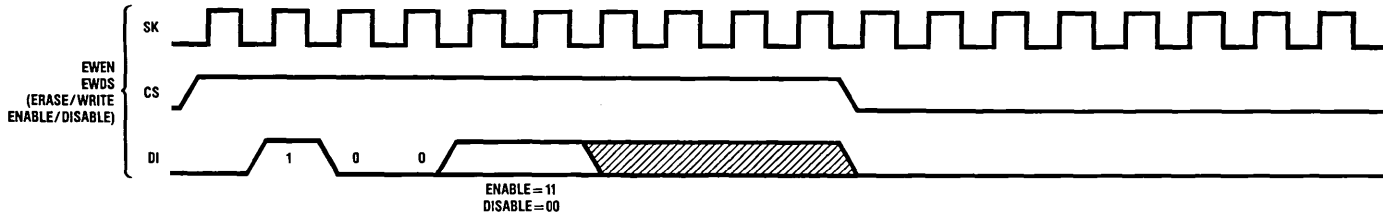


* $t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

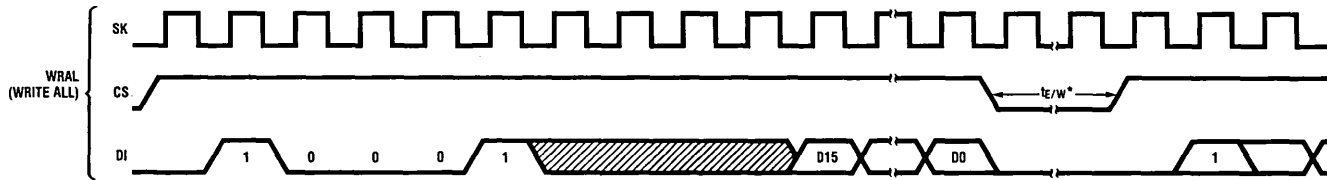
11-968/8796-10

11-968/8796-11

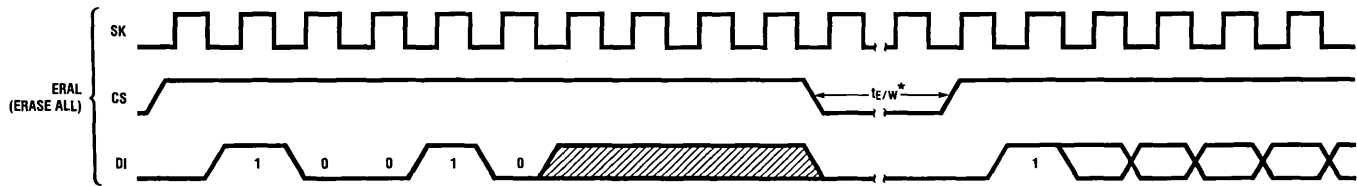
21-968/8796-12



TL/DD/8796-13



TL/DD/8796-14



TL/DD/8796-15

* $t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

I/O ROUTINE TO EVALUATE COP494

```

1          .TITLE      E494, "I/O ROUTINE TO EVALUATE COP494"
2      01A4      .CHIP      420
3      0000      .PAGE      0
4
5          ;THIS IS I/O ROUTINE TO EVALUATE COP494
6          ;
7          ;
8
9          ;RAM VARIABLES DECLARATIONS:
10     000E      COMMAND = 0, 14      ;494 8BITS INST/ADDR WORD
11     001C      RWDATA = 1, 12      ;494 16BITS R/W DATA BUFFER
12
13 000 00      PON:      CLRA          ;POWER-ON INIT
14 001 32          RC          ;RESET SK CLOCK
15 002 4F          XAS
16 003 3F      CLRAM:  LBI      3, 0      ;CLEAR RAM FROM 7, 0 TO 0, 15
17 004 00      CLR:      CLRA          ;
18 005 04          XIS          ;
19 006 C4          JP      CLR          ;CONTI CLEAR REG
20 007 12          XABR          ;(A) TO BR
21 008 5F          AISC      15      ;REG 0 CLEARED?
22 009 600F      DONE:  JMP      C494DR      ;Y, DONE CLEAR RAM, CALL 494 D
23 00B 12          XABR          ;N, DEC BR
24 00C C4          JP      CLR          ;CONTI CLEAR REG TILL DONE
25 00D 44          NOP
26 00E 44          NOP
27
28          ;***      START 494 DRIVER SAMPLE CALLING SEQUENCE      ***
29
30          C494DR:          ;INIT CALLING SEQUENCE
31 00F 3350      OGI      0          ;GO=L TO DESELECT 494
32 011 3368      LEI      8          ;ENABLE SIO AS S.R.
33          ERASE:
34 013 0D          LBI      COMMAND      ;PRELOAD 494 ERASE REG A3-A0
35 014 7C          STII     0C          ;PRELOAD 494 ERASE INST
36 015 70          STII     0          ;SELECT REG A3-A0
37 016 690E      JSR      WI4P4      ;SEND IT
38          WEEN:
39 01B 0D          LBI      COMMAND      ;LOAD 494 WHEN REG A3-A0
40 019 73          STII     3          ;PRELOAD 494 WREN INST
41 01A 70          STII     0          ;SELECT REG A3-A0
42 01B 690E      JSR      WI494      ;SEND IT
43          WRITE:
44 01D 0D          LBI      COMMAND      ;PRELOAD WR REG A3-A0
45 01E 74          STII     4          ;PRELOAD 494 WRITE INST
46 01F 70          STII     0          ;SELECT REG A3-A0
47 020 1B          LBI      RWDATA      ;PRELOAD 494 SAMPLE WRITE DATA
48 021 75          STII     5
49 022 7A          STII     ON
50 023 75          STII     5

```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```

51 024 7A          STII  0A
52 025 6900        JSR   WD494          ;SEND THEM TO 494
53                READ:
54 027 0D          LBI   COMMAND       ;PRELOAD READ REG A3-A0
55 028 78          STII  8             ;PRELOAD 494 READ INST
56 029 70          STII  0             ;SELECT REG A3-A0
57 02A 6908        JSR   RD494         ;READ 494 DATA BACK VIA SI
58 02C 44          NOP
59 02D 44          NOP
60
61 0080            .PAGE  2             ;SUBROUTINE PAGE
62 080 32          SETUP: RC           ;RESET SK BEFORE SELECT 494
63 081 4F          XAS
64 082 3351        OGI   1             ;GO=1 TO SELECT 494
65 084 00          CLRA                ;ENSURE SO=L BEFORE GEN START B
66 085 22          SC
67 086 4F          XAS                 ;TURN ON SK CLOCK
68 087 00          CLRA                ;GENERATE 494 START BIT
69 088 51          AISC  1             ;
70 089 22          SC
71 08A 4F          XAS                 ;SEND IT AS MSB VIA SO
72 08B 0D          LBI   COMMAND       ;FETCH 1ST INST/ADDR WORD
73 08C 05          LD
74 08D 44          NOP
75 08E 4F          XAS                 ;SEND IT (MSB OF INST FIRST)
76 08F 0E          LBI   COMMAND+1     ;FETCH 2ND INST/ADDR NIBBLE
77 090 05          LD
78 091 44          NOP
79 092 4F          XAS                 ;SEND IT
80 093 1B          LBI   RWDATA        ;POINT TO READ/WRITE DATA BUFFER
81 094 48          RET                 ;RET OF SETUP
82
83 095 00          TWEDLY: CLRA         ;VPP WIDTH, TWE>20MS @ 4Us/INST
84 096 5B          TWECONT: AISC  11   ;5 SKT LOOPS?
85 097 99          JP    . + 2         ;N,CONTI
86 098 48          TWEDONE: RET        ;Y,DONE
87 099 41          SKT
88 09A 99          JP    . -1         ;
89 09B 96          JP    TWECONT      ;CONTI TWE TIME
90
91 09C 48          RET;  RET           ;2 CYCLES DELAY
92
93 0100            .PAGE  4             ;
94
95                ;***  START 494 I/O DRIVER SUBROUTINE  ***
96
97 100 80          WD494: JSRP  SETUP    ;ENTRY TO WRITE 494 REG A3-A0
98 101 05          RWLOOP: LD          ;R/W 494 16 DATA BITS
99 102 4F          XAS
100 103 04         XIS

```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```

101 104 C1      JP    RWLOOP
102 105 3350    OGI    0      ;DESELECT 494 AFTER R/W DATA
103 107 D1      JP    FINI    ;
104 108 80     RD494: JSRP  SETUP ;ENTRY TO RD 494 REG A3-A0
105 109 00     CLRA          ;FINISH SEND OUT A3-A0 VIA S0
106 10A 44     NOP          ;
107 10B 44     NOP          ;WAIT 1BIT TIME FOR VALID D15
108 10C 44     NOP
109 10D C1      JP    RWLOOP  ;
110 10E 80     WI494: JSRP  SETUP ;ENTRY TO WRITE INST TO 494
111 10F 00     CLRA          ;ENSURE S0 = L
112 110 4F     XAS          ;
113 111 00     FINI: CLRA          ;ENSURE S0 = L BETWEEN INST
114 112 3350    OGI    0      ;DESELECT 494 BETWEEN INST WRIT
115 114 32     RC          ;
116 115 4F     XAS          ;TURN OFF SK CLOCK
117 116 95     JSRP  TWEDLY ;DELAY TWE >20MS TO PULSE VPP=21
118 117 4B     RET          ;RET OF WD494 OR RD494 OR WI494
119
120           .END

```

SOFTWARE DEBUG OF SERIAL REGISTER FUNCTIONS

In order to understand the method of software debug when dealing with the SIO register, one must first become familiar with the method in which the COPS MOLE™ (Development System) BREAKPOINT and TRACE operations are carried out. Once these operations are explained, the difficulties which could arise when interrogating the status of the SIO register should become apparent.

SERIAL OUT DURING BREAKPOINT

When the MOLE BREAKPOINTS, the COPS user program execution is stopped and execution of a monitor-type program, within the COP device, is started. At no time does the COP part "idle." The monitor program loads the development system with the information contained in the COP registers.

Note also that single-step is simply a BREAKPOINT on every instruction.

If the COP chip is BREAKPOINTed while a serial function is in progress, the contents of the SIO register will be destroyed.

By the time the monitor program dumps the SIO register to the MOLE, the contents of the SIO register will have been written over by clocking in SI. To inspect the SIO register using BREAKPOINT, an XAS must be executed prior to BREAKPOINT; therefore, the SIO register will be saved in the accumulator.

An even more severe consequence is that the monitor program executes an XAS instruction to get the contents of the SIO register to the MOLE. Therefore, the SK latch is dependent on the state of the CARRY prior to the BREAKPOINT. In order to guarantee the integrity of the SIO register, one must carefully choose the position of the BREAKPOINT address.

As can be seen, it is impossible to single-step or BREAKPOINT through a serial operation in the SIO register.

SERIAL OUT DURING TRACE

In the TRACE mode, the user's program execution is never stopped. This mode is a real-time description of the program counter and the external event lines; therefore, the four external event lines can be used as logic analyzers to monitor the state of any input or output on the COPS device. The external event lines must be tied to the I/O which is to be monitored.

The state of these I/O (external event lines) is displayed along with the TRACE information. The safest way to monitor the real-time state of SO is to use the TRACE function in conjunction with the external event lines.

CONCLUSIONS

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and may permit the COPS controller to be packaged in a smaller package.

Automotive Multiplex Wiring

National Semiconductor
Application Note 454
Abdul H. Aleaf



INTRODUCTION

The evolutionary development of vehicle electronic systems has rapidly increased the number of individual wires in the vehicle. The conventional wiring harness will not provide solutions to the problems such as reducing size and weight in addition to meeting cost and reliability objectives. Several approaches have been taken to provide long term solutions. None has succeeded. Miniaturization of cables and wires is one example of a temporary solution.

Multiplexing on the other hand has been regarded as a technique which allows considerable savings to be made in the size and cost of the harness. It can also enhance reliability by reducing the number of electrical connections.

In a multiplex system the control functions will be distributed around the vehicle and complex interconnections between diagnostic terminals, sensors, instruments and switches will not add to the harness complexity. With all its advantages it has not been implemented on a production car yet. The reason has been economical feasibility and lack of suitable semiconductor components for power switching. But, with the rapid technology advances in power FETs and introduction of low cost microcomputers, multiplex wiring can be regarded as a logical successor to conventional wiring systems. Extended development efforts are necessary to introduce a reliable system at reasonable cost.

The Microcontroller Applications Group at National Semiconductor has taken a step towards this goal. A low end multiplex wiring system focusing on asynchronous serial communication in a multi node network has been developed. This paper describes the development of this system on an abstract model which forms the basis for analysis of communication protocol and various node functions.

SYSTEM CONFIGURATION

Figure 1 presents a general view of the system. The system is a centralized single master multiple slave-node scheme. All units are connected together by a balanced twisted pair. The expandable interconnection of different subsystems is achieved with 9600 Baud communication over a standard UART bus. The bus handles the interface between a master controller and the intelligent nodes.

The approach to have a centralized control system offers several advantages as compared against a non-centralized system. It prevents the problem of bus monopolization by a faulty node and is potentially cheaper due to the need for only one complex node (master). The master-slave architecture also prevents bus contention problems.

The master is a COP420L. The COP420L is a 4-bit microcontroller with a software UART that handles asynchronous communication with other processors at speeds up to 9600 Baud.

The use of 4-bit 49¢ microcontrollers (COP413L) at the nodes not only provides intelligence which reduces the required bus bandwidth, it also reduces the incremental cost associated with automotive multiplexing. All standard nodes

are identical. One standard program is used. This uniformity contributes to the system flexibility and expandability. External standard nodes may be added to the system to control additional functions. Node types and addresses are selected via external wire jumpers or switches. The slave nodes consist of four remote units to handle functions such as headlamps, tail lamps, etc. These nodes are the front right, front left, rear right and rear left nodes. Incorporated into the system are also a keyboard node, a EIC node and a display node.

The keyboard node may call for a control action at any time. This node is being continuously monitored by the master controller which receives status and processes the command or information.

Overall system intelligence and flexibility is increased by dedicating a node to NS455 the Terminal Management Processor. This node takes the responsibility to display information on a 4" flat CRT display.

An Electronic Instrument Cluster (EIC) system is a completely independent system. It typically performs all functions associated with the automobile dashboard such as vehicle speed, odometers to accumulate mileage, gauges to display engine temperature, fuel level and so on. It also indicates error conditions such as high engine temperatures, low fuel level etc. The multiplex wiring system uses a standard slave node as a bridge between the two independent systems. The slave node monitors error conditions from the EIC system and passes them to the master node upon request. It becomes relatively simple to allow the master to access all activity in the EIC system via additional commands to the slave node serving the EIC system:

THE COMMUNICATION PROTOCOL

The master unit addresses the remote units sequentially and receives a status reply from each individual node. Data communication is via the standard UART format. It has a start bit, eight data bits, an even parity bit and one stop bit. Information to be transmitted from the master to a slave node is organized as a frame. Each frame contains the address of destination and command or data. The information in a frame is transmitted as byte format. Address/data differentiation is done by means of a flag. The byte is an address byte if the MSB is set ("1"), otherwise it is a data byte.

Two different types of addressing schemes have been incorporated into the communication protocol; node addressing and class addressing. A class of nodes is formed by grouping together slave nodes with common functions. Commands may be executed either by specific individual nodes or by slave classes. All nodes of the same class execute the command simultaneously. The system implementation at National involved four classes with seven slave nodes per class. So, the total number of nodes possible in this system is 28.

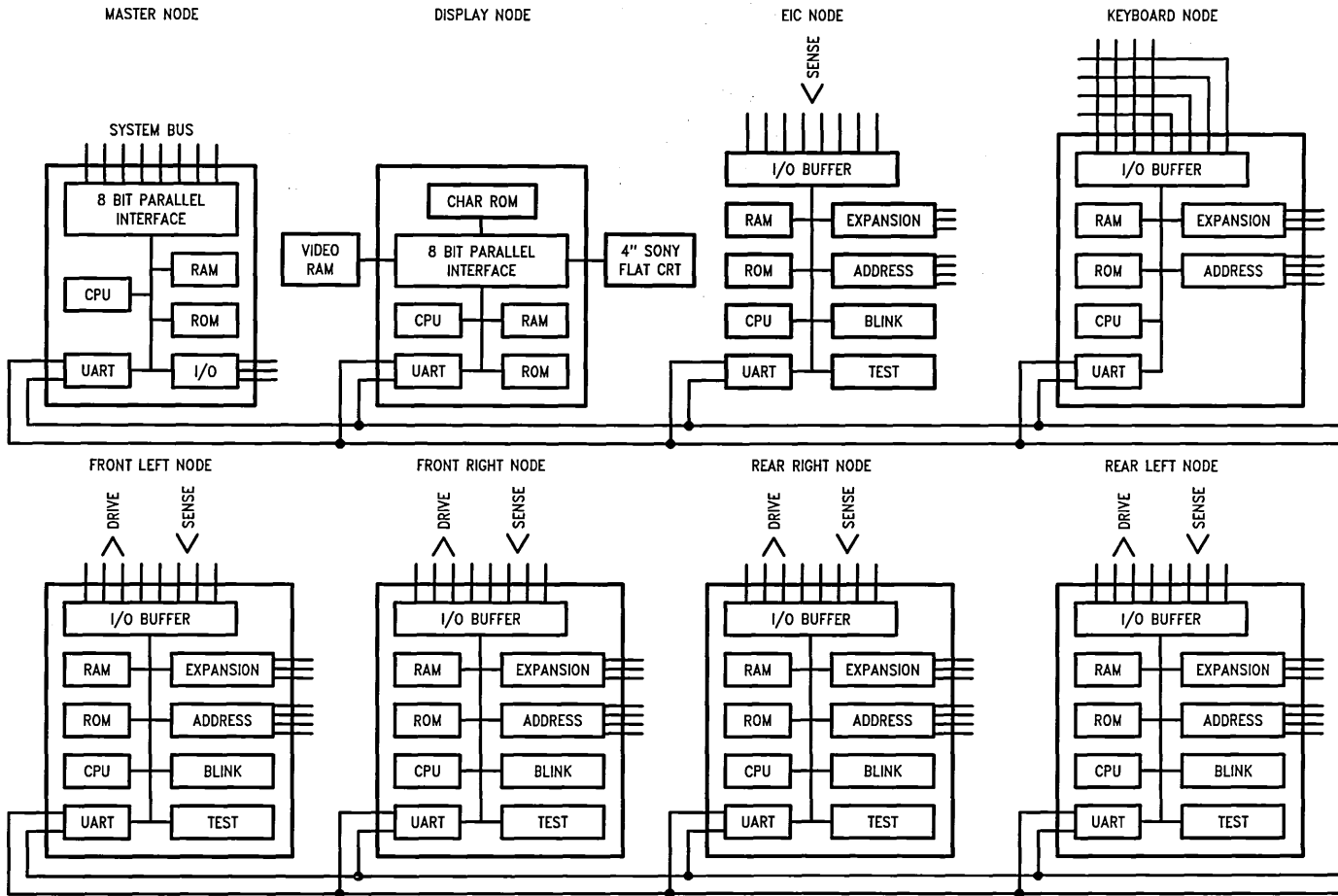


FIGURE 1. Block Diagram

TL/DD/8799-1

The partitioning between the class address and node address reduces the density of bus traffic significantly by eliminating repetitive command transmission to individual node class. Lower bus traffic implies that lower transmission bit rate can be used, allowing additional noise immunity. Another advantage of the class addressing is the provision of synchronization for control signals such as HAZARD, LEFT/RIGHT turns.

Error correction is incorporated into the communication protocol. The UART error flags such as PARITY and FRAMING ERRORS protect the system at the physical layer. At the system level, the nodes simply avoid sending an acknowledgement to the master when an error is detected. The master times out and sends the command again.

THE MASTER NODE

The master controller is the heart of the system. Its responsibility is to generate the controlling commands and synchronize the system. It transmits to the remote units and listens to them to get the vehicle status and acts accordingly. Circuit complexity is reduced by implementing extensive software programming in the master controller. This means that the burden is essentially on the master and must be engineered to very high standards of reliability. The device used in the implementation as the master is the COP1430. It is a cost effective 4-bit single chip microcontroller. It features on chip UART which handles asynchronous communications at speeds up to 9600 Baud.

THE SLAVE NODES

The standard slave nodes are based upon the COP413L. The COP413L is a low cost 4-bit microcontroller which may be customized in production. A system such as multiplex wiring requires power consumption to be absolutely minimal. Another basic requirement is that the system should be cost effective. These two facts directed us to use the COP413L at the standard slave node. The COP413L is a low cost (49¢) low power microcontroller from NSC drawing less

than 7 mA at 4.5V to 5.5V. The device contains an 8-bit bidirectional I/O port and a serial expansion port. The CMOS version of COP413L will also be available.

THE DISPLAY NODE

This node can serve as a condition monitoring unit for the vehicle. A considerable quantity of diagnostic information collected from transducers, switches, sensors and various loads are fed to this unit to be displayed on a CRT display. The node is based on a Terminal Management Processor the NS455. The NS455 is a CRT controller on chip. The messages are updated over the serial I/O line by the master controller. The communication format is:

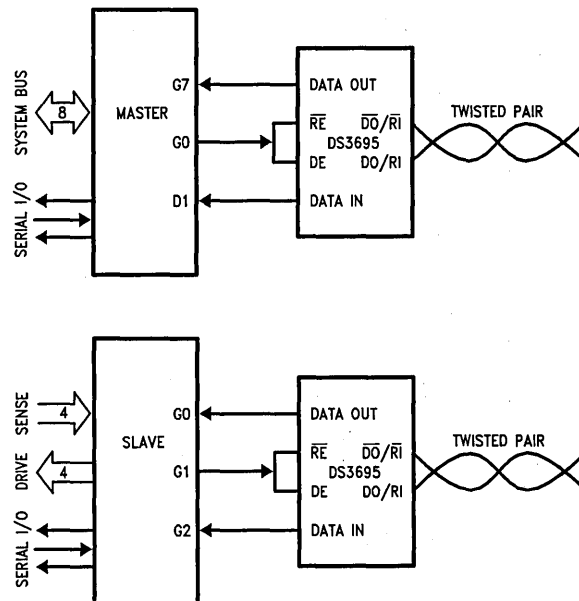
- a) The node receives the address.
- b) If address matches the local node address, send the copy command
- c) Receive new address and execute.

OUTPUT STAGES

The power FETs used for local switching throughout the system are IRF541 (4). These N-channel FETs provide much better drive circuit specification as compared to bipolar output stages. They also feature all of the well established advantages of MOSFET such as voltage control, very fast switching, and very low on state resistance. Another advantage is the lower cost as compared to comparably rated p-channel devices.

TRANSMISSION MEDIUM

A balanced twisted pair is used for bus medium which provides high noise immunity. The transceiver selected for the bus is DS3695 (Figure 2). This device is a high speed differential TRI-STATE® Bus/line transceiver designed to meet EIA standard for multipoint bus transmission. Bus contention or fault situations that cause excessive power dissipation within the device are handled by a standard thermal shutdown circuit, which forces the driver outputs into the high impedance state.



TL/DD/8799-2

TL/DD/8799-3

FIGURE 2. Bus Interface

CONCLUSIONS

Multiplex wiring system potentially seems to be a good replacement for conventional wiring system. Reduced complexity, increased flexibility and diagnostic capability could be achieved by incorporating microcontroller devices at nodes within the wiring system. The 4-bit microcontrollers selected are available in a price range, as low as 49¢, that will allow multiplex wiring to compare favorably on a cost-performance basis with the conventional harness.

REFERENCES

1. Michael W. Lowndes and Paul E. V. Phillips, "The Motorcar Multiplex Systems", IEE Conference on Automotive Electronics, 229, England, Nov. 1983.
2. R. F. Robins/W. J. Brittain/M. R. Lunt, "A Car Multiplex Wiring System with Self Coding Control Modules", IEE Conference on Automotive Electronics, 229, Ford Motor Company, UK, Nov. 1983.
3. Booth, J. A., 1983 "Vehicle Interconnection Systems for the Future", IEE Conference on Automotive Electronics, London, Nov. 1983.
4. International Rectifier, HEXFET Databook, 1985.

Dual Tone Multiple Frequency (DTMF)

National Semiconductor
Application Note 521
Verne H. Wilson



The DTMF (Dual Tone Multiple Frequency) application is associated with digital telephony, and provides two selected output frequencies (one high band, one low band) for a duration of 100 ms. A benchmark subroutine has been written for the COP820C/840C microcontrollers, and is outlined in detail in this application note. This DTMF subroutine takes 110 bytes of COP820C/840C code, consisting of 78 bytes of program code and 32 bytes of ROM table. The timings in this DTMF subroutine are based on a 20 MHz COP820C/840C clock, giving an instruction cycle time of 1 μ s.

The matrix for selecting the high and low band frequencies associated with each key is shown in *Figure 1*. Each key is uniquely referenced by selecting one of the four low band frequencies associated with the matrix rows, coupled with selecting one of the four high band frequencies associated with the matrix columns. The low band frequencies are 697, 770, 852, and 941 Hz, while the high band frequencies are 1209, 1336, 1477, and 1633 Hz. The DTMF subroutine assumes that the key decoding is supplied as a low order hex digit in the accumulator. The COP820C/840C DTMF subroutine will then generate the selected high band and low band frequencies on port G output pins G3 and G2 respectively for a duration of 100 ms.

The COP820C/840C each contain only one timer. The problem is that three different times must be generated to satisfy the DTMF application. These three times are the periods of the two selected frequencies and the 100 ms duration period. Obviously the single timer can be used to generate any one (or possibly two) of the required times, with the program having to generate the other two (or one) times.

The solution to the DTMF problem lies in dividing the 100 ms time duration by the half periods (rounded to the nearest micro second) for each of the eight frequencies, and then examining the respective high band and low band quotients and remainders. The results of these divisions are detailed in Table I. The low band frequency quotients range from 139 to 188, while the high band quotients range from 241 to 326. The observation that only the low band quotients will each fit in a single byte dictates that the high band frequency be produced by the 16 bit (2 byte) COP820C/840C timer running in PWM (Pulse Width Modulation) Mode.

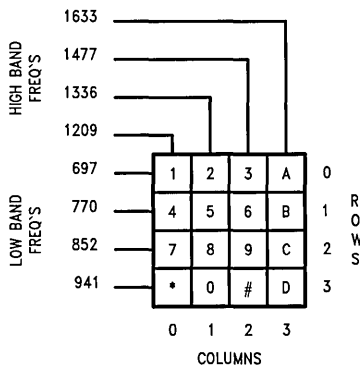


FIGURE 1. DTMF Keyboard Matrix

TL/DD/9662-1

The solution then is to use the program to produce the selected low band frequency as well as keep track of the 100 ms duration. This is achieved by using three programmed register counters R0, R2, and R3, with a backup register R1 to reload the counter R0. These three counters represent the half period, the 100 ms quotient, and the 100 ms remainder associated with each of the four low band frequencies.

The theory of operation in producing the selected low band frequency starts with loading the three counters with values obtained from a ROM table. The half period for the selected frequency is counted out, after which the G2 output bit is toggled. During this half period countout, the quotient counter is decremented. This procedure is repeated until the quotient counter counts out, after which the program branches to the remainder loop. During the remainder loop, the remainder counter counts out to terminate the 100 ms. Following the remainder countout, the G2 and G3 bits are both reset, after which the DTMF subroutine is exited. Great care must be taken in time balancing the half period loop for the selected low band frequency. Furthermore, the toggling of the G2 output bit (achieved with either a set or reset bit instruction) must also be exactly time balanced to maintain the half period time integrity. Local stall loops (consisting of a DRSZ instruction followed by a JP jump back to the DRSZ for a two byte, six instruction cycle loop) are embedded in both the half period and remainder loops. Consequently, the ROM table parameters for the half period and remainder counters are approximately only one sixth of what otherwise might be expected. The program for the half period loop, along with the detailed time balancing of the loop for each of the low band frequencies, is shown in *Figure 2*.

The DTMF subroutine makes use of two 16 byte ROM tables. The first ROM table contains the translation table for the input hex digit into the core vector. The encoding of the hex digit along with the hex digit ROM translation table is shown in Table II. The row and column bits (RR, CC) representing the low band and high band frequencies respectively of the keyboard matrix shown in *Figure 1*, are encoded in

TABLE I. Frequency Half Periods, Quotients, and Remainders

	Freq. Hz	Half Period 0.5P	Half Period in μ s	100 ms/0.5P	
				Quotient	Remainder
Low Band Freq.'s	697	717.36	717	139	337
	770	649.35	649	154	54
	852	586.85	587	170	210
	941	531.35	531	188	172
High Band Freq.'s	1209	413.56	414 (256 + 158)	241	226
	1336	374.25	374 (256 + 118)	267	142
	1477	338.52	339 (256 + 83)	294	334
	1633	306.18	306 (256 + 50)	326	244

the two upper and two lower bits of the hex digit respectively. Consequently, the format for the hex digit bits is RRCC, so that the input byte in the accumulator will consist of 0000RRCC. The program changes this value into 1101RRCC before using it in setting up the address for the hex digit ROM translation table.

The core vectors from the hex digit ROM translation table consist of a format of XX00TT00, where the two T (Timer) bits select one of four high band frequencies, while the two X bits select one of four low band frequencies. The core vector is transformed into four different inputs for the second ROM table. This transformation of the core vector is shown in Table III. The core vector transformation produces a timer vector 1100TT00 (T), and three programmed coun-

ter vectors for R1, R2, and R3. The formats for the three counter vectors are 1100XX11 (F), 1100XX10 (Q), and 1100XX01 (R) for R1, R2, and R3 respectively. These four vectors produced from the core vector are then used as inputs to the second ROM table. One of these four vectors (the T vector) is a function of the T bits from the core vector, while the other three vectors (F, Q, R) are a function of the X bits. This correlates to only one parameter being needed for the timer (representing the selected high band frequency), while three parameters are needed for the three counters (half period, 100 ms quotient, 100 ms remainder) associated with the low band frequency and 100 ms duration. The frequency parameter ROM translation table, accessed by the T, F, Q, and R vectors, is shown in Table IV.

Program	Bytes/Cycle	Conditional Cycles		Cycles	Total Cycles
LD B,#PORTGD	2/3				
LD X,#R1	2/3				
LUP1: LD A,[X-]	1/3			3	
IFBIT 2,[B]	1/1			1	
JP BYP1	1/3	3	1		
X A,[X+]	1/3		3		
SBIT 2,[B]	1/1		1		
JP BYP2	1/3		3		
BYP1: NOP	1/1	1			
RBIT 2,[B]	1/1	1			
X A,[X+]	1/3	3			
BYP2: DRSZ R2	1/3 DECREMENT			3	
JP LUP2	1/3 Q COUNT			3	
JP FINI	1/3				
LUP2: DRSZ R0	1/3 DECREMENT		3	3	
JP LUP2	1/3 F COUNT		3	1	
NOP	1/1			1	
LD A,[X]	1/3			3	
IFEQ A,#104	2/2			2	
JP LUP1	1/3		1	3	31
NOP	1/1		1		
IFEQ A,#93	2/2		2		
BACK: JP LUP1	1/3	1	3		35
JP BACK	1/3	3			
		3			39

Table IV	×	Stall	+	Total	=	Half
Frequency		Loop		Cycles		Period
((114 - 1)		x 6)		+ 39		= 717
((104 - 1)		x 6)		+ 31		= 649
((93 - 1)		x 6)		+ 35		= 587
((83 - 1)		x 6)		+ 39		= 531

FIGURE 2. Time Balancing for Half Period Loop

TABLE II. Hex Digit ROM Translation Table

	0	1	2	3
ROW	697 Hz	770 Hz	852 Hz	941 Hz
COLUMN	1209 Hz	1336 Hz	1477 Hz	1633 Hz

ADDRESS	DATA (HEX)	KEYBOARD	
*			* HEX DIGIT IS RRCC,
0xD0	000	1	WHERE R = ROW #
0xD1	004	2	AND C = COLUMN #
0xD2	008	3	- - - EXAMPLE: KEY 3 IS ROW #0,
0xD3	00C	A	COLUMN #2, SO HEX DIGIT
0xD4	040	4	IS 0010 = 2
0xD5	044	5	RRCC
0xD6	048	6	
0xD7	04C	B	
0xD8	080	7	
0xD9	084	8	
0xDA	088	9	
0xDB	08C	C	
0xDC	0C0	*	
0xDD	0C4	0	
0xDE	0C8	#	
0xDF	0CC	D	

TABLE III. Core Vector Translation

CORE VECTOR	-	XX00TT00	- - - - -	
				*
				**

TIMER VECTOR	TIMER	T		1100TT00
HALF PERIOD VECTOR	R1	F		1100XX11
QUOTIENT VECTOR	R2	Q		1100XX10
REMAINDER VECTOR	R3	R		1100XX01

TABLE IV. Frequency Parameter ROM Translation Table

T - TIMER F - FREQUENCY Q - QUOTIENT R - REMAINDER

ADDRESS	DATA (DEC)	VECTOR
0xC0	158	T
0xC1	53	R
0xC2	140	Q
0xC3	114	F
0xC4	118	T
0xC5	6	R
0xC6	155	Q
0xC7	104	F
0xC8	83	T
0xC9	32	R
0xCA	171	Q
0xCB	93	F
0xCC	50	T
0xCD	25	R
0xCE	189	Q
0xCF	83	F

In summary, the input hex digit selects one of 16 core vectors from the first ROM table. This core vector is then transformed into four other vectors (T, F, Q, R), which in turn are used to select four parameters from the second ROM table. These four parameters are used to load the timer, and the respective half period, quotient, and remainder counters. The first ROM table (representing the hex digit matrix table) is arbitrarily placed starting at ROM location 01D0, and has a reference setup with the ADD A,#0D0 instruction. The second ROM table (representing the frequency parameter table) must be placed starting at ROM location 01C0 (or 0xC0) in order to minimize program size, and has reference setups with the OR A,#0C3 instruction for the F vector and with the OR A,#0C0 instruction for the T vector.

The three parameters associated with the two X bits of the core vector require a multi-level table lookup capability with the LAID instruction. This is achieved with the following section of code in the DTMF subroutine:

```

LD      B, #R1
LD      X, #R4
X       A, [X]
LUP:   LD      A, [X]
        LAID
X       A, [B+]
DRSZ   R4
IFBNE  #4
JP     LUP

```

This program code loads the F frequency vector into R4, and then decrements the vector each time around the loop. This successive loop decrementation of the R4 vector changes the F vector into the Q vector, and then changes the Q vector into the R vector. This R4 vector is used to access the ROM table with the LAID instruction. The X pointer references the R4 vector, while the B pointer is incremented each time around the loop after it has been used to store away the three selected ROM table parameters (one per loop). These three parameters are stored in sequential RAM locations R1, R2, and R3. The IFBNE test instruction is used to skip out of the loop once the three selected ROM table parameters have been accessed and stored away.

The timer is initialized to a count of 15 so that the first timer underflow and toggling of the G3 output bit (with timer PWM mode and G3 toggle output selected) will occur at the same time as the first toggling of the G2 output bit. The half period counts for the high band frequencies range from 306 to 414, so these values minus 256 are stored in the timer section of the second ROM table. The selected value from this frequency ROM table is then stored in the lower half of the timer autoreload register, while a 1 is stored in the upper half. The timer is selected for PWM output mode and started with the instruction LD [B],#0B0 where the B pointer is selecting the CNTRL register at memory location 0EE.

The DTMF subroutine for the COP820C/840C uses 110 bytes of code, consisting of 78 bytes of program code and 32 bytes of ROM table. A program routine to sequentially call the DTMF subroutine for each of the 16 hex digit inputs is supplied with the listing for the DTMF subroutine.

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV: B, 20 JAN 87
 DTMF

PAGE: 1

```

1          ;DTMF PROGRAM FOR COP820C/840C          VERNE H. WILSON
2          ;                                         5/1/89
3          ;DTMF - DUAL TONE MULTIPLE FREQUENCY
4          ;
5          ;PROGRAM NAME: DTMF.MAC
6          ;
7          .TITLE DTMF
8          .CHIP 840
9          ;***** THE DTMF SUBROUTINE CONTAINS 110 BYTES *****
10         ; ***** THE DTMF SUBROUTINE TIMES OUT IN 100MSEC *****
11         ; ** FROM THE FIRST TOGGLE OF THE G2/G3 OUTPUTS **
12         ; ** BASED ON A 20 MHZ COP820C/840C CLOCK **
13         ;
14         ;G PORT IS USED FOR THE TWO OUTPUTS
15         ; - HIGH BAND (HB) FREQUENCY OUTPUT ON G3
16         ; - LOW BAND (LB) FREQUENCY OUTPUT ON G2
17         ;
18         ;TIMER COUNTS OUT
19         ; - HB FREQUENCIES
20         ;
21         ;PROGRAM COUNTS OUT
22         ; - LB FREQUENCIES
23         ; - 100 MSEC DIVIDED BY LB HALF PERIOD QUOTIENT
24         ; - 100 MSEC DIVIDED BY LB HALF PERIOD REMAINDER
25         ;
26         ;FORMAT FOR THE 16 HEX DIGIT MATRIX VECTOR IS 1101RRCC,
27         ; WHERE - RR IS ROW SELECT (LB FREQUENCIES)
28         ; - CC IS COLUMN SELECT (HB FREQUENCIES)
29         ;
30         ;FORMAT FOR THE 16 CORE VECTORS FROM THE MATRIX SELECT
31         ; TABLE IS XX00T100, WHERE - TT IS HB SELECT
32         ; - XX IS LB SELECT
33         ;
34         ;FREQUENCY VECTORS (HB & LB) FOR FREQ PARAMETER TABLE
35         ; MADE FROM CORE VECTORS
36         ;
37         ;HB FREQUENCY VECTORS(4) END WITH 00 FOR TIMER COUNTS,
38         ; WHERE VECTOR FORMAT IS 1100T100
39         ;
40         ;LB FREQUENCY VECTORS(12) END WITH:
41         ; 11 FOR HALF PERIOD LOOP COUNTS,
42         ; WHERE VECTOR FORMAT IS 1100XX11
43         ; 10 FOR 100 MSEC DIVIDED BY HALF PERIOD QUOTIENTS,
44         ; WHERE VECTOR FORMAT IS 1100XX10
45         ; 01 FOR 100 MSEC DIVIDED BY HALF PERIOD REMAINDERS,
46         ; WHERE VECTOR FORMAT IS 1100XX01
47         ;
48         ;HEX DIGIT MATRIX TABLE AT HEX 01D* (OPTIONAL LOCATION,
49         ; DEPENDING ON 'ADD A,#0D0' INST. IMMEDIATE VALUE)
50         ;
51         ;FREQ PARAMETER TABLE AT HEX 01C* (REQUIRED LOCATION)

```

TL/DD/9662-2

```

52          .FORM
53
54          ;MAGIC:          CORE VECTOR
55          ;                  XX00TT00
56          ;
57          ;      TIMER      T      TT00
58          ;      R1         F      XX11
59          ;      R2         Q      XX10
60          ;      R3         R      XX01
61          ;
62          ;DECLARATIONS:
63          00D0      PORTLD = 0D0      ; PORTL DATA REG
64          00D1      PORTLC = 0D1      ; PORTL CONFIG REG
65          00D4      PORTGD = 0D4      ; PORTG DATA REG
66          00D5      PORTGC = 0D5      ; PORTG CONFIG REG
67          00DC      PORTD = 0DC      ; PORTD REG
68          00EA      TIMERLO = 0EA     ; TIMER LOW COUNTER
69          00EE      CNTRL = 0EE      ; CONTROL REG
70          00EF      PSW = 0EF        ; PROC STATUS WORD
71          00F0      R0 = 0F0        ; LB FREQ LOOP COUNTER
72          00F1      R1 = 0F1        ; LB FREQ LOOP COUNT
73          00F2      R2 = 0F2        ; LB FREQ Q COUNT
74          00F3      R3 = 0F3        ; LB FREQ R COUNT
75          00F4      R4 = 0F4        ; LB FREQ TABLE VECTOR
76
77          ;START:      LD      SP, #02F      ; HEX DIGIT MATRIX
78          0002      BCD1FF      LD      PORTLC, #0FF      ; 1 2 3 A
79          0005      BCD080      LD      PORTLD, #080      ; 4 5 6 B
80          0008      DEDC        LD      B, #PORTD      ; 7 8 9 C
81          000A      9E00        LD      [B], #0      ; * 0 # D
82          000C      AE          LOOP:   LD      A, [B]      ; DTMF TEST LOOP
83          000D      3160        JSR     DTMF      ; HEX MATRIX DIGIT
84          000F      DEDC        LD      B, #PORTD      ; TO SUBROUTINE IS
85          0011      AE          LD      A, [B]      ; OUTPUT TO PORTD
86          0012      9405        ADD     A, #5      ; DO WILL TOGGLE
87          0014      A6          X        A, [B]      ; FOR EACH CALL OF
88          0015      6C          RBIT    4, [B]      ; DTMF SUBROUTINE
89          0016      9DD0        LD      A, PORTLD     ; PORTL OUTPUTS
90          0018      A1          SC      ; PROVIDE SYNC
91          0019      B0          RRC     A      ; OUTPUT ORDER IS
92          001A      9CD0        X        A, PORTLD     ; 1,5,9,D,4,8,#,A,
93          001C      EF          JP      LOOP      ; 7,0,3,B,* ,2,6,C
94          ;
95          ;
96

```

TL/DD/9662-3

```

97      0160      . = 0160
98
99      0160 DED5   ; DTMF: LD B, #PORTGC
100     0162 9B3F   LD [B-], #03F
101     0164 6B     RBIT 3, [B] ; OPTIONAL
102     0165 6A     RBIT 2, [B] ; OPTIONAL
103
104     0166 94D0   ; ADD A, #0D0
105     0168 A4     LAID ; DIGIT MATRIX TABLE
106
107     0169 5F     ; LD B, #0
108     016A A6     X A, [B]
109     016B AE     LD A, [B]
110     017B 65     SWAP A
111     016C 97C3   OR A, #0C3
112     016E DEF1   LD B, #R1
113     0170 DCF4   LD X, #R4
114     0172 B6     X A, [X]
115     0173 BE     LUP: LD A, [X]
116     0174 A4     LAID ; LB FREQ TABLES
117     0175 A2     X A, [B+] ; (3 PARAMETERS)
118     0176 C4     DRSZ R4
119     0177 44     IFBNE #4
120     0178 FA     JP LUP
121
122     0179 5F     ; LD B, #0
123     017A AE     LD A, [B]
124     017C 97C0   OR A, #0C0
125     017E A4     LAID ; HB FREQ TABLE
126     017F DEEA   LD B, #TIMERL0 ; (1 PARAMETER)
127     0181 9A0F   LD [B+], #15
128     0183 9A00   LD [B+], #0
129     0185 A2     X A, [B+]
130     0186 9A01   LD [B+], #1
131     0188 9EBO   LD [B], #0B0 ; START TIMER PWM
132
133     018A DED4   ; LD B, #PORTGD
134     018C DCF1   LD X, #R1
135
136     018E BB     LUP1: LD A, [X-]
137     018F 72     IFBIT 2, [B] ; TEST LB OUTPUT
138     0190 03     JP BYP1
139     0191 B2     X A, [X+]
140     0192 7A     SBIT 2, [B] ; SET LB OUTPUT
141     0193 03     JP BYP2
142     0194 B8     BYP1: NOP
143     0195 6A     RBIT 2, [B] ; RESET LB OUTPUT
144     0196 B2     X A, [X+]
145     0197 C2     BYP2: DRSZ R2 ; DECR. QUOT. COUNT
146     0198 01     JP LUP2
147     0199 0C     JP FINI ; Q COUNT FINISHED
148
149     019A C0     ; LUP2: DRSZ R0 ; DECR. F COUNT
150     019B FE     JP LUP2 ; LB (HALF PERIOD)
151
152     019C B8     ; NOP ; *****
153     019D BE     LD A, [X] ; BALANCE
154     019E 9268   IFEQ A, #104 ; LB FREQUENCY
155     01A0 ED     JP LUP1 ; HALF PERIOD
156
157     01A1 B8     ; NOP ; RESIDUE
158     01A2 925D   IFEQ A, #93 ; DELAY FOR
159     01A4 E9     BACK: JP LUP1 ; EACH OF 4
160     01A5 FE     JP BACK ; LB FREQ'S
161
162     01A6 C3     ; FINI: DRSZ R3 ; DECR. REM. COUNT
163     01A7 FE     JP FINI ; R CNT NOT FINISHED
164
165     01A8 BDEE6C ; RBIT 4, CNTRL ; STOP TIMER
166     01AB 6B     RBIT 3, [B] ; CLR HB OUTPUT
167     01AC 6A     RBIT 2, [B] ; CLR LB OUTPUT
168
169     01AD 8E     ; RET
170

```

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV:B, 20 JAN 87
 DTMF

PAGE: 4

```

171                                     .FORM
172                                     ;
173                                     ;FREQUENCY AND 100MSEC PARAMETER TABLE
174                                     ;
175                                     ;
176 01C0 9E                               .BYTE      158           ; T
177 01C1 35                               .BYTE      53           ; R
178 01C2 8C                               .BYTE     140           ; Q
179 01C3 72                               .BYTE     114           ; F
180 01C4 76                               .BYTE     118           ; T
181 01C5 06                               .BYTE       6           ; R
182 01C6 9B                               .BYTE     155           ; Q
183 01C7 68                               .BYTE     104           ; F
184 01C8 53                               .BYTE      83           ; T
185 01C9 20                               .BYTE      32           ; R
186 01CA AB                               .BYTE     171           ; Q
187 01CB 5D                               .BYTE      93           ; F
188 01CC 32                               .BYTE      50           ; T
189 01CD 19                               .BYTE      25           ; R
190 01CE BD                               .BYTE     189           ; Q
191 01CF 53                               .BYTE      83           ; F

```

```

192                                     ;
193                                     ;DIGIT MATRIX TABLE
194                                     ;
195                                     ;

```

```

196 01D0 00                               .BYTE     000           ; 1
197 01D1 04                               .BYTE     004           ; 2
198 01D2 08                               .BYTE     008           ; 3
199 01D3 0C                               .BYTE     00C           ; A
200 01D4 40                               .BYTE     040           ; 4
201 01D5 44                               .BYTE     044           ; 5
202 01D6 48                               .BYTE     048           ; 6
203 01D7 4C                               .BYTE     04C           ; B
204 01D8 80                               .BYTE     080           ; 7
205 01D9 84                               .BYTE     084           ; 8
206 01DA 88                               .BYTE     088           ; 9
207 01DB 8C                               .BYTE     08C           ; C
208 01DC C0                               .BYTE     0C0           ; *
209 01DD C4                               .BYTE     0C4           ; 0
210 01DE C8                               .BYTE     0C8           ; #
211 01DF CC                               .BYTE     0CC           ; D
212                                     ;
213                                     .END

```

	ROW	COL
	0	0
	0	1
	0	2
	0	3
	1	0
	1	1
	1	2
	1	3
	2	0
	2	1
	2	2
	2	3
	3	0
	3	1
	3	2
	3	3

TL/DD/9662-5

SYMBOL TABLE

B	00FE	BACK	01A4	BYP1	0194	BYP2	0197
CNTRL	00EE	DTMF	0160	FINI	01A6	LOOP	000C
LUP	0174	LUP1	018E	LUP2	019A	PORTD	00DC
PORTGC	00D5	PORTGD	00D4	PORTLC	00D1	PORTLD	00D0
PSW	00EF *	R0	00F0	R1	00F1	R2	00F2
R3	00F3	R4	00F4	SP	00FD	START	0000 *
TIMERL	00EA	X	00FC				

MACRO TABLE

NO WARNING LINES

NO ERROR LINES

139 ROM BYTES USED

SOURCE CHECKSUM = 99A7
 OBJECT CHECKSUM = 03E1

INPUT FILE C:DTMF.MAC
 LISTING FILE C:DTMF.PRN
 OBJECT FILE C:DTMF.LM

TL/DD/9662-6

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
 Voice (408) 721-5582

For Additional Information, Please Contact Factory

MICROWIRE/PLUS™ Serial Interface for COP800 Family

National Semiconductor
Application Note 579
Ramesh Sivakolundu
Sunder Velamuri



AN-579

INTRODUCTION

National Semiconductor's COP800 family of full-feature, cost-effective microcontrollers use a new 8-bit single chip core architecture fabricated with M²CMOS process technology. These high performance microcontrollers provide efficient system solutions with a versatile instruction set and high functionality.

The COP800 family of microcontrollers feature the MICROWIRE/PLUS mode of serial communication. MICROWIRE/PLUS is an enhancement of the MICROWIRE™ synchronous serial communications scheme, originally implemented on the COP400 family of microcontrollers. The MICROWIRE/PLUS interface on the COP800 family of microcontrollers enables easy I/O expansion and interfacing to several COPS peripheral devices (A/D converters, EEPROMs, Display drivers etc.), and interfacing with other microcontrollers which support MICROWIRE/PLUS or SPI* modes of serial interface.

MICROWIRE/PLUS DEFINITION

MICROWIRE/PLUS is a versatile three wire, SI (serial input), SO (serial output), and SK (serial clock), bidirectional serial synchronous communication scheme where the COP800 is either the Master providing the Shift Clock (SK) or a slave accepting an external Shift Clock (SK). The COP800 MICROWIRE/PLUS system block diagram is shown in Figure 1. The MICROWIRE/PLUS serial interface utilizes an 8-bit memory mapped MICROWIRE/PLUS serial shift register, SIOR, clocked by the SK signal. As the name suggests, the SIOR register serves as the shift register for serial transfers. SI, the serial input line to the COP800 microcontroller, is the shift register input. SO, the shift register output, is the serial output to external devices. SK is the serial synchronous clock. Data is clocked into and out of the

peripheral devices with the SK clock. The SO, SK and SI are mapped as alternate functions on pins 4, 5, and 6 respectively of the 8-bit bidirectional G Port.

MICROWIRE/PLUS OPERATION

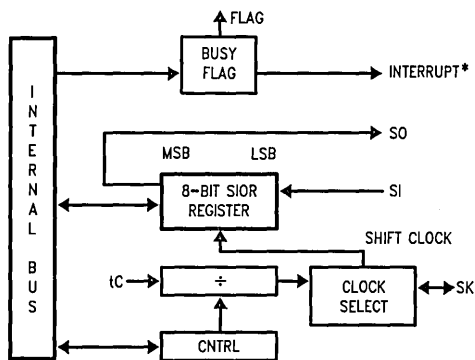
In MICROWIRE/PLUS serial interface, the input data on the SI pin is shifted high order first into the Least Significant Bit (LSB) of the 8-bit SIOR shift register. The output data is shifted out high order first from the Most Significant Bit (MSB) of the shift register onto the SO pin. The SIOR register is clocked on the falling edge of the SK clock signal. The input data on the SI pin is shifted into the LSB of the SIOR register on the rising edge of the SK clock. The MSB of the SIOR register is shifted out to the SO pin on the falling edge of the SK clock signal. The SK clock signal is generated internally by the COP800 for the master mode of MICROWIRE/PLUS operation. In the slave mode, the SK clock is generated by an external device (which acts as the master) and is input to the COP800.

The MSEL (MICROWIRE Select) flag in the CNTRL register is used to enable MICROWIRE/PLUS operation. Setting the MSEL flag enables the gating of the MICROWIRE/PLUS interface signals through the G port. Pins G4, G5, and G6 of the G port are used for the signals SO, SK and SI, respectively. It should be noted that the G port configuration register must be set up appropriately for MICROWIRE/PLUS operation. Table I illustrates the G-port configurations. In the master mode of MICROWIRE/PLUS operation, G4 and G5 need to be selected as outputs for SO and SK signals. Alternatively, in the slave mode of operation, G5 needs to be configured as an input for the external SK. The SI signal is a dedicated input on G6 and therefore no further setup is required.

TABLE I. G Port Configurations

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE Master
0	1	TRI-STATE	Int. SK	MICROWIRE Master
1	0	SO	Ext. SK	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE Slave

The SL1 and SL0 (S1 and S0 in COP820C and COP840C) bits of the CNTRL register are used to select the clock division factor (2, 4, or 8) for SK clock generation in MICROWIRE/PLUS master mode operation. A clock select table for these bits of the CNTRL register along with the CNTRL register is shown in Table II. The counter associated with



TL/DD/10252-1

*only in COP88XX series

FIGURE 1. MICROWIRE/PLUS Block Diagram

3

the master mode clock division factor is cleared when the MICROWIRE/PLUS BUSY flag is low. The clock division factor is relative to the instruction cycle frequency. For example, if the COP800 is operating with an internal clock of 1 MHz, the SK clock rate would be 500 kHz, 250 kHz, or 125 kHz for SL1 and SL0 values of 00, 01 and 10 (or 11) respectively.

TABLE II

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE clock divide by (00 = 2, 01 = 4, 1X = 8)
- IEDG External Interrupt Edge Polarity Select (0 = Rising Edge, 1 = Falling Edge)
- MSEL Selects G5 and G4 as MICROWIRE Signals SK and SO Respectively
- T1C0 Timer T1 Start/Stop Control in Timer Modes 1 and 2
Timer T1 Underflow Interrupt Pending Flag in Timer Mode 3
- T1C1 Timer T1 Mode Control Bit
- T1C2 Timer T1 Mode Control Bit
- T1C3 Timer T1 Mode Control Bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7				Bit 0			

SL1	SL0	SK
0	0	2 x t _C
0	1	4 x t _C
1	x	8 x t _C

Where t_C is the instruction cycle clock

MICROWIRE/PLUS MASTER MODE OPERATION

In the MICROWIRE/PLUS master mode, the BUSY flag of PSW (Processor Status Word) is used to control the shifting

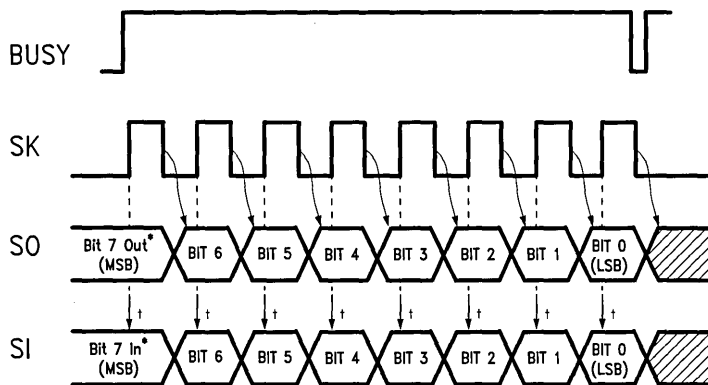
of the MICROWIRE/PLUS 8-bit shift register. Setting the BUSY flag causes the SIOR register to shift out 8 bits of data from SO at the high order end of the shift register. During the same time, 8 new bits of data from SI are shifted into the low order end of the SIOR register. The BUSY flag is automatically reset after the 8 bits of data have been shifted (Figure 2). The COP888XX series of microcontrollers provide a vectored maskable interrupt when the BUSY goes low indicating the end of an 8-bit shift. Input data is clocked into the SIOR register from the SI pin with the rising edge of the SK clock, while the MSB of the SIOR is shifted onto the SO pin with the falling edge of the SK clock. The user may reset the BUSY bit by software to allow less than 8 bits to shift. However, the user should ensure that the software BUSY resets only occurs when the SK clock is low, in order to avoid a narrow SK terminal clock.

MICROWIRE/PLUS SLAVE MODE OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be configured as an input and the SO pin configured as an output by resetting and setting the appropriate bits in the Port G configuration register. The user must set the BUSY flag immediately upon entering the Slave mode. After eight clock pulses the Busy flag will be cleared and the sequence may be repeated. However, in the Slave mode the COP888 series does not shift data if the BUSY flag is reset, whereas the COP820C and COP840C continues to shift regardless of the BUSY flag, if the SK clock is active.

MICROWIRE/PLUS ALTERNATE SK MODE

The COP888XX series of microcontrollers also allow an additional Alternate SK Phase Operation. In the normal mode data is shifted in on the rising edge of the SK clock and data is shifted out on the falling edge of the SK clock (Figure 2). The SIOR register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and data is shifted out on the rising edge of the SK clock (Figure 3).

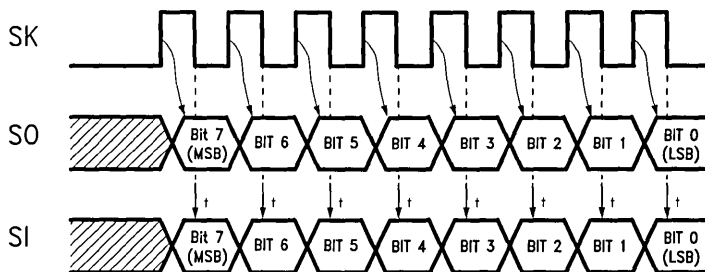


*This bit becomes valid immediately after loading the SIOR register of the transmitting device.

†Arrows indicate points at which SI is sampled.

FIGURE 2. MICROWIRE/PLUS Timing

TL/DD/10252-2



TL/DD/10252-3

↑ Arrows indicates points at which SI is sampled.

FIGURE 3. Alternate Phase SK Clock Timing

A control flag, SKSEL, allows either the normal SK clock or alternate SK clock to be selected. Resetting SKSEL selects the normal SK clock and setting SKSEL selects the alternate SK clock for the MICROWIRE/PLUS logic. The SKSEL flag is mapped into the G6 configuration bit. The SKSEL flag is reset after power up, selecting the normal SK clock signal. The alternate mode facilitates the usage of the MICROWIRE/PLUS protocol for serial data transfer between peripheral devices which are not compatible with the normal SK clock operation, i.e., shifting data out on the falling edge of the SK clock and shifting in data on the rising edge of the SK clock.

MICROWIRE/PLUS SAMPLE PROTOCOL

This section gives a sample MICROWIRE/PLUS protocol using a COP888CL and COP840C. The slave mode operating procedure for this sample protocol is explained, and a timing illustration of the protocol is provided.

1. The MSEL bit in the CNTRL register is set to enable MICROWIRE; G0 (\overline{CS}) and G5 (SK) are configured as inputs and G4 (SO) as an output. G6 (SI) is always an input.
2. Chip Select line (\overline{CS}) from master device is connected to G0 of the slave device. An active-low level on \overline{CS} line causes the slave to interrupt.
3. From the high-to-low transition on the \overline{CS} line, there is no data transfer on the MICROWIRE until time "T" (See Figure 4).
4. The master initiates data transfer on the MICROWIRE by turning on the SK clock.
5. A series of data transfers take place between the master and slave devices.
6. The master pulls the \overline{CS} line high to end the MICROWIRE operation. The slave device returns to normal mode of operation.

SLAVE MODE OPERATING PROCEDURE

1. The MSEL bit in the CNTRL register is set to enable MICROWIRE; G0 (\overline{CS}) and G5 (SK) are configured as inputs and G4 (SO) as an output. G6 (SI) is always an input.
2. Normal mode of operation until interrupted by \overline{CS} going low.

3. Set the BUSY flag and load SIOR register with the data to be sent out on SO. (The shift register shifts 8 bits of data from SO at the high order end of the shift register. During the same time, 8 new bits of data from SI are loaded into the low order end of the shift register.)
4. Wait for the BUSY flag to reset. (The BUSY flag is automatically reset after 8 bits of data have been shifted.)
5. If data is being read in, the user should save contents of the SIOR register.
6. The prearranged set of data transfers are performed.
7. Repeat steps 3 through 6. The user must ensure steps 3 through 6 are performed in time "T" (See Figure 4) as agreed upon in the protocol.

DIFFERENCES BETWEEN COP888 AND COP820/COP840

The COP888 series MICROWIRE/PLUS feature differs from that of the COP820/COP840 in some respects. The COP888 series can be configured to interrupt the processor after the completion of a MICROWIRE/PLUS operation indicated by the BUSY flag going low. The COP888 series supports a vectored interrupt scheme. Two bytes of program memory space are reserved for each interrupt source. The user would do any required context switching and then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS instruction. The addresses of the different interrupt service routines are chosen by the user and stored in a table starting at 0yE0 where "y" depends on the 256 byte block (0y00 to 0yFF) in which the VIS instruction is located. The vector address for the MICROWIRE/PLUS interrupt is 0yF2-0yF3.

Secondly, the COP888 series supports the alternate SK phase mode of MICROWIRE/PLUS operation. This feature facilitates the usage of the MICROWIRE/PLUS protocol for serial data transfer between peripheral devices which are not compatible with the normal SK clock operation, i.e., shifting data out on the falling edge of SK clock and shifting in data on the rising edge of the SK clock.

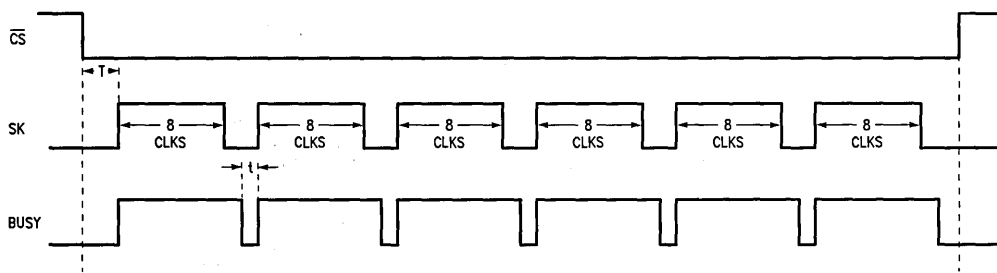


FIGURE 4. MICROWIRE/PLUS Sample Protocol Timing Diagram

TL/DD/10252-4

INTERFACE CONSIDERATIONS

To preserve the integrity of data exchange using MICROWIRE/PLUS, two aspects have to be considered:

1. Serial data exchange timing.
2. Fan-out/fan-in requirements.

Theoretically, infinite devices can access the same interface and be uniquely enabled sequentially in time. In practice, however, the actual number of devices that can access the same serial interface depends on the following: System data transfer rate, system supply requirement, capacitive loading on SK and SO outputs, the fan-in requirements of the logic families or discrete devices to be interfaced.

HARDWARE INTERFACE

For proper data transfer to occur the output should be able to switch between a HIGH level and a LOW level in a predetermined amount of time. The transfer is strictly synchronous and the timing is related to the MICROWIRE/PLUS system clock (SK). For example, if a COPS controller outputs a value at the falling edge of the clock and is latched in by the peripheral device at the rising edge, then the following relationship has to be satisfied:

$$t_{\text{DELAY}} + t_{\text{SETUP}} \leq t_{\text{CK}}$$

where t_{CK} is the time from data output starts to switch to data being latched into the peripheral chip, t_{SETUP} is the setup time for the peripheral device where the data has to be at a valid level, and t_{DELAY} is the time for the output to read the valid level. t_{CK} is related to the system clock provided by the SK pin of the COPS controller and can be increased by increasing the COPS instruction cycle time.

Besides the timing requirements, system supply and fan-out/fan-in requirements also have to be considered when interfacing with MICROWIRE/PLUS. To drive multi-devices on the same MICROWIRE/PLUS, the output drivers of the controller need to source and sink the total maximum leakage current of all the inputs connected to it and keep the signal level within the valid logic "1" and "0" input voltage levels. Thus, if devices of different types are connected to the same serial interface, output driver of the controller must satisfy all the input requirements of each device. Similarly, devices with TRI-STATE® outputs, when connected to the SI input, must satisfy the minimum valid input level of the controller and the maximum TRI-STATE® leakage current of all outputs.

So, for devices that have incompatible input levels or source/sink requirements, external pull-up resistors or buffers are necessary to provide level-shifting or driving.

TABLE III

Features	Part Number								
	DS890XX	MM545X	COP470	COP472	ADC83X (COP430)	COP498/499	COP452L	NMC9306 (COP494)	
GENERAL									
Chip Function	AM/PM PLL	LED Display Driver	VF Display Driver	LCD Display Driver	A/D	RAM & Timer	Frequency Generator	E ² PROM	
Process	ECL	NMOS	PMOS	CMOS	CMOS	CMOS	NMOS	NMOS	
V _{CC} Range	4.75V–5.25V	4.5V–11V	–9.5V to –4.5V	3.0V–5.5V	4.5V–0.3V	2.4V–5.5V	4.5V–6.3V	4.5V–5.5V	
Pinout	20	40	20	20	8/14/20	14/8	14	14	
HARDWARE INTERFACE									
Min V _{IH} /Max V _{IL}	2.1V/0.7V	2.2V/0.8V	–1.5V/–4.0V	0.7 V _{CC} /0.8V	2.0V/0.8V	0.8 V _{CC} /0.4 V _{CC}	2.0V/0.8V	2.0V/0.8V	
SK Clock Range	0–625 kHz	0–500 kHz	0–250 kHz	4–250 kHz	10–200 kHz	4–250 kHz	25–250 kHz	0–250 kHz	
Write Data DI	Setup Min	0.3 μs	0.3 μs	1.0 μs	1.0 μs	0.2 μs	0.4 μs	800 ns	0.4 μs
	Hold Min	0.8 μs	(Note 3)	50 ns	100 ns (Note 1)	0.2 μs	0.4 μs	1.0 μs	0.4 μs
Read Data Prop Delay	(Note 4)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	2 μs (Note 2)	1 μs (Note 2)	2.0 μs	
Chip Enable	Setup	0.275 μs	0.4 μs	1.0 μs Min	1 μs (Note 1)	0.2 μs	0.2 μs (Note 1)	(Note 3)	0.2 μs
	HOLD	0.300 μs	(Note 3)	1.0 μs Min	1 μs (Note 2)	0.2 μs	0 (Note 2)	(Note 3)	0
Max Frequency Range	AM	8 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
	FM	120 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
Max Osc. Freq.	(Note 3)	(Note 3)	250 kHz	(Note 3)	(Note 3)	2.1 MHz (–21) 32 kHz (–15)	256–2100 kHz (–4) 64–525 kHz (–2)	(Note 3)	
SOFT									
Serial I/O Protocol	11D1–D20	1D1–D35	8 Bits At a Time	b1–b40	1xxx	1yxxD6–D0 Start Bit	1yxxxx	1AA–DD	
Instruction/Address Word	None	None	None	None	(Note 4)	(Note 4)	(Note 4)	(Note 4)	

Note 1: Reference to SK rising edge.

Note 2: Reference to SK falling edge.

Note 3: Not defined.

Note 4: See data sheet for different modes of operation.

TYPICAL APPLICATIONS

A whole family of off-the-shelf devices exist that are directly compatible with MICROWIRE/PLUS protocol. This allows direct interface with the COP800 family of microcontrollers. Table III provides a summary of the existing devices, their function and specification.

NMC9306-COP888CG INTERFACE

The pin connection involved in interfacing an NMC9306 (COP494), a 256 bit E²PROM, with the COP888CG microcontroller is shown in Figure 5. Some notes on the NMC9306 interface requirements are:

1. The SK clock frequency should be in the 0 kHz–250 kHz range.
2. \overline{CS} low period following an Erase/Write instruction must not exceed 30 ms maximum. It should be set at typical or minimum specification of 10 ms.

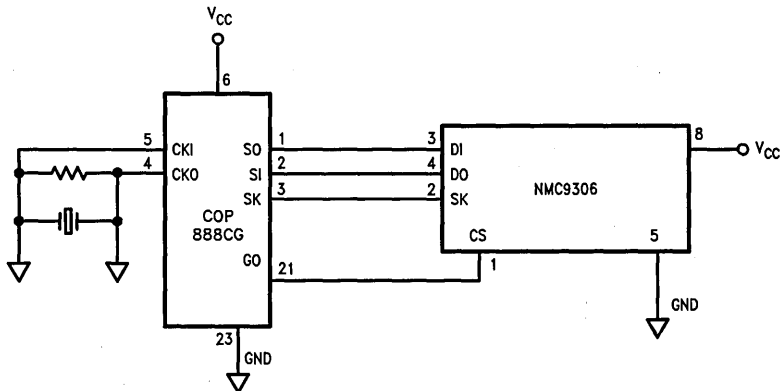


FIGURE 5. NMC9306-COP888CG Interface

TL/DD/10252-5

Instruction Set

Commands	Start Bit	Opcode	Address	Comments
READ	1	0000	A3A2A1A0	Read Register 0–15
WRITE	1	1000	A3A2A1A0	Write Register 0–15
ERASE	1	0100	A3A2A1A0	Erase Register 0–15
EWEN	1	1100	00 01	Write/Erase Enable
ENDS	1	1100	00 10	Write/Erase Disable
***WRAL	1	1100	01 00	Write All Registers
ERAL	1	1100	01 01	Read All Registers

Where A3A2A1A0 corresponds to one of the sixteen 16-bit registers.

All commands, data in, and data out are shifted in/out on the rising edge of the SK clock.

Write/Erase is then done by pulsing \overline{CS} low for 10 ms.

All instructions are initiated by a LO–HI transition on \overline{CS} followed by a LO–HI transition on DI.

READ— After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE— Write command shifted in followed by data in (16 bits) the \overline{CS} pulsed low for 10 ms minimum.

3. The start bit on DI must be set by a "0" to "1" transition following a \overline{CS} enable ("0" to "1") when executing any instruction. One \overline{CS} enable transition can only execute one instruction.

4. In the read mode, following an instruction and data train, the DI can be a "don't care", while the data is being outputted, i.e., for the next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.

5. The data out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential.

If \overline{CS} is held on after all 16 of the data bits have been outputted, the DO will output the state of DI until another \overline{CS} LO to HI transition starts a new instruction cycle.

6. After a read cycle, the \overline{CS} must be brought low for one SK clock cycle before another instruction cycle starts.

ERASE/ERASE ALL— Command shifted in followed by \overline{CS} low.

WRITE ALL— Pulsing \overline{CS} low for 10 ms.

ENABLE/DISABLE— Command shifted in.

A detailed explanation of the E²PROM timing diagrams, instruction set and the various considerations could be found in the NMC9306 data sheet. A source listing of the software to interface the NMC9306 with the COP888CG is provided.

SOURCE LISTING

```

.INCLD COP888.INC
;
;This program provides in the form of subroutines, the ability to erase,enable, disable, read and write to the COP494 EEPROM.
;
;
SNDBUF = 0           ;CONTAINS THE COMMAND BYTE TO BE WRITTEN TO COP494
RDATL  = 1           ;LOWER BYTE OF THE COP494 REGISTER DATA READ
RDATH  = 2           ;UPPER BYTE OF THE COP494 REGISTER DATA READ
WDATL  = 3           ;LOWER BYTE OF THE DATA TO BE WRITTEN TO COP494
                    ;REGISTER
WDATH  = 4           ;UPPER BYTE OF THE DATA TO BE WRITTEN TO COP494
                    ;REGISTER
ADDRESS = 5          ;THE LOWER 4-BITS OF THIS LOCATION CONTAIN THE
                    ;ADDRESS
                    ;OF THE COP494 REGISTER TO BE READ/WRITTEN
FLAGS  = 6           ;USED FOR SETTING UP FLAGS
                    ;
                    ; FLAG VALUE   ACTION
                    ;-----
                    ; 00          ERASE,ENABLE,DISABLE,ERASE ALL
                    ; 01          READ CONTENTS OF COP494 REGISTER
                    ; 03          WRITE TO COP494 REGISTER
                    ; OTHERS     ILLEGAL COMBINATION

DLYH  = 0F0
DLYL  = 0F1
;
;THE INTERFACE BETWEEN THE COP888CG AND THE COP494 (256-BIT EEPROM) CONSISTS OF FOUR LINES. THE
;G0 (CHIP SELECT LINE), G4 (SERIAL OUT SO), G5 (SERIAL CLOCK SK) ;AND G6 (SERIAL IN SI).
;
;  INITIALIZATION
;
;          LD          PORTGC,#031          ;Setup G0,G4,G5 as outputs
;          LD          PORTGD,#00          ;Initialize G data reg to zero
;          LD          CNTRL,#08           ;Enable MSEL, select MW rate of 2tc
;          LD          B,#PSW
;          LD          X,#SIOR
;
;THIS ROUTINE ERASES THE MEMORY LOCATION POINTED TO BY THE ADDRESS CONTAINED IN THE LOCATION
;"ADDRESS". THE LOWER NIBBLE OF "ADDRESS" CONTAINS THE COP494 REGISTER ADDRESS AND THE UPPER NIBBLE
;SHOULD BE SET TO ZERO.
;
ERASE:   LD          A,ADDRESS
         OR          A,#0C0
         X          A,SNDBUF
         LD          FLAGS,#0
         JSR        INIT
         RET
;
;THIS ROUTINE ENABLES PROGRAMMING OF THE COP494. PROGRAMMING MUST BE PRECEDED ONCE BY A
;PROGRAMMING ENABLE (EWEN).
;
EWEN:   LD          SNDBUF,#030

```

TL/DD/10252-6

```

        LD          FLAGS,#0
        JSR        INIT
        RET

;
;THIS ROUTINE DISABLES PROGRAMMING OF THE COP494.
;
EWDS:   LD          SNDBUF,#0
        LD          FLAGS,#0
        JSR        INIT
        RET

;
;THIS ROUTINE ERASES ALL REGISTERS OF THE COP494.
;
ERAL:   LD          SNDBUF,#020
        LD          FLAGS,#0
        JSR        INIT
        RET

;
;THIS ROUTINE READS THE CONTENTS OF THE COP494 REGISTER. THE COP494 ADDRESS IS SPECIFIED IN THE
;LOWER NIBBLE OF LOCATION "ADDRESS". THE UPPER NIBBLE SHOULD BE SET TO ZERO. THE 16-BIT CONTENTS OF
;THE COP494 REGISTER ARE STORED IN RDATL AND RDATH.
;
READ:   LD          A,ADDRESS
        OR          A,#080
        X          A,SNDBUF
        LD          FLAGS,#1
        JSR        INIT
        RET

;
;THIS ROUTINE WRITES A 16-BIT VALUE STORED IN WDATL AND WDATH TO THE COP494 REGISTER WHOSE ADDRESS
;IS CONTAINED IN THE LOWER NIBBLE OF THE LOCATION "ADDRESS". THE UPPER NIBBLE OF ADDRESS LOCATION
;SHOULD BE SET TO ZERO.
;
WRITE:  LD          A,ADDRESS
        OR          A,#040
        X          A,SNDBUF
        LD          FLAGS,#3
        JSR        INIT
        RET

;
;THIS ROUTINE SENDS OUT THE START BIT AND THE COMMAND BYTE. IT ALSO DECIPHERS THE CONTENTS OF THE
;FLAG LOCATION AND TAKES A DECISION REGARDING WRITE, READ OR RETURN TO THE CALLING ROUTINE.
;
INIT:   SBIT        0,PORTGD          ;SET CHIP SELECT HIGH
        LD          SIOR,#001        ;LOAD SIOR WITH START BIT
        SBIT        BUSY,[B]         ;SEND OUT THE START BIT
PUNT1:  IFBIT       BUSY,[B]
        JP          PUNT1
        LD          A,SNDBUF
        X          A,[X]             ;LOAD SIOR WITH COMMAND BYTE
        SBIT        BUSY,[B]         ;SEND OUT COMMAND BYTE
PUNT2:  IFBIT       BUSY,[B]
        JP          PUNT2
        IFBIT       0,FLAGS          ;ANY FURTHER PROCESSING ?

```

```

        JP      NOTDON      ;YES
        RBIT   0,PORTGD    ;NO, RESET CS AND RETURN
        RET

;
NOTDON:  IFBIT   1,FLAGS    ;READ OR WRITE?
        JP     WR494       ;JUMP TO WRITE ROUTINE
        LD     SIOR,#000   ;NO, READ COP494
        SBIT   BUSY,PSW   ;DUMMY CLOCK TO READ ZERO
        RBIT   BUSY,[B]
        SBIT   BUSY,[B]
PUNT3:   IFBIT   BUSY,[B]
        JP     PUNT3
        X     A,[X]
        SBIT   BUSY,[B]
        X     A,RDATH
PUNT4:   IFBIT   BUSY,[B]
        JP     PUNT4
        LD     A,[X]
        X     A,RDATL
        RBIT   0,PORTGD
        RET

;
WR494:   LD     A,WDATH
        X     A,[X]
        SBIT   BUSY,[B]
PUNT5:   IFBIT   BUSY,[B]
        JP     PUNT5
        LD     A,WDATL
        X     A,[X]
        SBIT   BUSY,[B]
PUNT6:   IFBIT   BUSY,[B]
        JP     PUNT6
        RBIT   0,PORTGD
        JSR   TOUT
        RET

;
;ROUTINE TO GENERATE DELAY FOR WRITE
;
TOUT:    LD     DLYH,#00A
WAIT:    LD     DLYL,#0FF
WAIT1:   DRSZ   DLYL
        JP     WAIT1
        DRSZ   DLYH
        JP     WAIT
        RET
        .END

```

TL/DD/10252-8

COP472-COP820 Interface

The pin connection required for interfacing COP472-3 Liquid Crystal Display (LCD) Controller with COP820C microcontroller is shown in Figure 6. The COP472-3 drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. One COP472-3 can drive 36 segments and two or more COP472-3's can be cascaded to drive additional segments as long as the output loading capacitance does not exceed specifications.

The COP472-3 requires 40 information bits: 36 data and 4 control. The function of each control bit is described briefly. Data is loaded in serially, in sets of eight bits. Each set of segment data is in the following format:

SA	SB	SC	SD	SE	SF	SG	SH
----	----	----	----	----	----	----	----

Data is shifted into an eight bit shift register. The first bit of data is for segment H, digit 1, and the eight bit is for segment A, digit 1. A set of eight bits are shifted in and then

loaded into the digit one latches. The second, third, and fourth set is then loaded sequentially. The fifth set of data bits contain special segment data and control data in the following format:

SYNC	Q7	Q6	X	SP4	SP3	SP2	SP1
------	----	----	---	-----	-----	-----	-----

The first four bits shifted in contain the special character segment data. The fifth bit is not used. The sixth and seventh bits program the COP472-3 as a stand alone LCD driver or as a master or slave for cascading COP472-3's. The Table IV summarizes the function of bits six and seven.

The eight bit is used to synchronize two COP472-3's to drive an 8½ digit display. A detailed explanation of the various timing diagrams, loading sequence and segment/backplane multiplex scheme can be found in the data sheets of COP472-3. The source listing of the software used in the interface is provided.

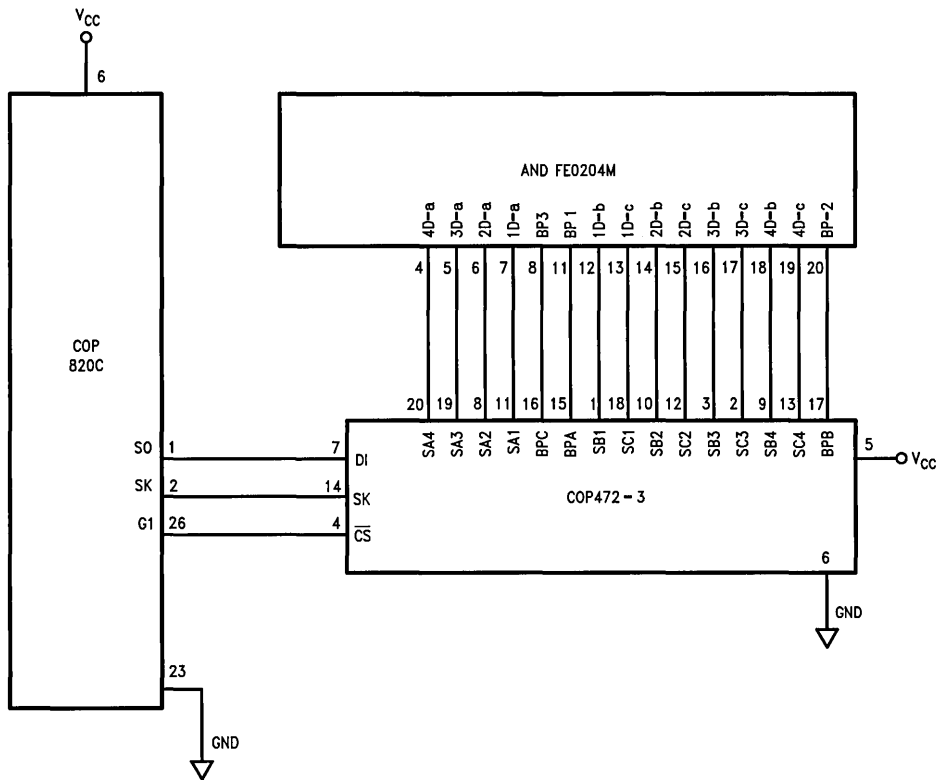


FIGURE 6. COP472-COP820C Interface

TL/DD/10252-12

SOURCE LISTING

;THIS PROGRAM DISPLAYS FOUR DIGITS OF THE RAM SPECIFIED BY; THE ADDRESS POINTER "HEAD" ON A 4 DIGIT 3
;DECIMAL POINT (MULTIPLICED) LCD DISPLAY. THE DATA STREAM IS SENT OUT SERIALY THROUGH THE
;MICROWIRE/PLUS INTERFACE TO THE COP472 LCD DISPLAY DRIVER. NOTE: THE RAM CONTENTS SHOULD BE
;BETWEEN "0" AND "F".

TL/DD/10252-9

```

:
:      .TITLE      LCD
:
:
:
:      .CHIP      820
:
:
:      PORTGD     = 0D4      ;PORT G DATA REGISTER
:      PORTGC     = 0D5      ;PORT G CONFIGURATION
:
:
:      SIO        = 0E9      ;MICROWIRE SHIFT REGISTER
:
:
:      PSW        = 0EF      ;PSW REGISTER
:      CNTRL      = 0EE      ;CNTRL REGISTER
:
:
:      CONTRL     = 04      ;MEMORY LOCATION FOR THE
:                        ;COP472 CONTROL WORD
:      HEAD       = 00      ;STARTING MEMORY LOC FOR
:                        ;DATA TO BE DISPLAYED
:      MEMSTR     = 05      ;STARTING MEMORY LOC FOR
:                        ;STORING SEGMENT DATA
:      MEMEND     = 08      ;MEMORY LOC FOR LAST
:                        ;SEGMENT DATA
:
:
: START:  LD      CNTRL,#08      ;SET MSEL BIT IN CNTRL
:         LD      PORTGC,#032   ;SET G5,G4& G1 AS OUTPUTS
:         LD      CNTRL,#0FC    ;SET COP472 IN STAND ALONE MODE
:
:
:
:

```

;THIS ROUTINE GETS THE SEGMENT DATA FOR RAM DIGITS POINTED BY B REGISTER AND STORES IN RAM MEMORY
;POINTED BY X REGISTER

```

:
:
: AGAIN:  LD      B,#HEAD      ;POINTER TO START ADDRESS
:         LD      X,#MEMSTR    ;POINTER TO STORE ADDRESS
: NEXDIG: LD      A,[B+]      ;LOAD A WITH RAM DIGIT AND
:                        ;INCREMENT B POINTER
:
:         ADD     A,#0F0      ;ADD OFFSET TO THE DIGIT
: LAID     ;LOOKUP SEGMENT DATA TO A
: X        A,[X+]            ;STORE IN MEMORY
: IFBNE   #04               ;CHECK FOR END OF FOUR
:                        ;DIGITS AND REPEAT
:         JP     NEXDIG      ;IF NECESSARY
:
:
:
:

```

; THIS ROUTINE DISPLAYS THE CONTENTS OF FOUR MEMORY LOCATION
; ON THE LCD DISPLAY.

```

:
:
: DSP:   LD      B,#MEMEND     ;LOAD THE START ADDRESS
:         RBIT   1,PORTGD     ;BIT G1 IS USED TO SELECT
:                        ;COP472 (PIN 4)

```

TL/DD/10252-10


```

REPEAT:  LD      A,[B-]          ;SEGMENT DATA TO A
          X      A,SIO          ;LOAD THE SIO REGISTER
          SBIT   #2,PSW        ;SET BUSY BIT IN PSW
WAIT:    IFBIT  #2,PSW        ;WAIT TILL SHIFTING IS
          JP     WAIT          ;COMPLETE
          IFBNE  #04           ;CHECK FOR END OF FOUR
          JP     REPEAT        ;DIGITS AND REPEAT
          SBIT   1,PORTGD      ;DESELECT COP472
LOOP:    JP     LOOP          ;DONE DISPLAYING

```

```

; STORE THE LOOKUP TABLE FOR SEGMENT DATA IN ROM LOCATION 0F0

```

```

.-0F0

```

```

.BYTE 03F,006,05B,04F ;DATA FOR 0,1,2,3
.BYTE 066,06D,07D,07 ;DATA FOR 4,5,6,7
.BYTE 07F,067,077,07C ;DATA FOR 8,9,A,B
.BYTE 039,05E,079,071 ;DATA FOR C,D,E,F

```

```

.END

```

TL/DD/10252-11

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162

Voice (408) 721-5582

For Additional Information, Please Contact Factory



OVERVIEW

This application note discusses the various arithmetic operations for National Semiconductor's COP800 family of 8-bit microcontrollers. These arithmetic operations include both binary and BCD (Binary Coded Decimal) operation. The four basic arithmetic operations (add, subtract, multiply, divide) are outlined in detail, with several examples shown for both binary and BCD addition and subtraction. Multiplication, division, and BCD conversion algorithms are also provided. Both BCD to binary and binary to BCD conversion subroutines are included, as well as the various multiplication and division subroutines.

Four sets of optimal subroutines are provided for

1. Multiplication
2. Division
3. Decimal (Packed BCD) to binary conversion
4. Binary to decimal (Packed BCD) conversion

One class of subroutines is optimized for minimal COP800 program code, while the second class is optimized for minimal execution time in order to optimize throughput time.

This application note is organized in four different sections. The first section outlines various addition and subtraction routines, including both binary and BCD (Binary Coded Decimal). The second section outlines the multiplication algorithm and provides several optimal multiply subroutines for 1, 2, 3, and 4 byte operation. The third section outlines the division algorithm and provides several optimal division subroutines for 1, 2, 3, and 4 byte operation. The fourth section outlines both the decimal (Packed BCD) to binary and binary to decimal (Packed BCD) conversion algorithms. This section provides several optimal subroutines for these BCD conversions.

The COP800 arithmetic instructions include the Add (ADD), Add with Carry (ADC), Subtract with Carry (SUBC), Increment (INCR), Decrement (DECR), Decimal Correct (DCOR),

Clear Accumulator (ACC), Set Carry (SC), and Reset Carry (RC). The shift and rotate instructions, which include the Rotate Right through Carry (RRC) and the Swap Accumulator Nibbles (SWAP), may also be considered as arithmetic instruction variations. The RRC instruction is instrumental in writing a fast multiply routine.

1.0 BINARY AND BCD ADDITION AND SUBTRACTION

In subtraction, a borrow is represented by the absence of a carry and vice versa. Consequently, the carry flag needs to be set (no borrow) before a subtraction, just as the carry flag is reset before an addition. The ADD instruction does not use the carry flag as an input, nor does it change the carry flag. It should also be noted that both the carry and half carry flags (bits 6 and 7, respectively, of the PSW control register) are cleared with reset, and remain unchanged with the ADD, INC, DEC, DCOR, CLR and SWAP instructions. The DCOR instruction uses both the carry and half carry flags. The SC instruction sets both the carry and half carry flags, while the RC instruction resets both these flags.

The following program examples illustrate additions and subtractions of 4-byte data fields in both binary and BCD (Binary Coded Decimal). The four bytes from data memory locations 24 through 27 are added to or subtracted from the four bytes in data memory locations 16 through 19. The results replace the data in memory locations 24 through 27.

These operations are performed both in Binary and BCD. It should be noted that the BCD pre-conditioning of Adding (ADD) the hex 66 is only necessary with the BCD addition, not with the BCD subtraction. The (Binary Coded Decimal) DCOR (Decimal Correct) instruction uses both the carry and half carry flags as inputs, but does not change the carry and half carry flags. Also note that the #12 with the IFBNE instruction represents $28 - 16$, since the IFBNE operand is modulo 16 (remainder when divided by 16).

BINARY ADDITION:

```

LD      X,#16      ; NO LEADING ZERO
LD      B,#24      ; INDICATES DECIMAL
RC      ; RESET CARRY TO START
LOOP:  LD      A,[X+] ; [X] TO ACC
      ADC     A,[B]  ; ADD [B] TO ACC
      X      A,[B+] ; RESULT TO [B]
      IFBNE  #12    ; IF STILL IN DATA FIELD
      JF     LOOP   ; JUMP BACK TO REPEAT LOOP
      IFC    ; IF TERMINAL CARRY,
      JF     OVFLOW ; JUMP TO OVERFLOW

```

BINARY SUBTRACTION:

```

LD      X,#010     ; LEADING ZERO
LD      B,#018     ; INDICATES HEX
SC      ; RESET BORROW TO START
LOOP:  LD      A,[X+] ; [X] TO ACC
      SUBC   A,[B]  ; SUBTRACT [B] FROM ACC
      X      A,[B+] ; RESULT TO [B]
      IFBNE  #12    ; IF STILL IN DATA FIELD
      JF     LOOP   ; JUMP BACK TO REPEAT LOOP
      IFNC   ; IF TERMINAL BORROW,
      JF     NEGRSLT ; JUMP TO NEGATIVE RESULT

```

BCD ADDITION:

```

LD      X,#010     ; LEADING ZERO
LD      B,#018     ; INDICATES HEX
RC      ; RESET CARRY TO START
LOOP:  LD      A,[X+] ; [X] TO ACC
      ADD     A,#066 ; ADD HEX 66 TO ACC
      ADC     A,[B]  ; ADD [B] TO ACC
      DCOR   A      ; DECIMAL CORRECT RESULT
      X      A,[B+] ; RESULT TO [B]
      IFBNE  #12    ; IF STILL IN DATA FIELD
      JF     LOOP   ; JUMP BACK TO REPEAT LOOP
      IFC    ; IF TERMINAL CARRY
      JF     OVFLOW ; JUMP TO OVERFLOW

```

BCD SUBTRACTION:

```

LD      X,#16      ; NO LEADING ZERO
LD      B,#24      ; INDICATES DECIMAL
C      ;
LOOP:  LD      A,[X+] ; [X] TO ACC
      SUBC   A,[B]  ; SUBTRACT [B] FROM ACC
      DCOR   A      ; DECIMAL CORRECT RESULT
      X      A,[B+] ; RESULT TO [B]
      IFBNE  #12    ; IF STILL IN DATA FIELD
      JF     LOOP   ; JUMP BACK TO REPEAT LOOP
      IFNC   ; IF TERMINAL BORROW
      JF     NEGRSLT ; JUMP TO NEGATIVE RESULT

```

The astute observer will notice that these previous additions and subtractions are not "adding machine" type arithmetic operations in that the result replaces the second operand rather than the first. The following program examples illus-

trate "adding machine" type operation where the result replaces the first operand. With subtraction, this entails the result replacing the minuend rather than the subtrahend. Note that the B and X pointers are now reversed.

BINARY ADDITION:

```

LD      B, #16      ; B POINTER AT FIRST OPERAND
LD      X, #24      ; X POINTER AT SECOND OPERAND
RC      ; RESET CARRY TO START
LOOP:   LD      A, [X+] ; [X] TO ACC
        ADC     A, [B]  ; ADD [B] TO ACC
        X      A, [B+] ; RESULT TO [B]
        IFBNE  #4      ; IF STILL IN DATA FIELD
        JP     LOOP    ; JUMP BACK TO REPEAT LOOP
        IFC    ; IF TERMINAL CARRY
        JP     OVFLOW  ; JUMP TO OVERFLOW

```

BINARY SUBTRACTION:

```

LD      B, #010     ; B POINTER AT FIRST OPERAND
LD      X, #018     ; X POINTER AT SECOND OPERAND
SC      ; RESET BORROW TO START
LOOP:   LD      A, [X+] ; [X] TO ACC
        X      A, [B]  ; EXCHANGE [B] AND ACC
        SUBC   A, [B]  ; SUBTRACT [B] FROM ACC
        X      A, [B+] ; RESULT TO [B]
        IFBNE  #4      ; IF STILL IN DATA FIELD
        JP     LOOP    ; JUMP BACK TO REPEAT LOOP
        IFNC   ; IF TERMINAL BORROW
        JP     NEGRSLT ; JUMP TO NEGATIVE RESULT

```

BCD ADDITION:

```

LD      B, #010     ; B POINTER AT FIRST OPERAND
LD      X, #018     ; X POINTER AT SECOND OPERAND
RC      ; RESET CARRY TO START
LOOP:   LD      A, [X+] ; [X] TO ACC
        ADD    A, #066 ; ADD HEX66 TO ACC
        ADC    A, [B]  ; ADD [B] TO ACC
        DCOR   A      ; DECIMAL CORRECT RESULT
        X      A, [B+] ; RESULT TO [B]
        IFBNE  #4      ; IF STILL IN DATA FIELD
        JP     LOOP    ; JUMP BACK TO REPEAT LOOP
        IFC    ; IF TERMINAL CARRY
        JP     OVFLOW  ; JUMP TO OVERFLOW

```

BCD SUBTRACTION:

```

LD      B, #16      ; B POINTER AT FIRST OPERAND
LD      X, #24      ; X POINTER AT SECOND OPERAND
SC      ; RESET BORROW TO START
LOOP:   LD      A, [X+] ; [X] TO ACC
        X      A, [B]  ; EXCHANGE [B] AND ACC
        SUBC   A, [B]  ; SUBTRACT [B] FROM ACC
        DCOR   A      ; DECIMAL CORRECT RESULT
        X      A, [B+] ; RESULT TO [B]
        IFBNE  #4      ; IF STILL IN DATA FIELD
        JP     LOOP    ; JUMP BACK TO REPEAT LOOP
        IFNC   ; IF TERMINAL BORROW
        JP     NEGRSLT ; JUMP TO NEGATIVE RESULT

```

Let us now consider a hybrid arithmetic example, where we wish to add five successive bytes of a data table in ROM program memory to a two byte sum, and then subtract the SUM result from a two byte total TOT. Let us further assume

that the ROM table is located starting at program memory address 0401, while SUM and TOT are at RAM data memory locations [1, 0] and [3, 2] respectively, and that we wish to encode the program as a subroutine.

ROM Table:

```
. = 0401
. Byte 102
. Byte 41
. Byte 31
. Byte 26
. Byte 5
```

ROM Table Accessed Top Down

```
SUMLO = 0
SUMHI = 1
TOTLO = 2
TOTHI = 3
```

```
ARITH1: LD      X,#5          ; SET UP ROM TABLE POINTER
        LD      B,#0        ; SET UP SUM POINTER
LOOP:   RC      ; RESET CARRY TO START ADDITION
        LD      A,X         ; ROM POINTER TO ACC
        LAID    ; TABLE VALUE FROM ROM TO ACC
        ADC    A,[B]        ; ADD SUMLO TO ACC
        X      A,[B+]       ; RESULT TO SUMLO
        CLR    A            ; CLEAR ACC
        ADC    A,[B]        ; ADD SUMHI TO ACC
        X      A,[B-]       ; RESULT TO SUMHI
        DRSZ   X            ; DECR AND TEST ROM PTR FOR ZERO
        JP     LOOP        ; JUMP BACK TO REPEAT LOOP
        ; IF X PTR NOT ZERO
        SC     ; RESET BORROW TO START SUBTRACTION
        LD     B,#2        ; SET UP TOT POINTER
LUP:    LD     A,[X+]       ; SUBTRAHEND (SUM) TO ACC
        X      A,[B]        ; REVERSE OPERANDS
        SUBC   A,[B]        ; FOR SUBTRACTION
        X      A,[B+]       ; RESULT TO TOT
        IFBNE #4           ; IF STILL IN TOT FIELD
        JP     LUP         ; JUMP BACK TO REPEAT LUP
        RET    ; RETURN FROM SUBROUTINE
```

2.0 MULTIPLICATION

The COP800 multiplications are all based on starting the multiplier in the low order end of the double length product space. The high end of the double length product space is initially cleared, and then the double length product is shifted right one bit. The bit shifted out from the low order end represents the low order bit of the multiplier. If this bit is a "1", the multiplicand is added to the high end of the double length product space. The entire shifting process and the conditional addition of the multiplicand to the upper end of the double length product is then repeated. The number of shift cycles is equal to the number of bit positions in the multiplier plus one extra shift cycle. This extra terminal shift cycle is necessary to correctly align the resultant product.

Note that an M byte multiplicand multiplied by an N byte multiplier will result in an M + N byte double length product. However, these multiplication subroutines will only use 2M + N + 1 bytes of RAM memory space, since the multiplier initially occupies the low order end of the double length product. The one extra byte is necessary for the shift counter CNTR.

The minimal code (28 byte) general multiplication subroutine is shown with two different examples, MY2448 and MY4824. Both examples multiply 24 bits by 48 bits. The MY2448 subroutine uses the 48-bit operand as the multiplier, and consequently uses minimal RAM as well as minimal program code. The MY4824 subroutine uses the 24-bit operand as the multiplier, and consequently executes considerably faster than the minimal RAM MY2448 subroutine.

- MPY88 — 8 by 8 Multiplication Subroutine
 - 19 Bytes
 - 180 Instruction Cycles
 - Minimum Code
- MLT88 — Fast 8 by 8 Multiplication Subroutine
 - 42 Bytes
 - 145 Instruction Cycles
- VFM88 — Very Fast 8 by 8 Multiply Subroutine
 - 96 Bytes
 - 116 Instruction Cycles
- MPY168 — Fast 16 by 8 Multiplication Subroutine
 - 36 Bytes
 - 230 Instruction Cycles Average
 - 254 Instruction Cycles Maximum

MPY816 (or MPY824, MPY832)

- 8 by 16 (or 24, 32) Multiply Subroutine
- 22 Bytes
- 589 (or 1065, 1669) Instruction Cycles Average
- 597 (or 1077, 1685) Instruction Cycles Maximum
- Minimum Code, Minimum RAM
- Extendable Routine for MPY8XX by Changing Parameters, with Number of Bytes (22) Remaining a Constant

MPY248

- Fast 24 by 8 Multiplication Subroutine
- 47 Bytes
- 289 Instruction Cycles Average
- 333 Instruction Cycles Maximum

MX1616

- Fast 16 by 16 Multiplication Subroutine
- 39 Bytes
- 498 Instruction Cycles Average
- 546 Instruction Cycles Maximum

MP1616

- 16 by 16 Multiplicand Subroutine
- 29 Bytes
- 759 Instruction Cycles Average
- 807 Instruction Cycles Maximum
- Almost Minimum Code

MY1616 (or MY1624, MY1632)

- 28 Bytes
- 16 by 16 (or 24, 32) Multiply Subroutine
- 861 (or 1473, 2213) Inst. Cycles Average
- 1029 (or 1725, 2549) Inst. Cycles Maximum
- Minimum Code, Minimum RAM
- Extendable Routine for MY16XX by Changing Parameters, with Number of Bytes (28) Remaining a Constant

Minimal general multiplication subroutine for any number of bytes in multiplicand and multiplier

- 28 Bytes
- Minimum Code
- MY2448 Used as First Example, with Minimum RAM and
 - 4713 Instruction Cycles Average
 - 5457 Instruction Cycles Maximum
- MY4824 Used as Second Example, with Non Minimal RAM and
 - 2751 Instruction Cycles Average
 - 3483 Instruction Cycles Maximum

MPY88—8 BY 8 MULTIPLICATION SUBROUTINE

```

MINIMUM CODE
19 BYTES
180 INSTRUCTION CYCLES
MULTIPLICAND IN [0]      (ICAND)
MULTIPLIER IN [1]       (IER)
PRODUCT IN [2,1]        (PROD)
MPY88:  LD          CNTR,#9      ; LD CNTR WITH LENGTH OF
RC          ; MULTIPLIER FIELD + 1
LD          B,#2
M88LUP: CLR          A          ; CLEAR UPPER PRODUCT
RRC          A          ; RIGHT SHIFT
X           A,[B-]        ; UPPER PRODUCT
LD          A,[B]
RRC          A          ; RIGHT SHIFT LOWER
X           A,[B-]        ; PRODUCT/MULTIPLIER
CLR          A          ; CLR ACC AND TEST LOW
IFC          ; ORDER MULTIPLIER BIT
LD          A,[B]        ; MULTIPLICAND TO ACC IF
RC          ; LOW ORDER BIT = 1
LD          B,#2        ; ADD MULTIPLICAND TO
ADC          A,[B]        ; UPPER PRODUCT
DRSZ        CNTR        ; DECREMENT AND TEST
JP          M88LUP       ; CNTR FOR ZERO
RET         ; RETURN FROM SUBROUTINE

```

MLT88—FAST 8 BY 8 MULTIPLICATION SUBROUTINE

42 BYTES

145 INSTRUCTION CYCLES

MULTIPLICAND IN [0] (ICAND)

MULTIPLIER IN [1] (IER)

PRODUCT IN [2,1] (PROD)

```

MLT88:  LD      CNTR,#3      ; LOAD CNTR WITH
        RC          ; 1/3 OF LENGTH OF
        LD      B,#2      ; (MULTIPLIER FIELD + 1)
        CLR     A         ; CLEAR UPPER PRODUCT
;
ML88LP: RRC     A         ; RIGHT SHIFT ***
        X      A,[B-]    ; UPPER PRODUCT
        LD      A,[B]
        RRC     A         ; RIGHT SHIFT LOWER
        X      A,[B-]    ; PRODUCT/MULTIPLIER
        CLR     A         ; CLR ACC AND TEST LOW
        IFC    ; ORDER MULTIPLIER BIT
        LD      A,[B]    ; MULTIPLICAND TO ACC IF
        RC          ; LOW ORDER BIT = 1
        LD      B,#2      ; ADD MULTIPLICAND TO
        ADC     A,[B]    ; UPPER PRODUCT ***
;
        RRC     A         ; REPEAT THE ABOVE
        X      A,[B-]    ; 11 BYTE
        LD      A,[B]    ; 13 INSTRUCTION
        RRC     A         ; CYCLE PROGRAM
        X      A,[B-]    ; SECTION (WITH
        CLR     A         ; THE *** DELIMITERS)
        IFC    ; TWICE MORE FOR A
        LD      A,[B]    ; TOTAL OF THREE TIMES
        RC
        LD      B,#2
        ADC     A,[B]    ; END OF SECOND REPEAT
;
        RRC     A         ; START OF THIRD REPEAT
        X      A,[B-]
        LD      A,[B]
        RRC     A
        X      A,[B-]
        CLR     A
        IFC
        LD      A,[B]
        RC
        LD      B,#2
        ADC     A,[B]    ; END OF THIRD REPEAT
;
        DRSZ   CNTR      ; DECREMENT AND TEST
        JMP    ML88LP    ; CNTR FOR ZERO
        RET          ; RETURN FROM SUBROUTINE

```


VFM88—VERY FAST 8 BY 8 MULTIPLY SUBROUTINE

96 BYTES
116 INSTRUCTION CYCLES

```

MULTIPLICAND IN [0]      (ICAND)
MULTIPLIER IN [1]       (IER)
PRODUCT IN [2,1]       (PROD)
VFM88:  RC
        LD      B,#2
        LD      [B-],#0    ; CLEAR UPPER PRODUCT
        LD      A,[B]
        RRC     A          ; RIGHT SHIFT LOWER
        X      A,[B-]     ; PRODUCT/MULTIPLIER
        CLR     A          ; CLR ACC AND TEST LOW
        IFC    A          ; ORDER MULTIPLIER BIT
        LD      A,[B]     ; MULTIPLICAND TO ACC IF
        RC     A          ; LOW ORDER BIT = 1
        LD      B,#2     ; ADD MULTIPLICAND TO
        ADC     A,[B]     ; UPPER PRODUCT
;
        RRC     A          ; RIGHT SHIFT ***
        X      A,[B-]     ; UPPER PRODUCT
        LD      A,[B]
        RRC     A          ; RIGHT SHIFT LOWER
        X      A,[B-]     ; PRODUCT/MULTIPLIER
        CLR     A          ; CLR ACC AND TEST LOW
        IFC    A          ; ORDER MULTIPLIER BIT
        LD      A,[B]     ; MULTIPLICAND TO ACC IF
        RC     A          ; LOW ORDER BIT = 1
        LD      B,#2     ; ADD MULTIPLICAND TO
        ADC     A,[B]     ; UPPER PRODUCT ***
;
; THE ABOVE 11 BYTE, 13 INSTRUCTION CYCLE SECTION WITH THE ***
; DELIMITERS REPRESENTS THE PROCESSING FOR ONE MULTIPLIER BIT.
;
;
; ---          ; REPEAT THE
;              ; ABOVE SECTION
; ---          ; SIX MORE TIMES,
;              ; FOR A TOTAL
; ---          ; OF SEVEN TIMES
;
        RRC     A          ; RIGHT SHIFT
        X      A,[B-]     ; UPPER PRODUCT
        LD      A,[B]
        RRC     A          ; RIGHT SHIFT LOWER
        X      A,[B]     ; PRODUCT/MULTIPLIER
        RET
;
;
;

```

MPY168—FAST 16 BY 8 MULTIPLICATION SUBROUTINE

36 BYTES
 230 INSTRUCTION CYCLES AVERAGE
 254 INSTRUCTION CYCLES MAXIMUM

	MULTIPLICAND IN [1,0]	(ICAND)
	MULTIPLIER IN [2]	(IER)
	PRODUCT IN [4,3,2]	(PROD)
MPY168:	LD CNTR,#9	; LD CNTR WITH LENGTH OF
	RC	; MULTIPLIER FIELD + 1
	LD B,#4	
	LD [B-],#0	; CLEAR
	LD [B-],#0	; UPPER PRODUCT
	JP MF168S	
M168LP:	RRC A	; RIGHT SHIFT UPPER
	X A,[B-]	; BYTE OF PRODUCT
	LD A,[B]	
	RRC A	; RIGHT SHIFT MIDDLE
	X A,[B-]	; BYTE OF PRODUCT
MP168S:	LD A,[B]	
	RRC A	; RIGHT SHIFT LOWER
	X A,[B]	; PRODUCT/MULTIPLIER
	IFNC	; TEST LOWER BIT
	JP MF168T	; OF MULTIPLIER
	RC	; CLEAR CARRY
	LD B,#0	; LOWER BYTE OF
	LD A,[B]	; MULTIPLICAND TO ACC
	LD B,#3	; ADD LOWER BYTE OF
	ADC A,[B]	; MULTIPLICAND TO
	X A,[B]	; MIDDLE BYTE OF PROD
	LD B,#1	; UPPER BYTE OF
	LD A,[B]	; MULTIPLICAND TO ACC
	LD B,#4	; ADD UPPER BYTE OF ICAND
	ADC A,[B]	; TO UPPER BYTE OF PROD
	DRSZ CNTR	; DECREMENT CNTR AND JUMP
	JP M168LP	; BACK TO LOOP; CNTR
		; CANNOT EQUAL ZERO
MP168T:	LD B,#4	; HIGH ORDER PRODUCT
	LD A,[B]	; BYTE TO ACC
	DRSZ CNTR	; DECREMENT AND TEST IF
	JP M168LP	; CNTR EQUAL TO ZERO
	RET	; RETURN FROM SUBROUTINE

MPY816—(OR MPY824, MPY832) 8 BY 16 (OR 24, 32) MULTIPLY SUBROUTINE

MINIMUM CODE, MINIMUM RAM

22 BYTES

589 (OR 1065, 1669) INSTR. CYCLES AVERAGE

597 (OR 1077, 1685) INSTR. CYCLES MAXIMUM

EXTENDABLE ROUTINE FOR MPY8XX BY CHANGING

PARAMETERS, WITH NUMBER OF BYTES (22)

REMAINING A CONSTANT.

MULTIPLICAND IN [0] (ICAND)

MULTIPLIER IN [2,1] FOR 16 BIT (IER)

OR [3,2,1] FOR 24 BIT

OR [4,3,2,1] FOR 32 BIT

PRODUCT IN [3,2,1] FOR 16 BIT (PROD)

OR [4,3,2,1] FOR 24 BIT

OR [5,4,3,2,1] FOR 32 BIT

```

MPY816:  LD          CNTR,#17          ; LD CNTR WITH LENGTH OF
;          ; MULTIPLIER FIELD + 1
;          ; #17 FOR MPY816 16 BIT
;          ; (#25 FOR MPY824 24 BIT)
;          ; (#33 FOR MPY832 32 BIT)

          RC
          LD          B,#3           ; #3 FOR MPY816
;          ; (#4 FOR MPY824)
;          ; (#5 FOR MPY832)
          LD          [B-],#0        ; CLEAR UPPER PRODUCT
M8XXLP:  LD          A,[B]           ; FIVE INSTRUCTION
M8XXL:   RRC          A             ; PROGRAM LOOP TO
          X           A,[B-]        ; RIGHT SHIFT
          IFBNE      #0            ; PRODUCT/MULTIPLIER
          JP          M8XXLP        ; LOOP JUMP BACK
          CLR        A             ; CLR ACC AND TEST LOW
          IFNC       #0            ; ORDER MULTIPLIER BIT
          JP          M8XXT        ; JP IF LOW ORDER BIT = 0
          RC
          LD          B,#0
          LD          A,[B]
M8XXT:   LD          B,#3           ; MULTIPLICAND TO ACC
;          ; #3 FOR MPY816
;          ; (#4 FOR MPY824)
;          ; (#5 FOR MPY832)
          ADC        A,[B]         ; ADD MULTIPLICAND TO
;          ; UPPER BYTE OF PRODUCT
          DRSZ       CNTR          ; DECREMENT AND TEST
          JP          M8XXL        ; CNTR FOR ZERO
          RET          ; RETURN FROM SUBROUTINE

```

MPY248—FAST 24 BY 8 MULTIPLICATION SUBROUTINE

47 BYTES

289 INSTRUCTION CYCLES AVERAGE

333 INSTRUCTION CYCLES MAXIMUM

MULTIPLICAND IN [2,1,0] (ICAND)

MULTIPLIER IN [3] (IER)

PRODUCT IN [6,5,4,3] (PROD)

```

MPY248: LD      CNTR,#9      ; LD CNTR WITH LENGTH OF
        RC          ; MULTIPLIER FIELD + 1
        LD      B,#6
        LD      [B-],#0    ; CLEAR THREE
        LD      [B-],#0    ; UPPER BYTES
        LD      [B-],#0    ; OF PRODUCT
MP248LP: JP      MP248S    ; JUMP TO START
        RRC     A          ; RIGHT SHIFT HIGH
        X      A,[B-]     ; ORDER PRODUCT BYTE
        LD      A,[B]
        RRC     A          ; RIGHT SHIFT NEXT LOWER
        X      A,[B-]     ; ORDER PRODUCT BYTE
        LD      A,[B]
        RRC     A          ; RIGHT SHIFT NEXT LOWER
        X      A,[B-]     ; ORDER PRODUCT BYTE
MP248S: LD      A,[B]
        RRC     A          ; RIGHT SHIFT LOW ORDER
        X      A,[B]     ; PRODUCT/MULTIPLIER
        IFNC    MP248T    ; TEST LOW ORDER
        JP      MP248T    ; MULTIPLIER BIT
        RC
        LD      B,#0      ; LOAD ACC WITH LOW ORDER
        LD      A,[B]     ; MULTIPLICAND BYTE
        LD      B,#4      ; ADD LOW ORDER ICAND
        ADC     A,[B]     ; BYTE TO NEXT TO LOW
        X      A,[B]     ; ORDER PRODUCT BYTE
        LD      B,#1      ; LOAD ACC WITH MIDDLE
        LD      A,[B]     ; MULTIPLICAND BYTE
        LD      B,#5      ; ADD MIDDLE ICAND BYTE
        ADC     A,[B]     ; TO NEXT TO HIGH ORDER
        X      A,[B]     ; MULTIPLICAND BYTE
        LD      B,#2      ; LOAD ACC WITH HIGH ORDER
        LD      A,[B]     ; MULTIPLICAND BYTE
        LD      B,#6      ; ADD HIGH ORDER ICAND BYTE
        ADC     A,[B]     ; TO HIGH ORDER PROD BYTE
        DRSZ    CNTR     ; DECREMENT CNTR AND JUMP
        JP      M248LP    ; BACK TO LOOP; CNTR
        ; CANNOT EQUAL ZERO
MP248T: LD      B,#6      ; HIGH ORDER PRODUCT
        LD      A,[B]     ; BYTE TO ACC
        DRSZ    CNTR     ; DECREMENT AND TEST
        JMP     M248LP    ; CNTR FOR ZERO
        RET          ; RETURN FROM SUBROUTINE

```

MX1616—FAST 16 BY 16 MULTIPLICATION SUBROUTINE

39 BYTES
 498 INSTRUCTION CYCLES AVERAGE
 546 INSTRUCTION CYCLES AVERAGE

MULTIPLICAND IN [1,0] (ICAND)
 MULTIPLIER IN [3,2] (IER)
 PRODUCT IN [5,4,3,2] (PROD)

```

MX1616:  LD      CNTR,#17      ; LD CNTR WITH LENGTH OF
          RC              ; MULTIPLIER FIELD + 1
          LD      B,#5
          LD      [B-],#0   ; CLEAR UPPER TWO
          LD      [B-],#0   ; PRODUCT BYTES
          JP      MXSTRT    ; JUMP TO START
MX1616L: RRC      A        ; RIGHT SHIFT
          X      A,[B-]    ; UPPER PRODUCT BYTE
          LD      A,[B]
          RRC      A        ; RIGHT SHIFT NEXT LOWER
          X      A,[B-]    ; PRODUCT BYTE
MXSTRT:  LD      A,[B]
          RRC      A        ; RIGHT SHIFT PRODUCT
          X      A,[B-]    ; UPPER MULTIPLIER BYTE
          LD      A,[B]
          RRC      A        ; RIGHT SHIFT PRODUCT
          X      A,[B]    ; LOWER MULTIPLIER BYTE
          IFNC
          JP      MX1616T  ; TEST LOW ORDER
          RC              ; MULTIPLIER BIT
          LD      B,#0     ; LOAD ACC WITH LOWER
          LD      A,[B]    ; MULTIPLICAND BYTE
          LD      B,#4     ; ADD LOWER ICAND BYTE
          ADC     A,[B]    ; TO NEXT TO HIGH
          X      A,[B]    ; ORDER PRODUCT BYTE
          LD      B,#1     ; LOAD ACC WITH UPPER
          LD      A,[B]    ; MULTIPLICAND BYTE
          LD      B,#5     ; ADD UPPER ICAND BYTE TO
          ADC     A,[B]    ; HIGH ORDER PRODUCT
          DRSZ    CNTR    ; DECREMENT CNTR AND JUMP
          JP      MX1616L  ; BACK TO LOOP; CNTR
          ; CANNOT EQUAL ZERO
MX1616T: LD      B,#5     ; HIGH ORDER PRODUCT
          LD      A,[B]    ; BYTE TO ACC
          DRSZ    CNTR    ; DECREMENT AND TEST
          JP      MX1616L  ; CNTR FOR ZERO
          RET             ; RETURN FROM SUBROUTINE
  
```

MP1616—16 BY 16 MULTIPLICATION SUBROUTINE

MINIMUM CODE
 29 BYTES
 759 INSTRUCTION CYCLES AVERAGE
 807 INSTRUCTION CYCLES MAXIMUM]
 MULTIPLICAND IN [1,0] (ICAND)
 MULTIPLIER IN [3,2] (IER)
 PRODUCT IN [5,4,3,2] (PROD)

```

MP1616: LD      CNTR,#17      ; LD CNTR WITH LENGTH OF
RC      ; MULTIPLIER FIELD + 1
LD      B,#5
LD      [B-],#0           ; CLEAR UPPER TWO
LD      [B-],#0           ; PRODUCT BYTES
M1616X: LD      A,[B]      ; FIVE INSTRUCTION
M1616L: RRC      A         ; PROGRAM LOOP TO
X       A,[B-]           ; RIGHT SHIFT
IFBNE  #1               ; PRODUCT/MULTIPLIER.
JP     M1616X           ; LOOP JUMP BACK
CLR    A                ; CLEAR ACC
IFNC   ; TEST LOW ORDER
JP     M1616T           ; MULTIPLIER BIT
RC
LD      B,#0            ; LOAD ACC WITH LOWER
LD      A,[B]           ; MULTIPLICAND BYTE
LD      B,#4            ; ADD LOWER ICAND BYTE
ADC    A,[B]            ; TO NEXT TO LOW
X      A,[B]            ; ORDER PRODUCT BYTE
LD      B,#1            ; LOAD ACC WITH UPPER
LD      A,[B]           ; MULTIPLICAND BYTE
M1616T: LD      B,#5     ; ADD UPPER ICAND BYTE TO
ADC    A,[B]           ; HIGH ORDER PRODUCT
DRSZ  CNTR             ; DECREMENT AND TEST
JP     M1616L          ; CNTR EQUAL TO ZERO
RET    ; RETURN FROM SUBROUTINE

```

MY1616 (OR MY1624, MY1632)—16 BY 16 (OR 24, 32) MULTIPLY SUBROUTINE

MINIMUM CODE, MINIMUM RAM

28 BYTES

861 (OR 1473, 2213) INST. CYCLES AVERAGE

1029 (OR 1725, 1473) INST. CYCLES MAXIMUM

EXTENDABLE ROUTINE FOR MY16XX BY CHANGING

PARAMETERS, WITH NUMBER OF BYTES (28)

REMAINING A CONSTANT

MULTIPLICAND IN [1,0] (ICAND)

MULTIPLIER IN [3,2] FOR 16 BIT (IER)

OR [4,3,2] FOR 24 BIT

OR [5,4,3,2] FOR 32 BIT

PRODUCT IN [5,4,3,2] FOR 16 BIT (PROD)

OR [6,5,4,3,2] FOR 24 BIT

OR [7,6,5,4,3,2] FOR 32 BIT

```

MY1616: LD      CNTR,#17      ; LD CNTR WITH LENGTH OF
          ; MULTIPLIER FIELD + 1
          ; #17 FOR MY1616
          ; (#25 FOR MY1624)
          ; (#33 FOR MY1632)
LD      B,#5              ; #5 FOR MY1616
          ; (#6 FOR MY1624)
          ; (#7 FOR MY1632)
LD      [B-],#0          ; CLEAR UPPER TWO
LD      [B-],#0          ; PRODUCT BYTES
RC
MY16XS: LD      A,[B]      ; FIVE INSTRUCTION
RRC     A                ; PROGRAM LOOP TO
X       A,[B-]          ; RIGHT SHIFT
IFBNE  #1              ; PRODUCT/MULTIPLIER
JP     M16XS           ; LOOP JUMP BACK
IFNC   #1              ; TEST LOW ORDER
JP     MY16XT         ; MULTIPLIER BIT
RC
LD      B,#4            ; #4 FOR MY1616
          ; (#5 FOR MY1624)
          ; (#6 FOR MY1632)
          ; LOAD ACC WITH
MY16XL: LD      X,#0      ;
LD      A,[X+]          ; MULTIPLICAND BYTES
ADC     A,[B]           ; ADD MULTIPLICAND TO
X       A,[B+]          ; HI TWO PROD. BYTES
IFBNE  #2              ; LOOP BACK FOR SECOND
JP     MY16XL         ; MULTIPLICAND BYTE
MY16XT: LD      B,#5      ; #5 FOR MY1616
          ; (#6 FOR MY1624)
          ; (#7 FOR MY1632)
DRSZ   CNTR            ; DECREMENT AND TEST
JP     MY16XS         ; CNTR EQUAL TO ZERO
RET    ; RETURN FROM INTERRUPT
;

```

MY2448—MINIMAL GENERAL MULTIPLICATION SUBROUTINE (28 BYTES)

ANY NUMBER OF BYTES IN MULTIPLICAND
AND MULTIPLIER

FIRST EXAMPLE: (MY2448)
24 BY 48 MULTIPLICATION SUBROUTINE
--28 BYTES
--MINIMAL CODE, MINIMAL RAM
--4713 INSTRUCTION CYCLES AVERAGE
--5457 INSTRUCTION CYCLES MAXIMUM
MULTIPLICAND IN [2,1,0] (ICAND)
MULTIPLIER IN [8,7,6,5,4,3] (IER)
PRODUCT IN [11,10,9,8,7,6,5,4,3] (PROD)

SECOND EXAMPLE: (MY4824)
48 BY 24 MULTIPLICATION SUBROUTINE
--28 BYTES
--MINIMAL CODE, NON MINIMAL RAM
--2751 INSTRUCTION CYCLES AVERAGE
--3483 INSTRUCTION CYCLES MAXIMUM
MULTIPLICAND IN [5,4,3,2,1,0] (ICAND)
MULTIPLIER IN [8,7,6] (IER)
PRODUCT IN [14,13,12,11,10,9,8,7,6] (PROD)

```

MY2448: ; (OR MY4824)
        LD      CNTR, #49 ; LD CNTR WITH LENGTH OF
                        ; MULTIPLIER FIELD + 1
                        ; #49 FOR MY2448
                        ; (#25 FOR MY4824)
        LD      B, #11 ; TOP OF PROD TO B PTR
                        ; #11 FOR MY2448
                        ; (#14 FOR MY4824)
CLRLUP: LD      [B-], #0 ; CLR UNTIL TOP OF IER
        IFBNE  #8 ; #8 FOR BOTH MY2448
        JP      CLRLUP ; AND MY4824
        RC      ; INITIALIZE CARRY
SHFTLP: LD      A, [B] ; RIGHT SHIFT PRODUCT
        ADC    A, [B] ; AND MULTIPLIER
        X      A, [B-] ; UNTIL TOP OF ICAND
        IFBNE  #2 ; #2 FOR MY2448
        JP      SHFTLP ; (#5 FOR MY4824)
        IFNC   ; TEST LOW ORDER
        JP      MYTEST ; MULTIPLIER BIT
        LD      B, #9 ; TOP OF IER + 1 TO B PTR
        LD      X, #0 ; START OF ICAND TO X PTR
        RC
ADDLUP: LD      A, [X+] ; ADD MULTIPLICAND TO TOP
        ADC    A, [B] ; OF PRODUCT ABOVE
        X      A, [B+] ; MULTIPLIER UNTIL TOP
        IFBNE  #12 ; OF PRODUCT + 1
        JP      ADDLUP ; #12 FOR MY2448
                        ; (#15 FOR MY4824)
MYTEST: LD      B, #11 ; TOP OF PROD TO B PTR
                        ; #11 FOR MY2448
                        ; (#14 FOR MY4824)
        DRSZ   CNTR ; DECREMENT AND TEST
        JP      SHFTLP ; CNTR FOR ZERO
        RET    ; RETURN FROM SUBROUTINE

```


3.0 DIVISION

The COP 800 divisions are all based on shifting the dividend left up into a test field equal in length to the number of bytes in the divisor. The divisor is resident immediately above this test field. After each shift cycle of the dividend into the test field, a trial subtraction is made of the test field minus the divisor. If the divisor is found equal to or less than the contents of the test field, then the divisor is subtracted from the test field and a 1's quotient digit is recorded by setting the low order bit of the dividend field. The dividend and test field left shift cycle is then repeated. The number of left shift cycles is equal to the number of bit positions in the dividend. The quotient from the division is formed in the dividend field, while the remainder from the division is resident in the test field.

Note that an M byte dividend divided by an N byte divisor will result in an M byte quotient and an N byte remainder.

These division algorithms will use $M + 2N + 1$ bytes of RAM memory space, since the test field is equal to the length of the divisor. The one extra byte is necessary for the shift counter CNTR.

In special cases where the dividend has an upper bound and the divisor has a lower bound, the upper bytes of the dividend may be used as the test field. One example is shown (DV2815), where a 28 bit dividend is divided by a 15-bit divisor. The dividend is less than $2^{**}28$ (upper nibble of high order byte is zero), while the divisor is greater than $2^{**}12$ (4096) and less than $2^{**}15$ (32768). In this case, the upper limit for the quotient is $2^{**}28/2^{**}12$, which indicates a 16-bit quotient ($2^{**}16$) and a 15-bit remainder. Consequently, the upper two bytes of the dividend may be used as the test field for the remainder, since the divisor is greater than the test field (upper two bytes of the 28-bit dividend) initially.

The minimal code (40 byte) general division subroutine is shown with the example DV3224, which divides a 32 bit dividend by a 24 bit divisor.

DIV88	— 8 by 8 Division Subroutine	FDV168	— Fast 16 by 8 Division Subroutine
	— 24 Bytes		— 35 Bytes
	— 201 Instruction Cycles Average		— 481 Instruction Cycles Average
	— 209 Instruction Cycles Maximum		— 490 Instruction Cycles Maximum
	Minimum code	FDV248	— Fast 24 by 8 Division Subroutine
DV88	— Fast 8 by 8 Division Subroutine		— 38 Bytes
	— 28 Bytes		— 813 Instruction Cycles Average
	— 194 Instruction Cycles Average		— 826 Instruction Cycles Maximum
	— 202 Instruction Cycles Maximum	FDV328	— Fast 32 by 8 Division Subroutine
FDV88	— Very Fast 8 by 8 Division Subroutine		— 42 Bytes
	— 131 Bytes		— 1209 Instruction Cycles Average
	— 146 Instruction Cycles Average		— 1226 Instructions Maximum
	— 159 Instruction Cycles Maximum		
DIV168 (or DIV248, DIV328)	— 16 (or 24, 32) by 8 Division Subroutine		Divide by 16 Subroutines:
	— 26 Bytes	DV1616	— 16 by 16 Division Subroutine
	— 649 (or 1161, 1801) Instruction Cycles Average		— 34 Bytes
	— 681 (or 1209, 1865) Instruction Cycles Maximum		— 979 Instruction Cycles Average
	— Minimum Code		— 1067 Instruction Cycles Maximum
	— Extendable Routine for DIVXX8 by Changing Parameters, with Number of Bytes (26) Remaining a Constant		— Minimum Code
		DV2416 (or DV3216)	— 24 (or 32) by 16 Division Subroutine
			— 39 Bytes
			— 1694 (or 2410) Inst. Cycles Average
			— 1886 (or 2766) Inst. Cycles Maximum
			— Minimum code
			— Extendable Routine for DVXX16 by Changing Parameters, with Number of Bytes (39) Remaining a Constant
		DX1616	— Fast 16 by 16 Division Subroutine
			— 53 Bytes
			— 638 Instruction Cycles Average
			— 678 Instruction Cycles Maximum
		DV2815	— Fast 28 by 15 Division Subroutine, Where the Dividend is Less Than $2^{**}28$ and the Divisor is Greater than $2^{**}12$ (4096) and Less than $2^{**}15$ (32768)
			— 43 Bytes
			— 640 Instruction Cycles Average
			— 696 Instruction Cycles Maximum
		DX3216	— Fast 32 by 16 Division Subroutine
			— 70 Bytes
			— 1511 Instruction Cycles Average
			— 1591 Instruction Cycles Maximum
			Minimal General Division Subroutine for any Number of Bytes in Dividend and Divisor
			— 40 Bytes
			— Minimal Code
			— DV3224 Used as Example, with 3879 Instruction Cycles Average and 4535 Instruction Cycles Maximum

DIV88—8 BY 8 DIVISION SUBROUTINE

MINIMUM CODE

24 BYTES

201 INSTRUCTION CYCLES AVERAGE

209 INSTRUCTION CYCLES MAXIMUM

DIVIDEND IN [0] (DD)
 DIVISOR IN [2] (DR)
 QUOTIENT IN [0] (QUOT)
 REMAINDER IN [1] (TEST FIELD)

```

DIV88:  LD      CNTR,#8      ; LOAD CNTR WITH LENGTH
        LD      B,#1      ; OF DIVIDEND FIELD
        LD      [B],#0    ; CLEAR TEST FIELD
DIV88S: RC
        LD      B,#0
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT TEST FIELD
        X      A,[B]
        LD      A,[B+]    ; TEST FIELD TO ACC
        SC     ; TEST SUBTRACT DIVISOR
        SUBC   A,[B]      ; FROM TEST FIELD
        IFNC   ; TEST IF BORROW
        JP     DIV88B     ; FROM SUBTRACTION
        LD      B,#1      ; SUBTRACTION RESULT
        X      A,[B-]    ; TO TEST FIELD
        SBIT   O,[B]     ; SET QUOTIENT BIT
DIV88B: DRSZ   CNTR      ; DECREMENT AND TEST
        JP     DIV88S     ; CNTR FOR ZERO
        RET    ; RETURN FROM SUBROUTINE
  
```

DV88—FAST 8 BY 8 DIVISION SUBROUTINE

```

28 BYTES
194 INSTRUCTION CYCLES AVERAGE
202 INSTRUCTION CYCLES MAXIMUM
DIVIDEND    IN [0]      (DD)
DIVISOR IN [2]      (DR)
QUOTIENT IN [0]     (QUOT)
REMAINDER IN [1]    (TEST FIELD)

DV88:  LD      CNTR,#8      ; LOAD CNTR WITH LENGTH
       LD      B,#1       ; OF DIVIDEND FIELD
       LD      [B-],#0    ; CLEAR TEST FIELD
       RC

DV88S: LD      A,[B]
       ADC     A,[B]      ; LEFT SHIFT DIVIDEND
       X      A,[B+]
       LD      A,[B]
       ADC     A,[B]      ; LEFT SHIFT TEST FIELD
       X      A,[B]
       LD      A,[B+]    ; TEST FIELD TO ACC
       SC     ; TEST SUBTRACT DIVISOR
       SUBC   A,[B]      ; FROM TEST FIELD
       IFNC   ; TEST IF BORROW
       JP     DV88B      ; FROM SUBTRACTION
       LD      B,#1      ; SUBTRACTION RESULT
       X      A,[B-]    ; TO TEST FIELD
       SBIT   0,[B]     ; SET QUOTIENT BIT
       RC

DRSZ   CNTR      ; DECREMENT AND TEST
JP     DV88S    ; CNTR FOR ZERO
RET    ; RETURN FROM SUBROUTINE

DV88B: LD      B,#0
DRSZ   CNTR      ; DECREMENT AND TEST
JP     DV88S    ; CNTR FOR ZERO
RET    ; RETURN FROM SUBROUTINE

```

FDV88—VERY FAST 8 BY 8 DIVISION SUBROUTINE

```

131 BYTES
146 INSTRUCTION CYCLES AVERAGE
159 INSTRUCTION CYCLES MAXIMUM
DIVIDEND IN [0]          (DD)
DIVISOR IN [2]          (DR)
QUOTIENT IN [0]         (QUOT)
REMAINDER IN [1]       (TEST FIELD)

FDV88:  LD      B,#1
        LD      [B-],#0      ; CLEAR TEST FIELD
        RC
        LD      A,[B]
        ADC     A,[B]        ; LEFT SHIFT DIVIDEND
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]        ; LEFT SHIFT TEST FIELD
        X      A,[B]
        LD      A,[B+]      ; TEST FIELD TO ACC
        SC      ; TEST SUBTRACT DIVISOR
        SUBC   A,[B]        ; FROM TEST FIELD
        IFNC   ; TEST IF BORROW
        JP     DVBP1        ; FROM SUBTRACTION
        LD      B,#1        ; SUBTRACTION RESULT
        X      A,[B-]      ; TO TEST FIELD
        SBIT   O,[B]        ; SET QUOTIENT BIT
        RC

DVBP1:  LD      B,#0        ; THIS 16 BYTE SECTION
        LD      A,[B]      ; OF PROGRAM CODE
        ADC     A,[B]      ; CONTAINS
        X      A,[B+]      ; 16 INSTRUCTIONS,
        LD      A,[B]      ; AND REPRESENTS THE
        ADC     A,[B]      ; PROCESSING FOR THE
        X      A,[B]      ; GENERATION OF
        LD      A,[B+]      ; 1 QUOTIENT BIT.
        SC      ;
        SUBC   A,[B]      ; THE PROGRAM CODE
        IFNC   ; EXECUTION TIMES IS 16
        JP     DVBP2      ; INSTRUCTION CYCLES
        LD      B,#1      ; FOR A 0'S QUOTIENT BIT
        X      A,[B-]      ; AND 19 INSTRUCTION
        SBIT   O,[B]      ; CYCLES FOR A 1'S
        RC      ; QUOTIENT BIT.
;
; DVBP2:  LD      B,#0      ; REPEAT THE ABOVE
;
; DVBP3:  ;
;
; DVBP4:  ;SECTION OF CODE FIVE
;
; DVBP5:  ;MORE TIMES FOR A
;
; DVBP6:  ;TOTAL OF SIX TIMES
;
;
;
; DVBP7:  LD      B,#0
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT TEST FIELD
        X      A,[B]
        LD      A,[B+]      ; TEST FIELD TO ACC
        SC      ; TEST SUBTRACT DIVISOR
        SUBC   A,[B]      ; FROM TEST FIELD
        IFNC   ; TEST BORROW FROM SUBC
        RET    ; RETURN FROM SUBROUTINE
        LD      B,#1      ; SUBTRACTION RESULT
        X      A,[B-]      ; TO TEST FIELD
        SBIT   O,[B]      ; SET QUOTIENT BIT
        RET    ; RETURN FROM SUBROUTINE

```

DIV168—16 (OR 24, 32) BY 8 DIVISION SUBROUTINE

MINIMUM CODE
 26 BYTES
 649 (or 1161,1801) INST. CYCLES AVERAGE
 681 (or 1209,1865) INST. CYCLES MAXIMUM
 EXTENDABLE ROUTINE FOR DIVXX8 BY CHANGING
 PARAMETERS, WITH NUMBER OF BYTES (26)
 REMAINING A CONSTANT

DIVIDEND IN [1,0] FOR 16 BIT (DD)
 OR [2,1,0] FOR 24 BIT
 OR [3,2,1,0] FOR 32 BIT

DIVISOR IN [3] FOR 16 BIT (DR)
 OR [4] FOR 24 BIT
 OR [5] FOR 32 BIT

QUOTIENT IN [1,0] FOR 16 BIT (QUOT)
 OR [2,1,0] FOR 24 BIT
 OR [3,2,1,0] FOR 32 BIT

REMAINDER IN [2] FOR 16 BIT (TEST FIELD)
 OR [3] FOR 24 BIT
 OR [4] FOR 32 BIT

```

DIV168: LD CNTR,#16 ; LOAD CNTR WITH LENGTH
        ; OF DIVIDEND FIELD
        ; #16 FOR DIV168
        ; (#24 FOR DIV248)
        ; (#32 FOR DIV328)
        LD B,#2 ; (#3 FOR DIV168)
        ; (#3 FOR DIV248)
        ; (#4 FOR DIV328)
        LD [B],#0 ; CLEAR TEST FIELD
DVXX8L: RC
        LD B,#0
DXX8LP: LD A,[B] ; LEFT SHIFT DIVIDEND
        ADC A,[B] ; AND TEST FIELD
        X A,[B+]
        IFBNE #3 ; #3 FOR DIV168
        JP DXX8LP ; (#4 FOR DIV248)
        ; (#5 FOR DIV328)
        LD A,[B-] ; DIVISOR TO ACCUMULATOR
        IFC ; TEST IF BIT SHIFTED OUT
        JP DVXX8S ; OF TEST FIELD***
        IFGT A,[B] ; TEST DIVISOR GREATER
        JP DVXX8T ; THAN REMAINDER
        SC ;
DVXX8S: X A,[B] ; REMAINDER TO ACC
        SUBC A,[B] ; SUBTRACT DIVISOR
        X A,[B] ; FROM REMAINDER
        LD B,#0
DVXX8T: SBIT O,[B] ; SET QUOTIENT BIT
        DRSZ CNTR ; DECREMENT AND TEST
        JP DVXX8L ; CNTR FOR ZERO
        RET ; RETURN FROM SUBROUTINE

```

;

;

```

; *** SPECIAL CASE FOR DIVISION WHERE NUMBER OF BYTES
; IN DIVIDEND IS GREATER THAN NUMBER OF BYTES IN DIVISOR, AND
; DIVISOR CONTAINS A HIGH ORDER 1'S BIT. THE SHIFTED DIVIDEND
; MAY CONTAIN A HIGH ORDER 1'S BIT IN THE TEST FIELD AND
; YET BE SMALLER THAN THE DIVISOR SO THAT NO SUBTRACTION
; OCCURS. IN THIS CASE A 1'S BIT WILL BE SHIFTED OUT OF
; THE TEST FIELD AND AN OVERRIDE SUBTRACTION MUST BE PERFORMED

```

FDV168—FAST 16 BY 8 DIVISION SUBROUTINE

35 BYTES
 481 INSTRUCTION CYCLES AVERAGE
 490 INSTRUCTION CYCLES MAXIMUM
 DIVIDEND IN [1,0] (DD)
 DIVISOR IN [3] (DR)
 QUOTIENT IN [1,0] (QUOT)
 REMAINDER IN [2] (TEST FIELD)

```

FDV168: LD      CNTR,#16      ; LOAD CNTR WITH LENGTH
        LD      B,#3        ; OF DIVIDEND FIELD
        LD      [B],#0      ; CLEAR TEST FIELD
FD168S: LD      B,#0
FD168L: RC
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND LO
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND HI
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT TEST FIELD
        X      A,[B]
        LD      A,[B+]     ; TEST FIELD TO ACC
        IFC     ; TEST IF BIT SHIFTED OUT
        JP      FD168B     ; OF TEST FIELD***
        SC      ; TEST SUBTRACT DIVISOR
        SUBC   A,[B]      ; FROM TEST FIELD
        IFNC    ; TEST IF BORROW
        JP      FD168T     ; FROM SUBTRACTION
FD168R: LD      B,#2        ; SUBTRACTION RESULT
        X      A,[B]      ; TO TEST FIELD
        LD      B,#0
        SBIT   0,[B]      ; SET QUOTIENT BIT
        DRSZ   CNTR      ; DECREMENT AND TEST
        JP      FD168L     ; CNTR FOR ZERO
        RET     ; RETURN FROM SUBROUTINE
FD168T: DRSZ   CNTR      ; DECREMENT AND TEST
        JP      FD168S     ; CNTR FOR ZERO
        RET     ; RETURN FROM SUBROUTINE
FD168B: SUBC   A,[B]      ; SUBTRACT DIVISOR FROM
        JP      FD168R     ; TEST FIELD***
  
```

FDV248—FAST 24 BY 8 DIVISION SUBROUTINE

38 BYTES
 813 INSTRUCTION CYCLES AVERAGE
 826 INSTRUCTION CYCLES MAXIMUM
 DIVIDEND IN [2,1,0] (DD)
 DIVISOR IN [4] (DR)
 QUOTIENT IN [2,1,0] (QUOT)
 REMAINDER IN [3] (TEST FIELD)

```

FDV248: LD      CNTR,#24      ; LOAD CNTR WITH LENGTH
        LD      B,#4        ; OF DIVIDEND FIELD
        LD      [B],#0     ; CLEAR TEST FIELD
FD248S: LD      B,#0
FD248L: RC
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND LO
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND MID
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT DIVIDEND HI
        X      A,[B+]
        LD      A,[B]
        ADC     A,[B]      ; LEFT SHIFT TEST FIELD
        X      A,[B]
        LD      A,[B+]
        IFC     ; TEST IF BIT SHIFTED OUT
        JP      FD248B     ; OF TEST FIELD ***
        SC     ; TEST SUBTRACT DIVISOR
        SUBC   A,[B]      ; FROM TEST FIELD
        IFNC   ; TEST IF BORROW
        JP      FD248T     ; FROM SUBTRACTION
FD248R: LD      B,#3        ; SUBTRACTION RESULT
        X      A,[B]      ; TO TEST FIELD
        LD      B,#0
        SBIT   0,[B]      ; SET QUOTIENT BIT
        DRSZ   CNTR       ; DECREMENT AND TEST
        JP      FD248L     ; CNTR FOR ZERO
        RET    ; RETURN FROM SUBROUTINE
FD248T: DRSZ   CNTR       ; DECREMENT AND TEST
        JP      FD248S     ; CNTR FOR ZERO
        RET    ; RETURN FROM SUBROUTINE
FD248B: SUBC   A,[B]      ; SUBTRACT DIVISOR FROM
        JP      FD248R     ; TEST FIELD ***
  
```

DV1616—16 (OR 24, 32) BY 16 DIVISION SUBROUTINE

MINIMUM CODE
 34 BYTES
 979 (OR 1655,2459) INSTRUCTION CYCLES AVERAGE
 1067 (OR 1787,2635) INSTRUCTION CYCLES MAXIMUM
 DIVIDEND IN [1,0] (DD)
 DIVISOR IN [5,4] (DR)
 QUOTIENT IN [1,0] (QUOT)
 REMAINDER IN [3,2] (TEST FIELD)

```

DV1616: LD      CNTR,#16      ; LOAD CNTR WITH LENGTH
          ;              OF DIVIDEND FIELD
          LD      B,#3
          LD      [B-],#0    ; CLEAR
          LD      [B],#0     ; TEST FIELD
DV616S: RC
          LD      X,#2      ; INITIALIZE X POINTER
          LD      B,#0      ; INITIALIZE B POINTER
DV616L: LD      A,[B]       ; LEFT SHIFT DIVIDEND
          ADC     A,[B]      ; AND TEST FIELD
          X       A,[B+]
          IFBNE  #4
          JP     DV616L
          SC      ; RESET BORROW
          LD      A,[X+]     ; TEST FIELD LO TO ACC
          SUBC   A,[B]      ; SUBT DR LO FROM REM LO
          LD      A,[X]     ; TEST FIELD HI TO ACC
          LD      B,#5
          SUBC   A,[B]      ; SUBT DR HI FROM REM HI
          IFNC   ; TEST IF BORROW
          JP     DV616T     ; FROM SUBTRACTION
          X       A,[X-]    ; SUBT RESULT HI TO REM HI
          LD      A,[X]     ; TEST FIELD LO TO ACC
          LD      B,#4
          SUBC   A,[B]      ; SUBT DR LO FROM REM LO
          X       A,[X]     ; RESULT LO TO REM LO
          LD      B,#0
          SBIT   O,[B]      ; SET QUOTIENT BIT
DV616T: DRSZ   CNTR        ; DECREMENT AND TEST
          JP     DV616S     ; CNTR FOR ZERO
          RET      ; RETURN FROM SUBROUTINE

```


DX1616—FAST 16 BY 16 DIVISION SUBROUTINE

53 BYTES

638 INSTRUCTION CYCLES AVERAGE

678 INSTRUCTION CYCLES MAXIMUM

DIVIDEND IN [1,0] (DD)
 DIVISOR IN [5,4] (DR)
 QUOTIENT IN [1,0] (QUOT)
 REMAINDER IN [3,2] (TEST FIELD)

```

DX1616:  LD      CNTR,#16      ; LOAD CNTR WITH LENGTH
         LD      B,#5       ; OF DIVIDEND FIELD
         LD      A,[B]      ; REPLACE DIVISOR WITH
XOR      A,#OFF           ; 1'S COMPLEMENT OF
X        A,[B-]           ; DIVISOR TO ALLOW
LD      A,[B]             ; OPTIONAL ADDITION OF
XOR      A,#OFF           ; DIVISOR'S COMPLEMENT
X        A,[B-]           ; IN MAIN PROG. LOOP
LD      [B-],#0           ; CLEAR
LD      [B],#0            ; TEST FIELD
DX616S:  LD      B,#0
DX616L:  RC
         LD      A,[B]
ADC      A,[B]             ; LEFT SHIFT DIVIDEND LO
X        A,[B+]
LD      A,[B]
ADC      A,[B]             ; LEFT SHIFT DIVIDEND HI
X        A,[B+]
LD      A,[B]
ADC      A,[B]             ; LEFT SHIFT TEST FIELD LO
X        A,[B+]
LD      A,[B]
ADC      A,[B]             ; LEFT SHIFT TEST FIELD HI
X        A,[B+]
SC
LD      A,[B]             ; DIVISORX (DRX) LO TO ACC
LD      B,#2              ; (1'S COMPLEMENT)
ADC      A,[B]             ; ADD REM LO TO DRX LO
LD      B,#5
LD      A,[B]             ; DIVISORX (DRX) HI TO ACC
LD      B,#3              ; (1'S COMPLEMENT)
ADC      A,[B]             ; ADD REM HI TO DRX HI
IFNC
JP      DX616T            ; TEST IF NO CARRY FROM
JP      DX616T            ; 1'S COMPL.ADDITION
X        A,[B+]           ; RESULT TO REM HI
LD      A,[B]             ; DRX LO TO ACCUMULATOR
LD      B,#2
ADC      A,[B]             ; ADD REM LO TO DRX LO
X        A,[B]             ; RESULT TO REM LO
LD      B,#0
SBIT    O,[B]             ; SET QUOTIENT BIT
DRSZ    CNTR              ; DECREMENT AND TEST
JP      DX616L            ; CNTR FOR ZERO
RET
DX616T:  DRSZ    CNTR      ; DECREMENT AND TEST
         JMP     DX616S    ; CNTR FOR ZERO
         RET
  
```

DV2815—FAST 28 BY 15 DIVISION SUBROUTINE

WHERE THE DIVIDEND IS LESS THAN 2**28
 AND THE DIVISOR IS GREATER THAN 2**12 (4096) AND LESS THAN 2**15 (32768)
 43 BYTES
 640 INSTRUCTION CYCLES AVERAGE
 696 INSTRUCTION CYCLES MAXIMUM
 DIVIDEND IN [3,2,1,0] (DD)
 DIVISOR IN [5,4] (DR)
 QUOTIENT IN [1,0] (QUOT)
 REMAINDER IN [3,2] (TEST FIELD)

```
DV2815: LD      CNTR,#16      ; LOAD CNTR WITH LENGTH OF QUOTIENT FIELD
D2815S: LD      B,#0
D2815L: RC
LD      A,[B]
ADC     A,[B]      ; LEFT SHIFT LOWER
X       A,[B+]    ; BYTE OF DIVIDEND
LD      A,[B]
ADC     A,[B]      ; LEFT SHIFT NEXT HIGHER
X       A,[B+]    ; BYTE OF DIVIDEND
LD      A,[B]
ADC     A,[B]      ; LEFT SHIFT NEXT HIGHER
X       A,[B+]    ; BYTE OF DIVIDEND
LD      A,[B]
ADC     A,[B]      ; LEFT SHIFT UPPER
X       A,[B-]    ; BYTE OF DIVIDEND
***
```

NOTE THAT WITH A 16 BIT DIVISOR (DIV 2816) SUBROUTINE, A TEST FOR A HIGH ORDER BIT SHIFTED OUT OF THE TEST FIELD WOULD BE NECESSARY AT THIS POINT.
 IFC

```
JP      SUBTRMD      ; SUBTRACT REM MINUS DR
```

THE PRESENCE OF THIS CARRY WOULD REQUIRE THAT THE DIVISOR BE SUBTRACTED FROM THE REMAINDER AS SHOWN WITH THE DIV168*** SUBROUTINE.

```
LD      A,[B]      ; REM LOWER BYTE TO ACC
SC      ; TEST SUBTRACT LOWER
LD      B,#4      ; BYTE OF DR FROM
SUBC   A,[B]      ; LOWER BYTE OF REM
LD      B,#3      ; TEST SUBTRACT UPPER
LD      A,[B]      ; BYTE OF DIVISOR
LD      B,#5      ; FROM UPPER BYTE
SUBC   A,[B]      ; OF REMAINDER
IFNC   ; TEST IF BORROW
JP      D2815T    ; FROM SUBTRACTION
LD      B,#3      ; UPPER BYTE OF RESULT
X       A,[B+]    ; TO UPPER BYTE OF REM
LD      A,[B]      ; DR LOWER BYTE TO ACC
LD      B,#2      ; SUBTRACT LOWER BYTE
X       A,[B]      ; OF DIVISOR FROM
SUBC   A,[B]      ; LOWER BYTE OF
X       A,[B]      ; REMAINDER
LD      B,#0
SBIT   0,[B]      ; SET QUOTIENT BIT
DRSZ   CNTR      ; DECREMENT AND TEST
JMP    D2815L    ; CNTR FOR ZERO
RET    ; RETURN FROM SUBROUTINE
D2815T: DRSZ   CNTR      ; DECREMENT AND TEST
JMP    D2815S    ; CNTR FOR ZERO
RET    ; RETURN FROM SUBROUTINE
```

DX3216—FAST 32 BY 16 DIVISION SUBROUTINE

```

70 BYTES
1510 INSTRUCTION CYCLES AVERAGE
1590 INSTRUCTION CYCLES MAXIMUM
DIVIDEND IN [3,2,1,0]      (DD)
DIVISOR IN [7,6]          (DR)
QUOTIENT IN [3,2,1,0]    (QUOT)
REMAINDER IN [5,4]      (TEST FIELD)

DX3216:  LD      CNTR,#32      ; LOAD CNTR WITH LENGTH
         LD      B,#7        ; OF DIVIDEND FIELD
         LD      A,[B]       ; REPLACE DIVISOR WITH
XOR      A,#OFF            ; 1'S COMPLEMENT OF
         X       A,[B-]      ; DIVISOR TO ALLOW
         LD      A,[B]       ; OPTIONAL ADDITION OF
XOR      A,#OFF            ; DIVISOR'S COMPLEMENT
         X       A,[B-]      ; IN MAIN PROG. LOOP
         LD      [B-],#0     ; CLEAR
         LD      [B],#0     ; TEST FIELD

DX326S:  LD      B,#0

DX326L:  RC
         LD      A,[B]
         ADC     A,[B]       ; LEFT SHIFT DIVIDEND LO
         X       A,[B+]
         LD      A,[B]
         ADC     A,[B]       ; LEFT SHIFT NEXT HIGHER
         X       A,[B+]      ; DIVIDEND BYTE
         LD      A,[B]
         ADC     A,[B+]      ; LEFT SHIFT NEXT HIGHER
         X       A,[B+]      ; DIVIDEND BYTE
         LD      A,[B]
         ADC     A,[B]       ; LEFT SHIFT DIVIDEND HI
         X       A,[B+]
         LD      A,[B]
         ADC     A,[B]       ; LEFT SHIFT TST FIELD LO
         X       A,[B+]
         LD      A,[B]
         ADC     A,[B]       ; LEFT SHIFT TST FIELD HI
         X       A,[B+]
         IFC
         JP      DX326B     ; **TEST IF BIT SHIFTED
         SC
         LD      A,[B]       ; DVSORX (DRX) LO TO ACC
         LD      B,#4        ; (1'S COMPLEMENT)
         ADC     A,[B]       ; ADD REM LO TO DRX LO
         LD      B,#7
         LD      A,[B]       ; DVSORX (DRX) HI TO ACC
         LD      B,#5        ; (1'S COMPLEMENT)
         ADC     A,[B]       ; ADD REM HI TO DRX HI
         IFNC
         JP      DX326T     ; TEST IF NO CARRY FROM
         X       A,[B+]      ; 1'S COMPL. ADDITION
         LD      A,[B]       ; RESULT TO REM NI
         LD      B,#4        ; DRX LO TO ACCUMULATOR

DX326R:  ADC     A,[B]       ; ADD REM LO TO DRX LO
         X       A,[B]       ; ** ADD REM HI TO DRX HI
         ; RESULT TO REM LO
         ; ** RESULT TO REM HI

LD      B,#0
         SBIT   0,[B]       ; SET QUOTIENT BIT
         DRSZ   CNTR        ; DECREMENT AND TEST
         JMP    DX326L     ; CNTR FOR ZERO
         RET
         ; RETURN FROM SUBROUTINE

DX326T:  DRSZ   CNTR        ; DECREMENT AND TEST
         JMP    DX326S     ; CNTR FOR ZERO
         RET
         ; RETURN FROM SUBROUTINE

DX326B:  LD      A,[B]       ; ** REM LO TO ACC
         LD      B,#6        ; ** B PTR TO DRX LO
         ADC     A,[B]       ; ** ADD DRX LO TO REM LO
         X       A,[B]       ; ** RESULT TO REM LO
         LD      B,#7        ; **
         LD      A,[B]       ; ** DRX HI TO ACC
         LD      B,#5        ; ** B PTR TO REM HI
         JP      DX326R     ; **

```

** THESE INSTRUCTIONS UNNECESSARY IF DIVISOR
LESS THAN 2**15 (DX3215 SUBROUTINE)

MINIMAL GENERAL DIVISION SUBROUTINE (40 BYTES)

ANY NUMBER OF BYTES IN DIVIDEND AND DIVISOR

DV3224 SERVES AS EXAMPLE

32 BY 24 DIVISION SUBROUTINE

--40 BYTES

--MINIMAL CODE

--3879 INSTRUCTION CYCLES AVERAGE

--4535 INSTRUCTION CYCLES MAXIMUM

DIVIDEND IN [3,2,1,0] (DD)
 DIVISOR IN [9,8,7] (DR)
 QUOTIENT IN [3,2,1,0] (QUOT)
 REMAINDER IN [6,5,4] (TEST FIELD)

```

DV3224:  LD      CNTR,#32      ; LOAD CNTR WITH LENGTH
          LD      B,#6       ; OF DIVIDEND FIELD
CLRLUP:  LD      [B-],#0     ; CLEAR TEST FIELD
          IFBNE   #3        ; TOP OF DIVIDEND FIELD
          JP      CLRLUP
DVSHFT:  RC
          LD      B,#0
SHFTLP:  LD      A,[B]
          ADC     A,[B]      ; LEFT SHIFT DIVIDEND
          X      A,[B+]     ; AND TEST FIELD
          IFBNE   #7        ; BOTTOM OF DR FIELD
          JP      SHFTLP
          IFC      ; TEST IF BIT SHIFTED
          JP      DVSUBT    ; *** OUT OF TEST FIELD
          SC      ; RESET BORROW
          LD      X,#4
TSTLUP:  LD      A,[X+]     ; TEST SUBTRACT DIVISOR
          SUBC   A,[B]      ; FROM TEST FIELD
          LD      A,[B+]     ; INCREMENT B POINTER
          IFBNE   #10       ; TOP OF DIVISOR + 1
          JP      TSTLUP
          IFNC      ; TEST IF BORROW
          JP      DVTEST    ; FROM SUBTRACTION
          LD      B,#7
DVSUBT:  LD      X,#4
SUBTLP:  LD      A,[X]      ; SUBTRACT DIVISOR
          SUBC   A,[B]      ; FROM REMAINDER
          X      A,[X+]     ; IN TEST FIELD
          LD      A,[B+]     ; INCREMENT B POINTER
          IFBNE   #10       ; TOP OF DIVISOR + 1
          JP      SUBTLP
          LD      B,#0
          SBIT   O,[B]      ; SET QUOTIENT BIT
DVTEST:  DRSZ   CNTR       ; DECREMENT AND TEST
          JP      DVSHFT    ; CNTR FOR ZERO
          RET      ; RETURN FROM SUBROUTINE

```

4.0 DECIMAL (PACKED BCD)/BINARY CONVERSION

Subroutines For Two Byte Conversion:

- | | | | |
|--------|---|--------|--|
| DECBIN | — Decimal (Packed BCD) to Binary | FBTOD | — Fast Binary to Decimal (Packed BCD) |
| | — 24 Bytes *** | | — 59 Bytes |
| | — 1030 Instruction Cycles | | — 334 Instruction Cycles |
| FDTOB | — Fast Decimal (Packaged BCD) to Binary | VFBTOD | — Very Fast Binary to Decimal (Packed BCD) |
| | — 76 Bytes | | — 189 Bytes |
| | — 92 Instruction Cycles | | — 144 Instruction Cycles Average |
| | | | — 208 Instruction Cycles Maximum |
| BINDEC | — Binary to Decimal (Packed BCD) | | |
| | — 25 Bytes *** | | |
| | — 856 Instruction Cycles | | |

***These subroutines extendable to multiple byte conversion by simply changing parameters within subroutine as shown, with number of bytes in subroutine remaining constant.

DECBIN—Decimal (Packed BCD) to Binary

This 24 byte subroutine represents very minimal code for translating a packed BCD decimal number of any length to binary.

ALGORITHM:

The binary result is resident just below the packed BCD decimal number. During each cycle of the algorithm, the decimal operand and the binary result are shifted right one bit position, with the low order bit of the decimal operand shifting down into the high order bit position of the binary field. The residual decimal operand is then tested for a high order bit in each of its nibbles. A three is subtracted from each nibble in the BCD operand space that is found to contain a high order bit equal to one. (This process effectively right shifts the BCD operand one bit position, and then corrects the result to BCD format.) The entire cycle is then repeated, with the total number of cycles being equal to the number of bit positions in the decimal field.

16 Bit: Binary IN [1,0]

Packed BCD in [3, 2]

24 Bit: Binary in [2, 1, 0]

Packed BCD in [5, 4, 3]

32 Bit: Binary in [3, 2, 1, 0]

Packed BCD in [7, 6, 5, 4]

24 Bytes

1030 Instruction Cycles (16 Bit)

```

DECBIN:  LD          CNTR,#16      ; LOAD CNTR WITH NUMBER
;                                     ; OF BIT POSITIONS
;                                     ; IN BCD FIELD
;                                     ; #16 FOR 16 BIT (2 BYTE)
;                                     ; #'S 24/32 FOR 24/32 BIT
DB1:    LD          B,#3          ; #'S 5/7 FOR 24/32 BIT
RC
DB2:    LD          A,[B]         ; PROGRAM LOOP TO
RRC          A                  ; RIGHT SHIFT
X          A,[B-]              ; DECIMAL (BCD) AND
IFBNE      #OF                ; BINARY FIELDS.
JP         DB2                ; LOOP JUMP BACK
LD          B,#3              ; #'S 5/7 FOR 24/32 BIT
SC          ; SET CARRY FOR SUBTRACT
DB3:    LD          A,[B]         ; TEST HIGH ORDER BITS
IFBIT      7,[B]              ; OF BCD NIBBLES, AND
SUBC      A,#030              ; SUBTRACT A THREE
IFBIT      3,[B]              ; FROM EACH NIBBLE IF
SUBC      A,#3                ; HIGH ORDER BIT OF
X          A,[B-]              ; NIBBLE IS A ONE
IFBNE      #1                 ; #'S 2/3 FOR 24/32 BIT
JP         DB3                ; LOOP BACK FOR MORE BCD BYTES
DRSZ      CNTR                ; DECREMENT AND TEST IF
JP         DB1                ; CNTR EQUAL TO ZERO
RET          ; RETURN FROM SUBROUTINE

```

FDTOB—FAST DECIMAL (PACKED BCD) TO BINARY

BCD Format: Four Nibbles — W, X, Y, Z, with W = Hi Order Nibble

*** [1] = 16W + X

*** [0] = 16Y + Z

Algorithm: Binary Result is equal to $100(10W + X) + (10Y + Z)$

BCD IN [1, 0]***

Temp in [2]

Binary in [4, 3]

76 Bytes

92 Instruction Cycles

```

FDTOB:  RC
        LD      B, #1
        LD      A, [B+]          ; 16W + X
        AND     A, #0F0         ; EXTRACT 16W
        RRC     A                ; 8W
        X      A, [B]          ; 8W TO TEMP
        RRC     A                ; 4W
        RRC     A                ; 2W
        ADD     A, [B]          ; 2W + 8W = 10W
        X      A, [B-]         ; 10W TO TEMP
        LD      A, [B+]         ; 16W + X
        AND     A, #0F          ; EXTRACT X
        ADC     A, [B]          ; 10W + X
        X      A, [B]          ; 10W + X TO TEMP
        LD      A, [B]
        ADC     A, [B]          ; 2.(10W + X)
        X      A, [B]          ; 2.(10W + X) TO TEMP
        ADC     A, [B]          ; 3.(10W + X)
        LD      B, #3           ; = 16P + Q
        X      A, [B+]         ; 16P + Q TO [3]
        CLR     A
        IFC
        LD      A, #010         ; 16C TO A (C = CARRY)
        X      A, [B-]         ; 16C TO [4]
        LD      A, [B]          ; 16P + Q
        SWAP    A               ; 16Q + P
        X      A, [B]          ; 16Q + P TO [3]
        LD      A, [B+]         ; 16Q + P
        AND     A, #0F          ; EXTRACT P
        ADD     A, [B]          ; 16C + P
        X      A, [B-]         ; 16C + P TO [4]**
        LD      A, [B]          ; 16Q + P
        AND     A, #0F0         ; EXTRACT 16Q
        X      A, [B-]         ; 16Q TO [3]**
        LD      A, [B+]         ; 2.(10W + X)
        ADC     A, [B]          ; 2.(10W + X) + 16Q

```

```

X          A,[B+]          ; 2 BYTE 2.(10W + X)
CLR       A,[B-]          ; ADD: + 48.**(10W + X)
ADC       A,[B]           ; 16C + P + NU C
X         A,[B-]          ; 50.(10W + X)
LD        A,[B]
ADC       A,[B]           ; DOUBLE
X         A,[B+]          ; 50.(10W + X)
LD        A,[B]           ; TO FORM
ADC       A,[B]           ; 100.(10W + X)
X         A,[B]           ; IN [3,4]
LD        B,#0
LD        A,[B]           ; 16Y + Z
AND       A,#0F0          ; EXTRACT 16Y
LD        B,#2
RRC       A               ; 8Y
X         A,[B]           ; 8Y TO TEMP
LD        A,[B]
RRC       A               ; 4Y
RRC       A               ; 2Y
ADC       A,[B]           ; 2Y + 8Y = 10Y
X         A,[B]           ; 10Y TO TEMP
LD        B,#0
LD        A,[B]           ; 16Y + Z
AND       A,#0F          ; EXTRACT Z
LD        B,#2
ADD       A,[B]           ; 10Y + Z
LD        B, #3
ADC       A,[B]           ; TWO BYTE ADD
X         A,[B+]          ; 100.(10W + X)
CLR       A               ; + (10Y + Z)
ADC       A,[B]           ; WITH BINARY
X         A,[B]           ; RESULT TO [3,4]
RET

```


BINDEC—Binary to Decimal (Packed BCD)

This 25 byte subroutine represents very minimal code for translating a binary number of any length to packed BCD decimal.

ALGORITHM:

The packed BCD decimal result is resident just above the binary number. A sufficient number of bytes must be allowed for the BCD result. During each cycle of the algorithm the binary number is shifted left one bit position. The packed BCD decimal result is also shifted left one bit position, with the high order bit of the binary field being shifted up into the low order bit position of the BCD field. The shifted result in the BCD field is decimal corrected by using the DCOR instruction. Note that for addition an "ADD A, #066" instruction must be used in conjunction with the DCOR (Decimal Correct) instruction. The entire cycle is then repeated, with the total number of cycles being equal to the number of bit positions in the binary field.

16 Bit: Binary in [1, 0]
 Packed BCD in [4, 3, 2]
 24 Bit: Binary in [2, 1, 0]
 Packed BCD in [6, 5, 4, 3]
 32 Bit: Binary in [3, 2, 1, 0]
 Packed BCD in [8, 7, 6, 5, 4]

25 Bytes

856 Instructions Cycles (16 Bit)

```

BINDEC:  LD          CNTR,#16          ; LOAD CNTR WITH NUMBER OF BIT POSITIONS
          ; IN BINARY FIELD
          ; #16 FOR 16 BIT (2 BYTE)
          ; #'S 24/32 FOR 24/32 BIT
          RC
          LD          B,#2             ; #'S 3/4 FOR 24/32 BIT
BD1:     LD          [B+],#0          ; CLEAR BCD FIELD
          IFBNE     #5                ; #'S 7/9 FOR 24/32 BIT
          JP          BD1             ; JUMP BACK FOR CLR LOOP
BD2:     LD          B,#0
BD3:     LD          A,[B]            ; PROGRAM LOOP TO
          ADC        A,[B]            ; LEFT SHIFT
          X          A,[B+]          ; BINARY FIELD
          IFBNE     #2                ; #'S 3/4 FOR 24/32 BIT
          JP          BD3            ; JUMP BACK FOR SHIFT LOOP1
BD4:     LD          A,[B]            ; PROGRAM LOOP TO
          ADD        A,#066           ; LEFT SHIFT AND
          ADC        A,[B]            ; DECIMAL CORRECT
          DCOR      A                ; RESULT OF SHIFT
          X          A,[B+]          ; IN BCD FIELD
          IFBNE     #5                ; #'S 7/9 FOR 24/32 BIT
          JP          BD4            ; JUMP BACK FOR SHIFT LOOP2
          DRSZ      CNTR             ; DECREMENT AND TEST IF
          JP          BD2            ; CNTR EQUAL TO ZERO
          RET
          ; RETURN FROM SUBROUTINE

```

FBTOD—FAST BINARY TO DECIMAL (PACKED BCD)

Algorithm: This algorithm is based on the BINDEC algorithm, except that it is optimized for speed of execution.

Binary in [1, 0]
Packed BCD in [4, 3, 2]
59 Bytes
334 Instruction Cycles

```

FBTOD:  RC
        LD      B, #1
        LD      A, [B]
        SWAP   A          ; REVERSE NIBBLES IN
        X      A, [B]    ; UPPER BINARY BYTE
        LD      A, [B+]  ; EXTRACT ORIGINAL UPPER
        AND    A, #0F    ; NIBBLE OF HI BYTE
        IFGT   A, #9     ; IF NIBBLE GREATER THAN
        ADD    A, #06    ; NINE, THEN ADD SIX TO CORRECT BCD NIBBLE
        X      A, [B+]  ; NIBBLE TO LOWER BCD BYTE
        LD     [B+], #0  ; CLEAR UPPER BCD BYTES
        LD     [B], #0   ; INITIALIZE CNTR TO COVER
        LD     CNTR, #4  ; REMAINING HI NIBBLE (ORIGINALLY LO NIBBLE)
                          ; IN UPPER BINARY BYTE
FB1:    LD      B, #1    ; PROGRAM LOOP TO
        LD      A, [B]  ; LEFT SHIFT A BIT
        ADC    A, [B]  ; OUT OF UPPER BINARY
        X      A, [B+]  ; BYTE INTO LOW ORDER
        LD     A, [B]  ; BIT POSITION OF BCD
        ADD    A, #066  ; FIELD, AS LOWER TWO
        ADC    A, [B]  ; BYTES OF BCD FIELD
        DCOR   A        ; ARE LEFT SHIFTED WITH
        X      A, [B+]  ; THE LOWER BYTE BEING
        LD     A, [B]  ; DECIMAL CORRECTED
        ADC    A, [B]  ; MIDDLE BYTE OF BCD FIELD
        X      A, [B]  ; NEED NOT BE DECIMAL CORRECTED, SINCE
                          ; MAX VALUE IS 2 (256)
        DRSZ   CNTR    ; DECREMENT AND TEST IF
        JP     FBD1    ; CNTR EQUAL TO ZERO
        LD     CNTR, #8 ; INITIALIZE CNTR TO COVER
FB2:    LD      B, #0    ; LOWER BINARY BYTE
        LD      A, [B]  ; PROGRAM LOOP TO
        ADC    A, [B]  ; LEFT SHIFT A BIT
        X      A, [B]  ; OUT OF LOWER BINARY
        LD     B, #2    ; BYTE INTO LOW ORDER
        LD     A, [B]  ; BIT POSITION OF BCD
        ADD    A, #066  ; FIELD, AS BCD FIELD
        ADC    A, [B]  ; IS LEFT SHIFTED WITH
        DCOR   A        ; THE LOWER TWO BYTES
        X      A, [B+]  ; OF THE FIELD BEING
        LD     A, [B]  ; DECIMAL CORRECTED
        ADD    A, #066  ; ADD (NOT ADC) HEX 66
        ADC    A, [B]  ; TO SET UP "ADD" DCOR
        DCOR   A        ; DECIMAL CORRECT MIDDLE
        X      A, [B+]  ; BYTE OF BCD FIELD
        LD     A, [B]  ; UPPER BYTE OF BCD FIELD
        ADC    A, [B]  ; NEED NOT BE DECIMAL
        X      A, [B]  ; CORRECTED, SINCE MAX
                          ; VALUE IS 6 (65535)
        DRSZ   CNTR    ; DECREMENT AND TEST IF
        JP     FBD2    ; CNTR EQUAL TO ZERO
        RET
    
```

VFBTOD—VERY FAST BINARY TO DECIMAL (PACKED BCD)

Algorithm: Decimal (Packed BCD) result is equal to summation in BCD of powers of two corresponding to 1's bits present in binary number.

Note that binary field (2 bytes) is initially one's complemented by program, in order to facilitate bypass branching when a tested bit in the binary field is found equal to zero.

Binary in [1, 0]
BCD in [4, 3, 2]

189 Bytes
144 Instruction Cycles Average
208 Instruction Cycles Maximum

```

VFBTOD:  RC
          LD      B,#0
          LD      A,[B]
          AND     A,#0F          ; EXTRACT LO NIBBLE
          IFGT   A,#9          ; TEST NIBBLE 9
          ADD     A,#6          ; ADD 6 FOR CORRECTION
          LD      B,#2
          X      A,[B+]        ; STORE IN LO BCD NIBBLE
          LD     [B+],#0        ; CLEAR UPPER
          LD     [B],#0         ; BCD NIBBLES
          LD     B,#1
          LD     A,[B]
          XOR    A,#0FF        ; COMPLEMENT HI BYTE
          X      A,[B-]        ; FOR REVERSE TESTING
          LD     A,[B]         ; OF BINARY NUMBER
          XOR    A,#0FF        ; COMPLEMENT LO BYTE
          X      A,[B]         ; FOR REVERSE TESTING
          IFBIT  4,[B]        ; TEST BINARY BIT 4
          JP     VFB1         ; TO CONDITIONALLY
          LD     B,#2          ; ADD BCD 16
          LD     A,#07C        ; 16 + 66
          ADC    A,[B]         ; ADD BCD 16
          DCOR   A
          X      A,[B]
          LD     B,#0

VFB1:    IFBIT  5,[B]        ; TEST BINARY BIT 5
          JP     VFB2         ; TO CONDITIONALLY
          LD     B,#2          ; ADD BCD 32
          LD     A,#098        ; 32 + 66
          ADC    A,[B]         ; ADD BCD 32
          DCOR   A
          X      A,[B]
          LD     B,#0

VFB2:    IFBIT  6,[B]        ; TEST BINARY BIT 6
          JP     VFB3         ; TO CONDITIONALLY
          LD     B,#2          ; ADD BCD 64
          LD     A,#0CA        ; 64 + 66
          ADC    A,[B]         ; ADD BCD 64
          DCOR   A
          X      A,[B+]
          CLR    A
          ADC    A,[B]         ; ADD CARRY
          X      A,[B]
          LD     B,#0

```

```

VFB3:  IFBIT      7,[B]          ; TEST BINARY BIT 7
        JP        VFB4          ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 128
        LD        A,#08E        ; 28 + 66
        ADC       A,[B]         ; ADD BCD 28
        DCOR     A
        X        A,[B+]
        LD        A,#1
        ADC       A,[B]         ; ADD BCD 1
        X        A,[B]
VFB4:  LD        B,#1          ; HI BINARY BYTE
        IFBIT    0,[B]         ; TEST BINARY BIT 8
        JP        VFB5          ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 256
        LD        A,#0BC        ; 56 + 66
        ADC       A,[B]         ; ADD BCD 56
        DCOR     A
        X        A,[B+]
        LD        A,#2
        ADC       A,[B]         ; ADD BCD 2
        X        A,[B]
        LD        B,#1
VFB5:  IFBIT    1,[B]          ; TEST BINARY BIT 9
        JP        VFB6          ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 512
        LD        A,#078        ; 12 + 66
        ADC       A,[B]         ; ADD BCD 12
        DCOR     A
        X        A,[B+]
        LD        A,#06B        ; 5 + 66
        ADC       A,[B]         ; ADD BCD 5
        DCOR     A
        X        A,[B]
        LD        B,#1
VFB6:  IFBIT    2,[B]          ; TEST BINARY BIT 10
        JP        VFB7          ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 1024
        LD        A,#08A        ; 24 + 66
        ADC       A,[B]         ; ADD BCD 24
        DCOR     A
        X        A,[B+]
        LD        A,#076        ; 10 + 66
        ADC       A,[B]         ; ADD BCD 10
        DCOR     A
        X        A,[B]
        LD        B,#1
VFB7:  IFBIT    3,[B]          ; TEST BINARY BIT 11
        JP        VFB8          ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 2048
        LD        A,#0AE        ; 48 + 66
        ADC       A,[B]         ; ADD BCD 48
        DCOR     A
        X        A,[B+]
        LD        A,#086        ; 20 + 66
        ADC       A,[B]         ; ADD BCD 20
        DCOR     A
        X        A,[B]
        LD        B,#1

```

```

VFB8:  IFBIT      4,[B]          ; TEST BINARY BIT 12
        JP        VFB9          ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 4096
        LD        A,#0FC        ; 96 + 66
        ADC       A,[B]        ; ADD BCD 96
        DCOR     A
        X        A,[B+]
        LD        A,#0A6        ; 40 + 66
        ADC       A,[B]        ; ADD BCD 40
        DCOR     A
        X        A,[B]
        LD        B,#1
VFB9:  IFBIT      5,[B]          ; TEST BINARY BIT 13
        JP        VFB10         ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 8192
        LD        A,#0F8        ; 92 + 66
        ADC       A,[B]        ; ADD BCD 92
        DCOR     A
        X        A,[B+]
        LD        A,#0E7        ; 81 + 66
        ADC       A,[B]        ; ADD BCD 81
        DCOR     A
        X        A,[B]
        CLR     A
        ADC       A,[B]        ; ADD CARRY
        X        A,[B]
        LD        B,#1
VFB10: IFBIT      6,[B]          ; TEST BINARY BIT 14
        JP        VFB11         ;   TO CONDITIONALLY
        LD        B,#2          ;   ADD BCD 16384
        LD        A,#0EA        ; 84 + 66
        ADC       A,[B]        ; ADD BCD 84
        DCOR     A
        X        A,[B+]
        LD        A,#0C9        ; 63 + 66
        ADC       A,[B]        ; ADD BCD 63
        DCOR     A
        X        A,[B+]
        LD        A,#1
        ADC       A,[B]        ; ADD BCD 1
        X        A,[B]
        LD        B,#1
VFB11: IFBIT      7,[B]          ; TEST BINARY BIT 15
        RET
        LD        B,#2          ;   TO CONDITIONALLY
        LD        A,#0CE        ;   ADD BCD 32768
        ADC       A,[B]        ; 68 + 66
        DCOR     A
        X        A,[B+]
        LD        A,#08D        ; 27 + 66
        ADC       A,[B]        ; ADD BCD 27
        DCOR     A
        X        A,[B+]
        LD        A,#3
        ADC       A,[B]        ; ADD BCD 3
        X        A,[B]
        RET

```

Pulse Width Modulation A/D Conversion Techniques with COP800 Family Microcontrollers

National Semiconductor
Application Note 607
Kevin Daugherty



1.0 BASIC TECHNIQUE

This application note describes a technique for creating an analog to digital converter using a microcontroller with other low cost components. Many applications do not require the speed associated with a dedicated hardware A/D converter and it is worth evaluating a more cost effective approach.

With a high speed CMOS microcontroller an eight bit A/D can be implemented that converts in approximately 10 ms. This method is based on the fact that if a repetitive waveform is applied to an RC network, the capacitor will charge to the average voltage, provided that the RC time constant is much larger than the pulse widths. The basic equation for computing the analog to digital result is:

$$V_{in} = V_{ref} [T_{on} / (T_{on} + T_{off})] \quad (1)$$

With this equation it is necessary to precisely measure several time periods within both the T_{on} and T_{off} in order to achieve the desired resolution. Additionally, the waveform would have to be gradually adjusted to allow for the large RC time constant to settle out. This results in a relatively long conversion cycle. Modifying the equation and technique slightly, significantly speeds up the process. This technique works by averaging several pulses over a fixed period of time and is based on the following equation:

$$V_{in} = V_{ref} [\text{Sum of } T_{on} / (\text{Sum of } (T_{on} + T_{off}))] \quad (2)$$

2.0 IMPLEMENTATION

Figure 1 describes the basic circuit schematic that uses a National Semiconductor COP822C microcontroller, a low cost LM2901 comparator, two 100k resistors, and a 0.047 mfd film capacitor. The CMOS COP822C microcontroller provides a squarewave signal with logic levels very close to GND and V_{CC} . This generates a small ramp voltage on the capacitor for the LM2901 quad comparator input.

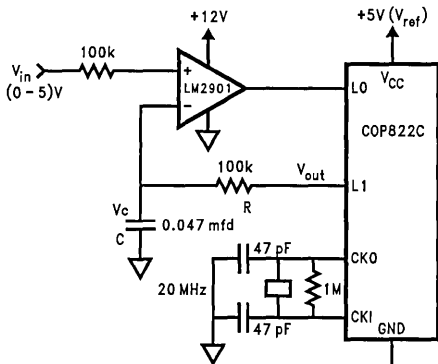


FIGURE 1. Basic Circuit

TL/DD/10407-1

To minimize error, a tradeoff must be made when selecting the resistor. The microcontroller output (L1) should have a large resistor to minimize the output switching offset (V_{os}), and the comparator should have a small resistor due to error caused by I_{bos} (input bias offset current).

Once the resistor is determined, the capacitor should be chosen so that the RC time constant is large enough to provide a small incremental voltage ramp. This design has a sample time of 20 μ s and has a 4.7 ms time constant with a 0.047 mfd film type capacitor which has low leakage current to prevent errors. Since a 100k resistor is used in the RC network for one comparator input, another 100k resistor is required for the V_{in} input to balance the offset voltage caused by the comparator I_b (input bias current).

Figure 2 illustrates the relationship between the microcontroller squarewave output and the capacitor charge and discharge. Every 20 μ s the comparator is sampled. If the capacitor voltage (V_c) is below V_{in} the RC network will receive a positive pulse. The inverse is true if V_c is above V_{in} at sample time. Note that with this approach, the PWM waveform is broken up into several small pulses over a fixed period instead of having a single pulse represent the duty cycle; thus a relatively small RC time constant can be used.

Mathematical Analysis:

let n = total number of T_{on} pulses and

m = total number of T_{off} pulses

then $V_c(t) = V_c + n[(V_{out} - V_c)(1 - e^{-t/RC})] - m[(V_c - V_o)(1 - e^{-t/RC})]$

let $V_c = V_{in}$ at start of conversion and

$$K = (1 - e^{-t/RC})$$

then $V_{in} = V_{in} + K_n V_{out} - K_n V_{in} - K_m V_{in} + K_m V_o$

$$0 = K_n V_{out} + K_m V_o - K V_{in} (n + m)$$

let $V_{out} = V_{ref} - V_{os}$

solving for V_{in} :

$$V_{in} = nV_{ref}/(n + m) - (nV_{os} - mV_o)(1/(n + m)) \quad (3)$$

Note that the RC value drops out of the equation and therefore is not an error factor.

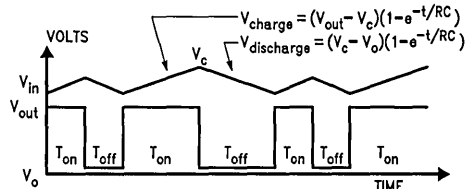


FIGURE 2. PWM Signal

TL/DD/10407-2

3.0 SOFTWARE DESCRIPTION

Single Channel

Referring to the flow chart in *Figure 3*, and the code listed in *Figure 4*, the software counters T_{ON} and TOTAL are first preloaded with the FF. The accumulator and register 0F1 are then loaded with 2 to provide for an initialization and final conversion cycle. Next, the L port is configured to complete the initialization of the microcontroller.

The comparator output is checked with the IFBIT 0,0D2 instruction. This will determine whether the RC network will receive a positive (V_{ref}) or ground pulse. You can think of the microcontroller as part of the feedback path of the comparator. The microcontroller uses the comparator output to decide what level output on L1 is required to keep the capacitor equal to the unknown input voltage. Each time the negative or GND pulse is applied, the T_{ON} counter is decremented by DRSZ. Similarly, each time a sample loop is completed the TOTAL counter is decremented by DRSZ. Note that NOP instructions are used in the high and low loops. These are necessary to provide exactly the same cycles for a high or low L1 output pulse.

Once the TOTAL register is decremented to zero, the initialization loop is completed. Immediately afterwards, the L1 output is put in TRI-STATE® mode to minimize capacitor voltage variations while other instructions are completed. After the first conversion, the IFEQ A,0F1 instruction will be true and the T_{ON} and TOTAL registers will be reloaded with FF. Following this, the L1 pin is restored as a high output and the 0F1 multiplier is decremented.

At this point the capacitor is equal to V_{in} and the actual conversion is started. When the TOTAL register is decremented to zero (255 samples later), the conversion is complete. T_{ON} will not be reloaded since 0F1 was decremented and IFEQ A,0F1 will no longer be true. The accumulator is then loaded with T_{ON} and stored in RAM location 00 with X A,00.

The final two instructions (RBIT 1,LCONF & RBIT 1[B]) are optional depending on the application and the amount of additional code required. This will prevent the capacitor from decaying appreciably between conversions and allow for a much quicker capacitor initialization time. Otherwise more time may be required, or a diode speed-up circuit as shown in *Figure 7d* is required to fully charge the capacitor prior to starting the actual conversion.

Eight Channel

This is basically the same as that for the single channel. Referring to the flow chart in *Figure 5* and the code in *Figure 6*, the differences are in the front and back ends. Before the

conversions are started, the X register is initialized to 00 for RAM location 00. The accumulator is then loaded with the current RAM pointer (LD A,X), OR'ed with the LDATA (OR A,LDATA), and finally the LDATA register is modified to provide for the proper output select (X A,LDATA).

Following the actual conversion cycle, the result is stored at the current RAM pointer (X A,[X+]) which also auto-increments the X register. The next conversion will use this to select the next channel and determine where to store the result. Once the eighth channel is converted, the IFEQ A,X instruction will be true and the RAM pointer will be reset (LD X,#00) before the next conversion is started.

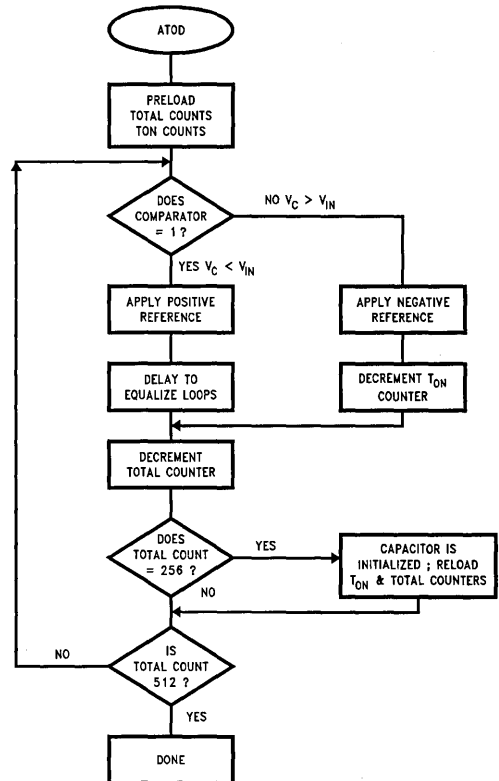


FIGURE 3. PWM A/D Flow Chart

TL/DD/10407-7

```

;The program listed below will work in any COP800 microcontroller
;(i.e. COP820, COP840, COP880, COP888). SET UP FOR .047 mfd CAP.,
;LOOK RES, @1 MICRO. CYCLE TIME. THE FIRST CONVERSION
;INITIALIZES, AND 2nd IS THE RESULT STORED IN RAM LOCATION 00.
.CHIP 820
LCONF=0D1
LDATA=0D0
TON=0F2
TOTAL=0F0
;
LD A,#02          ;USED TO DETERMINE WHEN TO RELOAD
LD TOTAL,#0FF    ;PRELOAD TOTAL COUNTS
LD OF1,#2        ;MULTIPLIER (255 TO INIT. PLUS 255 FOR RESULT)
LD TON,#0FF      ;PRELOAD Ton
LD OFE,#0D0      ;LOAD B REG TO POINT TO LDATA REG.
LD LDATA,#01     ;L PORT DATA REG, LO=WEAK PULL UP, L1=HIGH
LD LCONF,#02     ;L PORT CONFIG REG, LO=INPUT, L1=OUTPUT
LOOP: IFBIT 0,0D2 ;TEST COMPARATOR OUTPUT
JP HIGH          ;JUMP IF LO=1
NOP
NOP
NOP              ;EQUALIZE TIME FOR SETTING AND RESETTING
RBIT 1,[B]       ;DRIVE L1 LOW
DRSZ Ton        ;DECREMENT Ton WHEN DRIVING LOW
JMP COUNT
HIGH: SBIT 1,[B] ;DRIVE L1 HIGH
NOP
NOP
NOP
NOP
NOP
NOP              ;EQUALIZE HIGH AND LOW LOOPS
COUNT: DRSZ TOTAL ;DECREMENT TOTAL COUNTS
JP LOOP
RBIT 1,LCONF     ;TRISTATE L1 TO MINIMIZE ERRORS FROM EXTRA
RBIT 1,[B]       ;CYCLES
IFEQ A,OF1       ;CHECK INITIALIZATION LOOP COMPLETE
JP RELOAD        ;JUMP IF TRUE.
JP DEC           ;JUMP IF NOT END OF 2nd LOOP
RELOAD: LD OF2,#0FF ;RELOAD Ton WITH FF
LD OF0,#0FF      ;SYNC TOTAL AND Ton COUNTERS
DEC: SBIT 1,[B]   ;SET L1 HIGH
SBIT 1,LCONF     ;RESTORE L1 AS OUTPUT.
DRSZ OF1         ;DECREMENT MULTIPLIER UNTIL ZERO
JMP LOOP         ;CONTINUE A/D UNTIL AFTER 2nd CONVERSION
LD A,TON         ;LOAD A WITH Ton
X, A,00          ;STORE RESULT IN RAM LOCATION 00
.end

```

FIGURE 4. Single Channel PWM A/D Listing

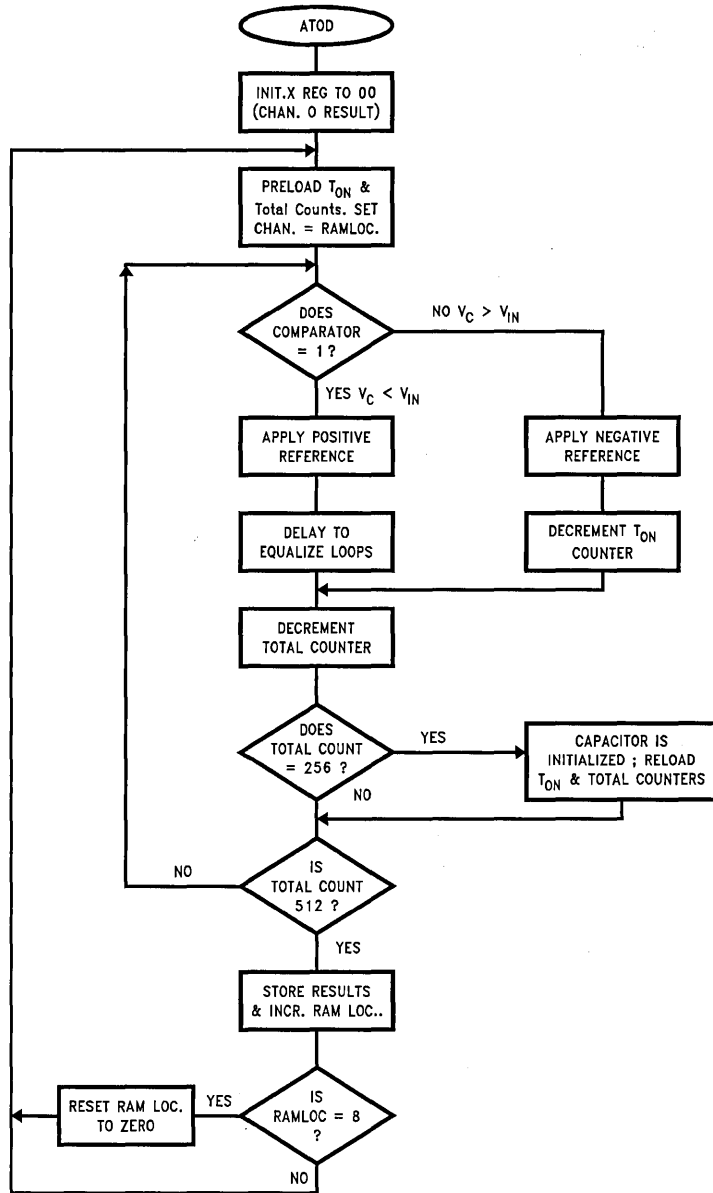


FIGURE 5. 8 Channel PWM A/D Flow Chart

TL/DD/10407-8

```

;L0,1,2 SELECTS CHANNEL OF CD4051 8:1 MUX, L3 IS THE COMP.
;OUTPUT, AND L4 DRIVES THE RC. RESULTS STORED IN RAM 00-07.
.CHIP 820
LDATA=0D0
LCONF=0D1
TON=0F2
TOTAL=0F0
      LD X,#00          ;INITIALIZE X REG FOR 1st RAM LOC.
CONVER: LD TOTAL,#OFF  ;PRELOAD TOTAL COUNTS
      LD OF1,#02       ;TOTAL LOOP COUNTER
      LD TON,#OFF      ;PRELOAD Ton
      LD OFE,#0D0      ;INIT. B REG TO POINT TO LDATA REG
      LD LDATA,#018    ;LDATA, L0-2=LOW, L3=PULLUP, L4=HIGH
      LD A,X           ;USED CURRENT RAM POINTER TO SELECT-
      OR A,LDATA       ;PROPER A/D CHANNEL.
      X A,LDATA        ;MODIFY LDATA FOR CHANNEL SELECTION.
      LD LCONF,#017   ;LCONF REG. L0-L2, L4=OUTPUT, L3=IN
LOOP:  IFBIT 3,0D2    ;TEST COMPARATOR OUTPUT AT L3 INPUT
      JMP HIGH        ;JUMP IF L3=HIGH
      NOP
      NOP             ;EQUALIZE TIME FOR SET AND RESET
      RBIT 4,[B]      ;DRIVE L4 LOW WHEN COMPARATOR IS LOW.
      DRSZ TON        ;DECREMENT Ton WHEN APPLYING NEG. REF.
      JMP COUNT       ;JUMP TO COUNT UNLESS Ton REACHES ZERO
HIGH:  SBIT 4,[B]     ;DRIVE L4 HIGH WHEN COMPARATOR IS HIGH
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP             ;EQUALIZE HIGH AND LOW LOOP TIMES
COUNT: DRSZ TOTAL   ;DEC. TOTAL COUNTS EACH LOOP
      JMP LOOP        ;JUMP UNLESS TOTAL CNTS.=0
      RBIT 4,LCONF    ;TRISTATE L4 TO MINIMIZE ERROR
      RBIT 4,[B]     ; "
      LD A,#02       ;USE TO DETERMINE WHEN TO RELOAD
      IFEQ A,0F1     ;CHECK FOR 2nd CONVERSION COMPLETE
      JP RELOAD      ;IF TRUE.
      JP DEC         ;OTHERWISE JUMP TO DEC
RELOAD: LD TON,#OFF  ;RELOAD Ton FOR START OF NEXT CONV.
      LD TOTAL,#OFF  ;SYNC Ton AND TOTAL COUNTERS
DEC:   SBIT 4,[B]    ;SET L4 HIGH
      SBIT 4,LCONF   ;RESTORE L4 AS OUTPUT.
      DRSZ OF1       ;DECREMENT TOTAL LOOP UNTIL ZERO
      JMP LOOP       ;DONE WHEN OF1 IS ZERO.
      LD A,TON       ;LOAD A WITH Ton RESULT
      X A,[X+]       ;STORE RESULT AT CURRENT RAM POINTER
                        ;AND AUTO INCREMENT POINTER
      LD A,#08       ;CHECK [X] RAM POINTER FOR
      IFEQ A,X       ;EIGHTH CHANNEL CONVERTER
      LD X,#00       ;RESET RAM POINTER IF [X]=8
      JMP CONVER
.END

```

FIGURE 6. 8-Channel PWM A/D Listing

4.0 ACCURACY AND CIRCUIT CONSIDERATIONS

The basic circuit will provide 8 bits ± 1 LSB accuracy depending on the choice of comparator, and passive components. With this type of design several tradeoffs and error sources should be considered. First of all, conversion equation 2 assumes that the microcontroller output switches exactly to GND and V_{CC} (or V_{ref}). The COP822C will typically switch between 10 mV and 20 mV from GND and V_{CC} with a light load. This will cause an error equal to the offset voltage times the duty cycle (equ. 3). Fortunately, the offsets tend to cancel each other at mid range voltages. At near GND and V_{CC} input voltages the offsets are minimal due to the very small voltage drop across the resistor. If the error is undesirable, the offset voltage can be reduced by paralleling outputs with the same levels together, or by using a CMOS buffer such as a 74HC04 to drive the RC network (see Figure 7 for suggested circuits).

Another possible source of error is with the LM2901 worst case input bias offset current of 200 nA over temperature. This will cause an error equal to $R_{in} \times I_{bos}$, which equals 20 mV with a 100k resistor. Either the resistor or the I_{bos} can be reduced to improve the error. If the resistor is reduced then the L port offset voltages will increase so the preferred approach is to select a comparator with lower I_{bos} such as the LP339 which has an I_{bos} of only ± 15 nA. The comparator V_{os} may also introduce error. The LM2901 V_{os} is ± 9 mV, the LP339 V_{os} is only ± 5 mV. An added benefit of using the LP339 is that since the I_{bos} is so small, the resistor for the RC network can be larger. In addition, one RC network could be used for several comparator input channels (refer to Figure 7A).

By using the LM604 (Figure 7B) the basic software can be easily extended for converting several channels. This will only require a control line to be selected before a conversion is started. Since the LM604 needs to be powered from a higher voltage than the input voltage range, the output voltage will also be higher than the microcontroller supply. This requires a current limiting resistor to be used in series

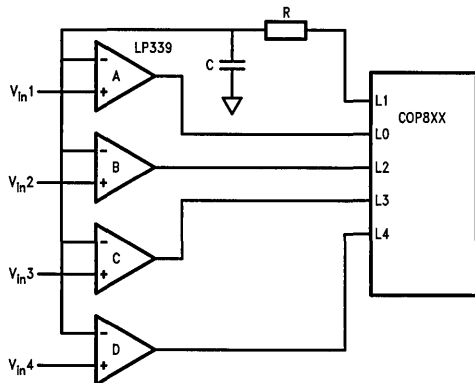
between the LM604 output and the COP8XX. Note that two or more LM604's can be paralleled for providing several more A/D channels by utilizing the EN control input that can TRI-STATE the LM604 output when high.

When more than 4 channels of analog signals are required to be measured, the circuit in Figure 7(d) is recommended. This circuit utilizes an inexpensive CD4051 8:1 multiplexer with a single comparator (which could be on-board the micro). When measuring several input voltages that can vary, TRI-STATING the output driving the RC between conversions is not possible. It is necessary to provide 6x RC time constants to charge the capacitor to within 0.25%. Note that there are two 1N4148's across the comparator inputs. The diodes provide a quick capacitor charge path providing that the total input resistance is much smaller than the resistor used in the RC network (a 2k resistor will meet the requirements within 255 sample times). Once the capacitor is charged to within about 0.6V, the diodes will start turning off. At this point the microcontroller will start dominating the charge/discharge of the capacitor. After the initialization cycle is complete, the capacitor is very close to the unknown V_{in} and the diodes are effectively out of the circuit.

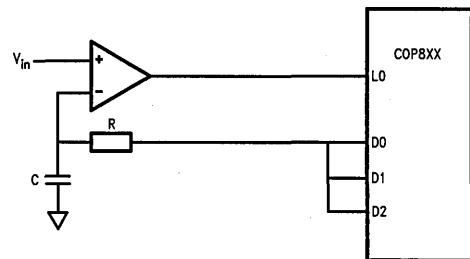
Depending on the speed and accuracy requirements, the total number of counts used in the conversion can be changed. Increasing the counts will give more accuracy with the practical limit of about 9–10 bits. With increased resolution, the capacitor ramp voltage per sample time should be decreased so that the capacitor can be initialized to within 1 LSB prior to conversion. This can be done by either increasing the RC time constant, or by using an initialization routine with a shorter sample time. The conversion time will depend on the total counts and the microcontroller oscillator frequency as described below:

$$T_{con} = \text{Total counts} \times (20 \text{ cycles}) \times (\text{instruction cycle time})$$

Another factor to consider is when a non-ratiometric conversion is required, the reference voltage must have the tolerance to match the desired accuracy.



A. Multiple Channels with LP339 Low I_{bos} Comparator

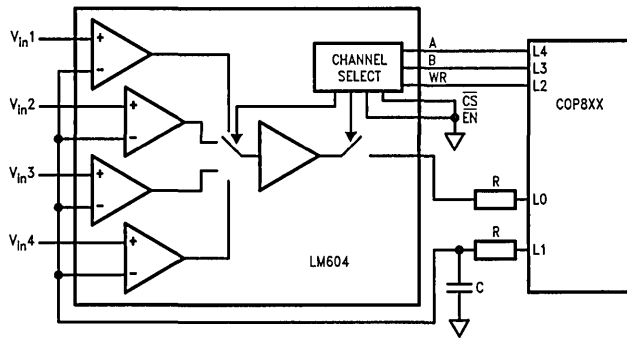


B. High Drive with Multiple Outputs

TL/DD/10407-4

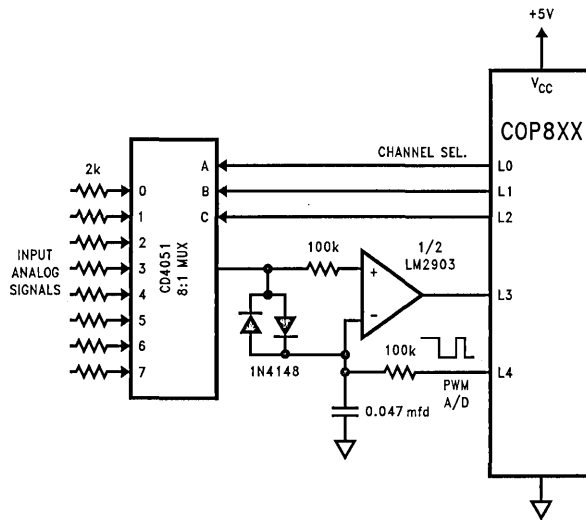
TL/DD/10407-5

FIGURE 7. Suggested Circuits



C. Four Channel A/D with LM604 MUX-Amplifier

TL/DD/10407-6



D. Eight Channel PWM A/D Circuit

TL/DD/10407-9

FIGURE 7. Suggested Circuits (Continued)

5.0 CONCLUSION

The PWM A/D technique described in this application note provides a relatively fast discrete implementation with substantial cost savings compared to a dedicated hardware A/D. Minimal microcontroller I/O and software is required to interface with a comparator and RC network. Depending on the application requirements, the designer can tailor the basic 8-bit A/D a number of ways. By varying the total software counts, the desired speed and resolution can be adjusted. The number of A/D channels will determine the number of comparators used. In choosing the comparator, it is recommended that the designer refer to the data sheets and match the I_{bos} and V_{os} to the desired accuracy.

When other than a $1 \mu s$ instruction cycle is used, the RC time constant of 4.7 ms should be scaled to provide for

a maximum peak-peak ramp voltage of <1 LSB of the desired accuracy. For example, if 8-bit accuracy is desired and the instruction cycle time is now $4 \mu s$ instead of $1 \mu s$, multiply 4.7 ms by 4 to calculate the new RC.

Keep in mind that the comparator input voltage is limited so that you do not get erroneous/nonlinear results. Another possible problem is during development. When doing in-circuit emulation with the development equipment, note that there will be ground loops in the cable thus causing errors in your measurements. You can reduce this by connecting an extra GND and V_{CC} wire between your prototype and development system power and GND. It is still possible to see offsets in the sockets holding the COP8XX in the development board, however this should be relatively small. The best test is to take accurate measurements with an emulator in the actual prototype circuit.

COP800 Based Automated Security/Monitoring System

National Semiconductor
Application Note 662
Ramesh Sivakolundu



INTRODUCTION

National Semiconductor's COP800 family of full-feature, cost effective, fully static, single chip micro CMOS micro-controllers provide efficient system solutions with a versatile instruction set and high functionality. The heart of the ASM System prototype is a COP800 family member with at least the following features: 4k bytes of on-board program memory, 192 bytes of on-board data memory, memory mapped I/O, fourteen multi-sourced vectored interrupts and a versatile instruction set. The family member used is the COP888CG microcontroller.

This application note describes the implementation of a Security/Monitoring System using the COP888CG microcontroller. The COP888CG contains features such as:

- Low power HALT and IDLE modes
- MICROWIRE/PLUS™ serial communication
- Multiple multi-mode general purpose timers
- Multi-input wakeup/interrupt
- WATCHDOG™ and Clock monitor
- Maskable vectored interrupt scheme
- UART

In addition to these features common to the COP888 sub-family of microcontrollers, COP888CG has a full duplex, double buffered UART and two Differential Comparators.

The COP888CG based Automated Security/Monitoring (ASM) System consists of several features:

- Automatic Telephone Dialing
- Real Time Clock
- Non-Volatile storage of real time information of events
- Continuous display of events on the terminal
- Battery operated remote sensors and transmitters
- Exit and Entry delays
- Expandable to add new features

SYSTEM OVERVIEW

Figure 1 gives the block diagram of the ASM System prototype hardware. The application consists of following major blocks:

- Central Controlling Unit
- Receiver
- Sensors and Transmitters
- Keypad Unit
- Auto-Dialer Unit
- Data Storage Unit
- Display Terminal Unit
- LED Display Unit

The implementation allows easy expansion of the ASM System features by adding new blocks to the Central Controlling Unit.

COP888CG is the workhorse of the ASM System and provides the processing power to scan the keypad, service the Receiver interrupts, update the real time clock, serially communicate with the LED display unit and Data Storage Unit, activate the Auto-Dialer Unit and use the full-duplex double buffered UART to interface with the Display Terminal Unit. System capabilities may be enhanced or scaled down by simply changing the processor's algorithm. The subsequent sections describe each of the units and their interface with the COP888CG.

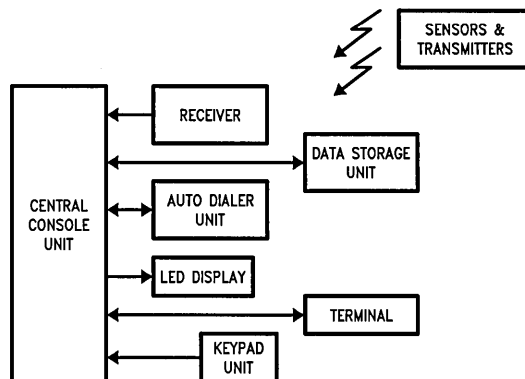


FIGURE 1. Block Diagram of Security/Monitoring System

TL/DD/10607-1

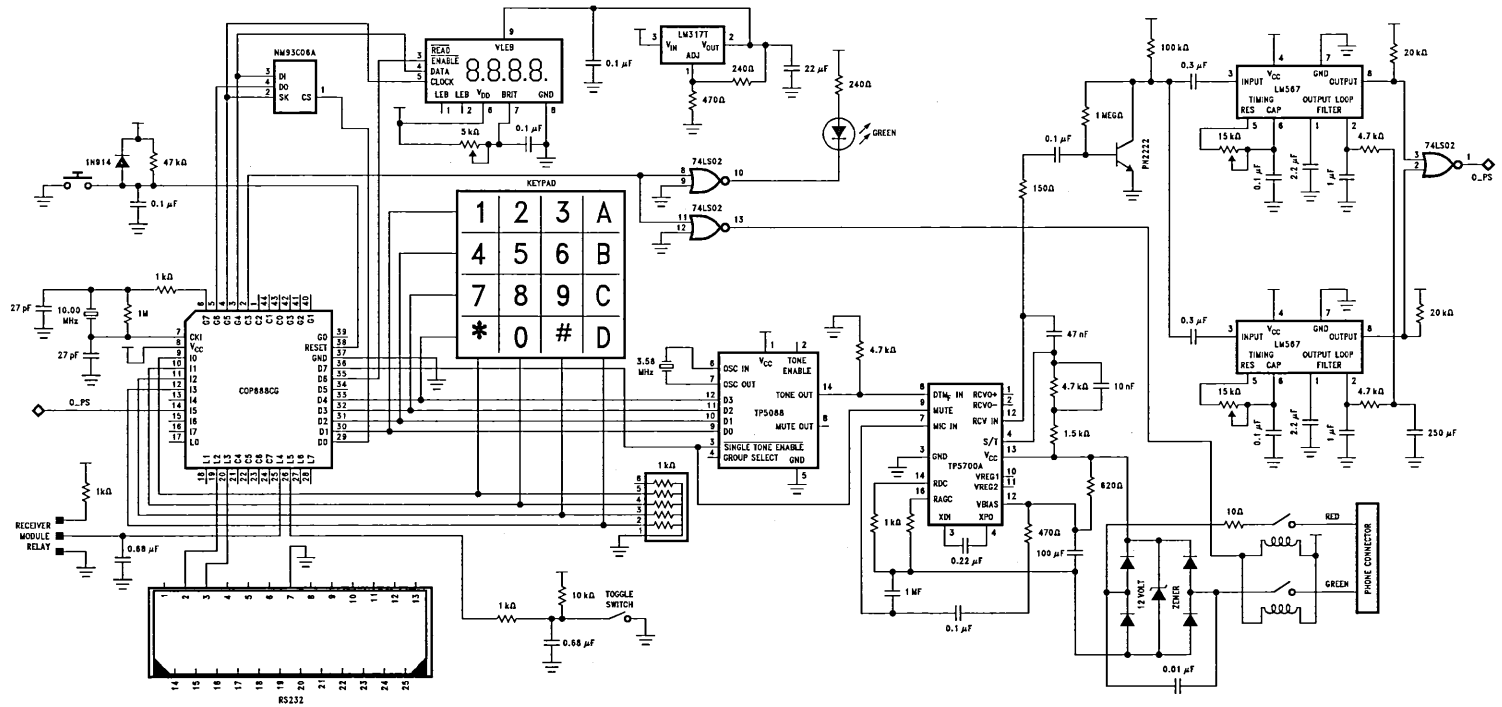


FIGURE 2. Schematic of the ASM System Prototype

TL/DD/10607-2

HARDWARE DESCRIPTION

This section describes the various blocks in the ASM System briefly and highlights the hardware considerations in the design of the System.

Receiver Unit

The Receiver Unit operates with the Sensors and Transmitter Unit. An eight-key dip switch makes it possible to select 256 different digital codes. A detector LED indicates the level of the radio frequency (RF) energy detected by the receiver and enables the user to determine the best locations for the transmitter(s) and receiver, assuring reliable operation.

Figure 2 shows the interface between the COP888CG and the Receiver Unit on the bi-directional I/O Port L capable of functioning as Multi-Input WakeUp (MIWU). In this implementation the WR-200 series of receivers manufactured by Visonic Ltd was used. These receivers are designed to operate with Visonic standard transmitters. The receiver operates on 12 VDC. When RF signal from the transmitter(s) is detected, the receiver activates a relay which in turn interrupts the microcontroller. The output of the relay is connected to the Port L of the COP888CG whose alternate function includes, the Multi-Input WakeUp feature. The COP888CG, after a time delay of 10 seconds, activates the Auto-Dialer Unit. The microcontroller turns on a LED to indicate an alarm signal was detected and is being processed.

Sensors and Transmitters

This unit has a built-in reed switch which can be used with a magnet to activate the transmitter. An eight-key dip switch forms the code selector and each key can be set to either ON or OFF position to create a unique code. This code should match with the code selected on the receiver unit.

Model WR-100 Universal Wireless Transmitter, manufactured by Visonic Ltd. was used in the implementation of the Security/Monitoring System.

Keypad Unit

The Keypad Unit consists of 4 x 4 matrix keyboard. The *Figure 2* shows the keyboard matrix interface to COP888CG. The keyboard is scanned periodically by addressing a column in the keyboard matrix. The program senses the key closure in that column by testing the Port I lines (I0 to I3) which are connected to the rows of the keyboard matrix. Thus, each key is associated with the conjunction of one Port D output line and one Port I input line only.

The keypad unit is used to program the real time clock in order to set the time and date. The telephone number to be dialed in case of a security breach can also be programmed through the keypad as well as the terminal keyboard in the Terminal Unit.

Auto-Dialer Unit

The Auto-Dialer Unit dials the number programmed by the user upon detection of RF signal by the Receiver from the Sensors and Transmitter Unit. The unit consists of two ICs and some peripheral circuitry. National Semiconductor's TP5700A is the Telephone Speech Circuit and TP5088 is the DTMF generator. These two chips are interfaced to the COP888CG as in *Figure 2*. The COP888CG outputs the digit to be dialed to TP5088 and the output of the DTMF generator is inputted to the Speech Circuit. The Speech Circuit interfaces with the telephone lines.

TP5088 is a low cost CMOS device that provides the toning capability in microprocessor-controlled telephone applications. TP5700A is a linear bipolar device which includes the functions required to build the speech circuit of a telephone. It replaces the hybrid transformer, compensation circuit and sidetone network used traditional designs.

Data Storage Unit

The Data Storage Unit stores the real time data of events that the Receiver Unit detects and informs the Central Controlling Unit. The storage is non-volatile and can be archived for later references. The Terminal Unit can request the Central Controlling Unit to display the events and the data stored in the Storage Unit. The telephone number to be dialed by the Auto-Dialer Unit is also stored in this unit. This unit interfaces with the COP888CG using the MICROWIRE/PLUSTM serial communication protocol.

In this implementation the COP888CG microcontroller interfaces with NM93C06A Serial EEPROM Memory. The NM93C06A contains 256 bits of read/write EEPROM organized as 16 registers of 16 bits each. Written information has a retention period of at least 10 years. *Figure 2* shows the interface between COP888CG and NMC9306.

Any sequentially accessible memory device that is compatible with the MICROWIRE/PLUSTM serial communication protocol can be used as a Data Storage Unit. The Central Controlling Unit checks for the availability of memory and informs the user of the same if memory is full. Upon receipt of memory full prompt, the user can decide to overwrite or replace the memory device.

Display Terminal Unit

The Display Terminal Unit interfaces with the COP888CG through the full-duplex, double buffered UART. The COP888CG is interrupted by the terminal and the microcontroller decodes the ASCII character sent and services the corresponding request. The terminal keyboard can be used to program the telephone number to be dialed by the Auto-Dialer Unit. The real time clock is displayed on the terminal screen. The user can request the Central Controlling Unit to display the history of events monitored by the AMS System. The Central Controlling Unit retrieves the information from the Date Storage Unit and displays it on the screen.

The ASM System utilized a Visual 550 terminal. The terminal employs two independent display memories: alphanumeric and graphics. The alphanumeric functions of the V550 is ANSI X3.64 compatible and the graphics functions are fully compatible with Tectronix Plot 10® software.

With slight modification of the Central Controlling Unit's algorithm it is possible to make the ASM System interface with any other terminal unit.

LED Display Unit

The LED Display Unit is used to display the time and date information. *Figure 2* shows the interface between COP888CG and the Display Terminal Unit. The COP888CG communicates with this unit serially using the MICROWIRE/PLUS protocol.

The NSM4000A LED Display with Driver is used in the ASM System. The NSM4000A is a 4-digit 0.3" height LED display with serial data-in parallel data-out LED driver designed to operate with minimal interface to the data source. The Cen-

tral Controlling Unit does not update the display when it is servicing the Receiver Unit. The APS System has a toggle switch that enables toggling the display between Hours-Minutes to Seconds-1/80th of Seconds. The Keypad Unit is used to toggle the display between time and date.

Central Controlling Unit

This is the main unit in the application and is responsible for the efficient operation of the various units in the ASM System. The unit consists of COP888CG and the application software. The next section describes the application software in detail. The COP888CG interfaces with the various units described in the previous sections (*Figure 2*).

The application is a real time system and is totally interrupt driven with some of the tasks being executed in the background. The various units that interface with the COP888CG can be considered as tasks and the Central Controlling Unit executes these tasks based on their priority and the sequence of occurrence. The real time clock counter is given the highest priority. The Receiver Unit uses the Multi-Input Wakeup/Interrupt feature of the COP888CG to wakeup the microcontroller and service the Alarm routine. The Display Unit has a display toggle switch which also uses the Multi-Input Wakeup/Interrupt to toggle the display between Hours-Minutes and Seconds-1/80th of Seconds.

The COP888CG communicates with the Terminal Unit through the on-board, full duplex, double buffered UART. The terminal keyboard can be used to interrupt the COP888CG to program the phone number to dial in case of an emergency. The COP888CG uses the MICROWIRE/PLUSTM serial communication protocol to display the time and date information on the LED display and also to store real time information of events in the non-volatile data storage unit. Thus the MICROWIRE/PLUS protocol is time shared between the Display Unit and Data Storage Unit.

The Keypad Unit is a 4 x 4 array of keys and the COP888CG periodically polls the keypad. The input/output ports of the COP888CG is used to read the key pressed and is decoded by the software. The Auto-Dialer Unit is driven by the input/output lines and the interface between COP888CG. This unit is activated by the COP888CG 10 seconds after the Receiver Unit interrupts the microcontroller. This delay is used to disarm the Alarm routine.

SOFTWARE DESCRIPTION

The instruction set of the COP800 family of microcontrollers provide easy optimization of program size and throughput efficiency. Most of the instructions of the COP800 family are single-byte, single-cycle instructions (approximately 60%). The COP800 family of microcontrollers has three memory mapped registers (B, X and SP). The B and X registers can be used as data store memory pointers for register indirect addressing with optional auto post incrementing or decrementing of the associated pointer. This allows greater efficiency in cycle time and program code. The COP800 family allows true bit-manipulation i.e., the ability to set, reset or test any individual bit in data memory including the memory mapped I/O ports.

The architecture of COP800 family is based on a modified Harvard type architecture, where the Control Store Program (in ROM) is separated from the Data Store Memory (in RAM). Both types of memory have their own separate addressing space and separate address busses. This architecture allows the overlap of ROM and RAM memory accesses which is not possible with single-address bus Von Neumann-style architecture. The modified Harvard architecture allows access to ROM data tables which is not possible with the classical Harvard architecture.

The COP888 sub-family of microcontrollers support a total of sixteen vectored interrupts, of which fourteen are maskable interrupts and two high-priority, non-maskable interrupts. A 2-byte interrupt vector is reserved for each of these sixteen interrupts and they are stored in a user-defined 32-byte program memory (ROM) table. Please refer to the COP888 users manual or the Microcontrollers Databook for more detailed information on interrupts.

The MIWU feature, which utilizes the Port L, of the COP888 sub-family can be used to wakeup the microcontroller from the two power saving modes, i.e., HALT or IDLE modes. Alternately, the MIWU/Interrupt allows the user to generate eight additional edge selectable external interrupts. Three 8-bit memory mapped registers (WKEDG, WKEN and WKPND) are used to implement the MIWU/Interrupt. The three control registers each contain an associated pin for each L port pin. The WKEN register is used to select which particular Port L inputs will be used. The user can select whether the trigger condition on a selected L port pin is to be a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made through the WKEDG register. The occurrence of the selected trigger condition for MIWU/Interrupt is latched into the associated bit of the Wakeup Pending Register (WKPND).

The COP800 family has the ability to detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc. Reading an undefined ROM location gets zeroes, which results in a non-maskable software interrupt thus signalling an illegal condition has occurred. In addition to this, the COP888 sub-family supports both WATCHDOG™ and Clock Monitor. The WATCHDOG™ is used to monitor the number of instruction cycles between WATCHDOG™ services in order to avoid runaway programs or infinite loops. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate. These features of the COP800 family provide easy implementation of real time applications where the proper execution of the software plays a crucial role.

The major features of the software written for the ASM System implementation are described on the flow chart *Figure 3*. The main program flow is to detect the flags set, service the flags and scan the Keypad. The rest of the software is interrupt driven. The program is real time and the interrupts are serviced as and when they occur. Some of the routines are running in the background all the time, such as, Time Keeping Routine and Keypad Scan Routine. *Figures 4 and 5* gives the flow of the various interrupt service routines. The following sub-sections briefly describe each module of software connected to the units described earlier.

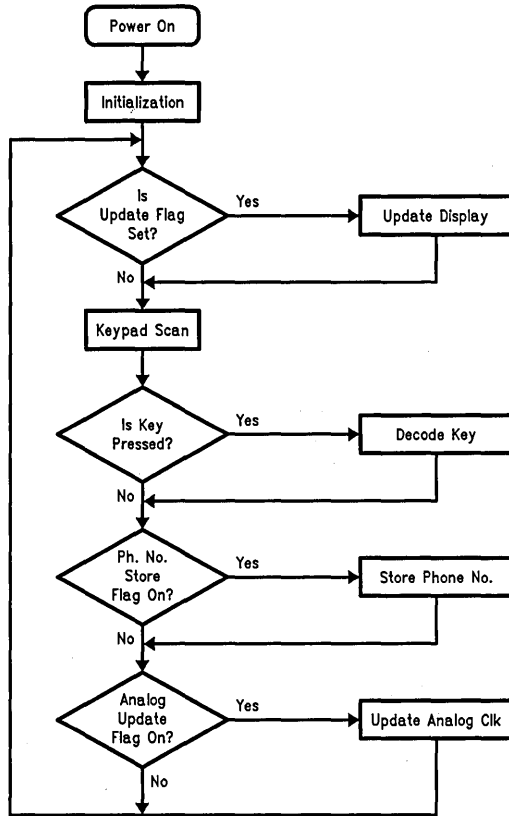


FIGURE 3. ASM System Program Flow

TL/DD/10607-3

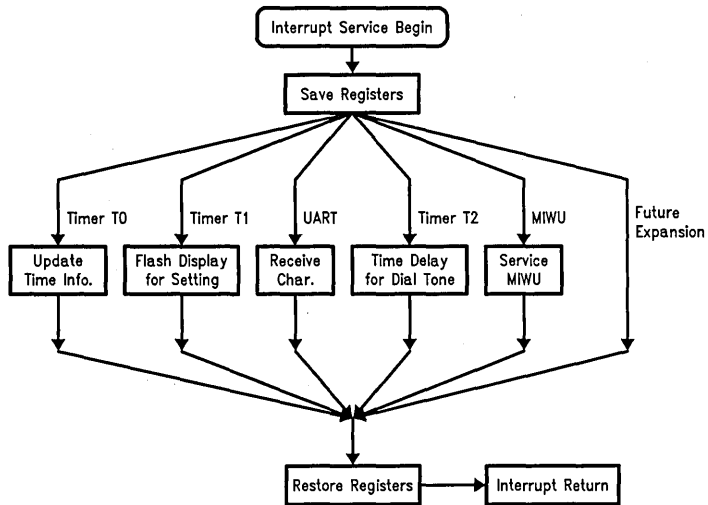
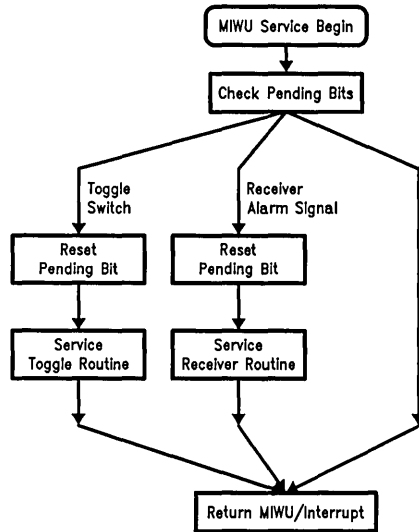


FIGURE 4. Interrupt Service Routines Flow

TL/DD/10607-4



TL/DD/10607-5

FIGURE 5. Multi-Input Wakeup/Interrupt Service Routines

Initialization Routine

The Initialization Routine loads the Data Memory locations being used in the program with default values and initializes the various control and configuration registers. It also brings up the display on the Terminal Unit and the LED Display Unit.

Time Keeping Routine

The Time Keeping Routine is the most important routine and is executed irrespective of the other modules being executed. The program uses the IDLE Timer T0 for this purpose. The IDLE Timer is a 16-bit timer and runs continuously at a fixed rate of the instruction cycle clock. The IDLE Timer counter is not memory mapped and consequently, the user cannot read or write to it. The toggling of the twelfth bit of the IDLE counter can be programmed to generate an interrupt. This interrupt is generated every 4 ms at the maximum instruction cycle clock rate of 1 MHz. The software uses this interrupt to update counters in Data Memory for time keeping. The Time Keeping routine then sets a flag to update the display which is then used by the main program.

LED Display Routine

The COP888CG uses the MICROWIRE/PLUS to interface with NSM4000 LED Display with Driver. The time and date information is displayed on the 4-digit LED display. The user is provided with a toggle switch connected to MIWU/Interrupt feature of the COP888CG to toggle the display between Hours-Minutes and Seconds-1/80th of Seconds. The toggle switch is connected to L port pin 5. Upon receipt of the MIWU/Interrupt of L port pin 5 this routine toggles the display. This routine upon receipt of the date display request through the Keypad Unit responds by switching the LED Display to show the date. The toggle switch could be used to change the display back to time. However, the display changes to time after a minute by default.

Keypad Scan Routine

This module scans the 4 x 4 matrix keyboard connected to Port D (D1-D4) as rows and to Port I (I0-I3) as columns. Thus each key in the matrix is associated with one Port D line and one Port I line. Each row in the matrix is addressed in sequence and the key closure is sensed by testing the Port I lines. The moment one key closure is detected the program jumps to load the debounce counter. The keypad scan is stopped at that particular row and the program returns to its main flow. The keypad is again scanned and when the debounce counter is decremented. When the debounce counter is zero the key pressed is accepted and decoded. The versatility of the COP888 family of instructions set allows decoding the key pressed with one instruction. The Port D (lines D1-D4) and Port I (lines I0-I3) in conjunction form an eight bit number that is unique to each key. The JID (Jump Indirect) instruction uses the contents of the accumulator to point to the indirect vector table of program address. The accumulator contents are transferred to the program counter (lower 8 bits). The data accessed from the program memory location addressed by program counter is transferred to the program counter (lower 8 bits). The JID instruction is a single-byte, three cycle instruction and provides an efficient way to decode and branch to service the appropriate routine based upon the key pressed.

The Keypad is used to set the time and data information after power up and can also be used to program the phone number to be dialed by the Auto-Dialing Unit.

Non-Volatile Data Storage Routine

The COP888CG interfaces with NM93C06A in the ASM System to store the real time data of the events monitored and also the telephone number to be dialed by the Auto-Dialer Unit. This routine is executed whenever the Receiver Unit detects a signal and the ASM System is not disarmed within 10 seconds of detection of the signal or when the

Display Terminal Unit programs the telephone number to be dialed. The Keypad can also be used to program the phone number to be dialed by the Auto-Dialer Unit. The Terminal Unit can request for the history of events, during which the COP888CG reads the NM93C06A. Please refer to the application note on MICROWIRE/PLUS for details regarding the interface between COP888CG and NMC9306.

Display Terminal Interface Routine

The Display Terminal as previously mentioned interfaces with the COP888CG through the full-duplex, double buffered UART. The terminal is used to display the history of events, real time, and sequence of operations upon detection of signal by the Receiver Unit.

The request for display of events and programming the phone number interrupts the COP888CG. However, the Time Keeping Routine updates the LED display and terminal with real time periodically, except when the COP888CG is servicing the Receiver Unit.

The operation mode of the UART may be selected in conjunction with both a prescaler and baud rate register. Character data lengths of seven, eight or nine bits are program selectable, in conjunction with a start bit, an optional parity bit, and stop bits of $\frac{7}{8}$, 1, 1 and $\frac{7}{8}$, or 2. The UART also contains a full set of error detection circuitry and a diagnostic test capability, as well as an ATTENTION mode to facilitate networking with other processors.

Please refer to the Users Manual or Microcontroller Data-book for details.

In the ASM System the COP888CG interfaces with the V550 terminal at 2400 baud, 8 data bits, 1 Stop bit, no parity. The receiver buffer full and transmit buffer empty generates an interrupt. The Port L (pins L1, L2, L3) are used for the UART interface as CKX (clock), TDX (transmit) and RDX (receive), respectively.

The display terminal is used to display time both in analog and digital form. The V550 allows interfacing both in alphanumeric and graphic modes with separate memory for each of the modes. The COP888CG is programmed to send out the ASCII ESC sequence required to generate the graphics on the screen.

Auto-Dialing Routine

This routine is responsible for dialing the number in the event of an emergency. The COP888CG interfaces with TP5088, which in turn interfaces with TP5700A. The COP888CG activates the relay that keeps the telephone line on-hook to the off-hook position. After this it times out to get the dial tone. After a fixed amount of time, the digit to be dialed is sent out on the D port, lines D1-D4, to TP5088 along with the Chip Select. The TP5088 generates the DTMF signal for the digit. The COP888CG takes care of the timing required between two digits and also the on-time of the DTMF signal for each digit. The output of the DTMF signal goes to the TP5700A which interfaces with the Tip and Ring of the telephone lines. The TP5700A receives the signal from the telephone lines and LM567 along with the associated circuitry is used to detect whether the required frequency signal was sent by the unit responding to the telephone. The output of the LM567 is connected to Port I pin 5. The Receiver Routine polls the Port I pin 5 periodically to check for response from the unit dialed by the Auto-Dialer Unit.

Receiver Routine

This is the main interrupt service routine of the ASM System. The Receiver Unit interfaces with the COP888CG

through the L port pin 4. Upon receipt of the signal from the Sensors and Transmitter Unit the Receiver Unit activates a relay which causes a MIWU/Interrupt. The interrupt service routine then waits for 10 seconds before reacting to the signal. This time is allowed to disarm the Security/Monitoring System. The Time Keeping Routine is used to calculate the delay and if the user disarms the System by toggling a switch the signal is ignored. Otherwise the Non-Volatile Storage Routine is executed to read the telephone number and this information is passed on to the Auto-Dialer Unit. The Auto-Dialer Unit dials the number and looks for a response over the telephone line. If however, there is no response, the Receiver Routine times out after a minute and tries the same number again. The number of trials can be modified in software and the time out period can also be changed. In the ASM System the number of trials is two. With slight modification the Auto-Dialer Unit can be made to dial a different number during the second attempt. The real time and date of occurrence of the event is stored in the NMC9306 along with the outcome of the telephone call. This routine keeps track of the non-volatile memory capacity and if it overflows, it prompts the user on the terminal of the same. The user is given the choice to overwrite the non-volatile memory or replace the device.

USING THE ASM SYSTEM

The ASM System upon installation and initial power-up has some preliminary steps to be performed. The time and date should be set, the phone number to be dialed by the Auto-Dialer Unit should be programmed. The toggle switch could be used to toggle the display between Hours-Minutes and Seconds-1/80th of Seconds.

Setting Time and Date

The steps involved in setting the time and date are:

1. Press key A on the keypad. The LED display flashes.
2. Set the desired time (Hours and Minutes) using the keypad.
3. The LED display and the Terminal Screen displays the time set.
4. Press key C on the keypad. The display toggles and displays the date.
5. Press key A on the keypad. The LED display begins to flash.
6. Set the date (month and day) using the keypad.
7. The LED display now shows the date set.
8. The LED display could be toggled to show the time using the toggle switch. However, the system after one minute will default to display time.

Programming the Phone Number

The phone number to be dialed could be programmed in two ways, i.e., using the terminal or the keypad. Using the terminal, the steps to be performed are:

1. Press CNTRL B on the terminal keyboard. The COP888CG sends a carriage return to terminal.
2. Press CNTRL D on the terminal keyboard. Then type the number to be dialed. At the end press CNTRL C to end programming.

Using the keypad, perform the following steps:

1. Press "*" key on the keypad.
2. Press the digits to be dialed.

3. Press “#” key on the keypad to end programming the number.

The ASM System is now ready to start monitoring. Upon receipt of the alarm signal from the Receiving Unit the ASM System will dial the number programmed. In order to display the history of events on the terminal screen press CNTRL S from the terminal keyboard.

CONCLUSIONS

The architecture, features and flexibility of the COP800 family of microcontrollers makes it cost-effective as the work-

horse of any system by eliminating external components from the circuit. This approach not only reduces the system cost and development time, but also increases the flexibility and market life of the product.

The Automated Security/Monitoring System implemented using the COP888CG illustrates a single chip system solution. The application also illustrates interfacing the COP888CG to a number of specialized peripherals using an absolute minimum number of I/O lines. The ASM System approximately uses 3k bytes of program memory (ROM) space and demonstrates an efficient method of handling multi-sourced interrupts.

Sound Effects for the COP800 Family

National Semiconductor
Application Note 663
Jerry Leventer



This application note describes the creation of sound effects using National Semiconductor's COP800 family of microcontrollers. The following applications are described in detail:

1. Whistle
2. White Noise
3. Explosion
4. Bomb
5. Laser Gun

These applications were developed on a COP820C using a 20 MHz crystal and a 1 μ s instruction cycle time. By making the appropriate changes to control registers within the routines, slower clock speeds may be used. Program flow diagrams and complete source codes are included in this document.

I. WHISTLE

The whistle routine utilizes the timer underflow interrupt and employs the TIO function on pin G3. Each timer underflow causes the TIO pin to toggle. This creates a tone whose frequency remains constant as long as the timer autoreload register value remains unchanged. In order to create a descending or ascending whistle tone, the autoreload register value is increased or decreased after every thirty-two timer interrupts (FCNTR register is used to count the interrupts). When the maximum or minimum frequency has been reached, the autoreload value must be reinitialized so that the whistle frequency does not exceed the desired range.

II. WHITE NOISE

White noise is generated by using a random number generating algorithm called a RING COUNTER. One random number is extracted periodically and placed into the MICROWIRE/PLUS™ serial shift register. These bits are shifted onto the serial output (SO) pin which is wired to a transistor amplifier that drives a speaker. The serial input (SI) and serial output (SO) pins must be tied together.

The RING COUNTER is a pseudo-random number generator which operates on the principle of a linear feedback shift register (see *Figure 1*). This shift register is not to be confused with the MICROWIRE/PLUS serial shift register. Rather it is created using two bytes of data memory (RAM), and the carry flag. Each bit is called a "stage" with the carry flag being "stage 1" and bit 0 of the two byte data register being "stage 17". Using a seventeen stage shift register results in a clean tone with little distortion.

Implementation of the ring counter shift register is accomplished by a rotate right with carry instruction (RRC A). The linear feedback function is accomplished using an "exclusive or" on stages fourteen and seventeen. This particular choice of feedback stages results in a complete cycle of bit combinations, ($2^{17} - 1$), as long as the loop does not begin with zero in the RINGVAL register.

The "exclusive or" function is not explicit in that the XOR instruction is not used. Rather, stages seventeen and fourteen are tested in software using the principle that if only one of them is set then the result is a logic one, otherwise the result is logic zero. It turns out that since the rotate occurs prior to the test, the actual bits tested are the carry flag (stage 1) and bit 2 (stage 15).

A short example using four bits can be used to demonstrate how the ring counter works (see *Figure 2*). If you perform the "exclusive or" on stages three and four, then a complete cycle results. If instead, you use stages two and four, two cycles of six and one cycle of three results depending on the bit combination you begin with.

III. EXPLOSION

The explosion sound effect is generated by manipulating the white noise algorithm to begin with a high pitch and progress to a lower pitch. This is done by altering the rate (contained in the register LUPREG) at which the random numbers are extracted from the ring counter before being placed into the MICROWIRE/PLUS serial shift register (SIOR). If for example LUPREG initially contains the value 4, the white noise will be at a high pitch. By incrementing this number after every ten timer interrupts (using the register TCNTR) the white noise pitch will be reduced. Several other registers are used to provide control of strategic portions of sound within the routine. First and last tones are controlled with FIRSTR and LASTR. The value in EXITR is used to control the overall length of the explosion and the length of each tone is controlled by the register TCNTR. To vary the white noise pitch, the register LUPCNT is used. The value in LUPCNT is incremented each time the pitch of the white noise is decreased within the timer interrupt routine. Prior to entering the ring count loop, LUPCNT is loaded into LUPREG. The serial input (SI) pin must be tied to the serial output (SO) pin.

IV. BOMB

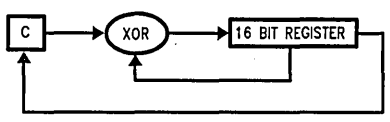
The bomb sound effect combines the descending whistle with an explosion at the end. The TIMER I/O (TIO) and serial input (SI) pins must be tied to the serial output (SO) pin. The explosion portion of this routine was altered slightly in that the first tone control register (FIRSTR) was removed. The first initialization of TCNTR, the tone control register, provides a means to control the first tone length. Subsequent tones are controlled (at label NF2 in the timer interrupt routine) where TCNTR is reinitialized. Both versions were retained for comparison and in the event that greater control of the first tone is needed.

V. LASER GUN

The laser gun sound effect combines the output from the white noise routine and the COP800 timer I/O (TIO) pin (tie TIO to SO). The SI pin is not tied to SO in this application and the ring counter uses only nine stages instead of seventeen.

The registers used for program control are EXITR, TCNTR, and the TIMER. By adjusting the value in EXITR the duration of the laser "shot" can be shortened or lengthened. (A value larger than 03F hex may create problems.) By adjusting the TIMER values (TVALO, TVALHI) and the tone counter (TCNTR) value, interesting variations in the laser sound can be attained.

NOTE: This note applies to all routines that use both the timer interrupt and the ring counter: In order to return to the main program from which the subroutine was called, the stack pointer must be manually restored during the timer interrupt before executing the return (RET) instruction. The reason for this is that the timer interrupt is two levels below the main program. A simple return statement will only serve to return to the ring counter routine from the point at which the timer interrupt occurred. By adding two to the stack pointer (SP + 2), the return statement will force the address of the instruction following the JSR in MAIN into the program counter (PC) from which point execution will continue.



TL/DD/10716-1

FIGURE 1. 17 Stage Ring Counter

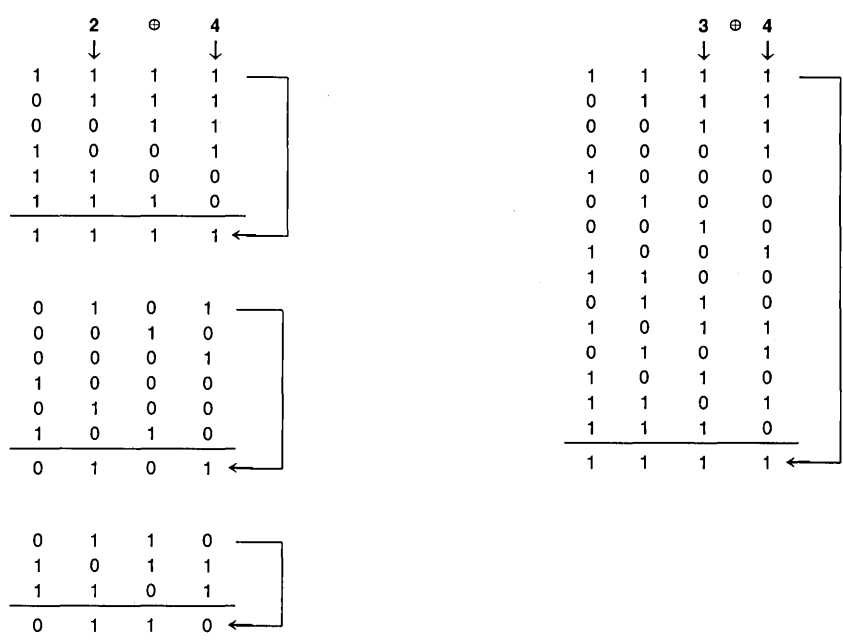
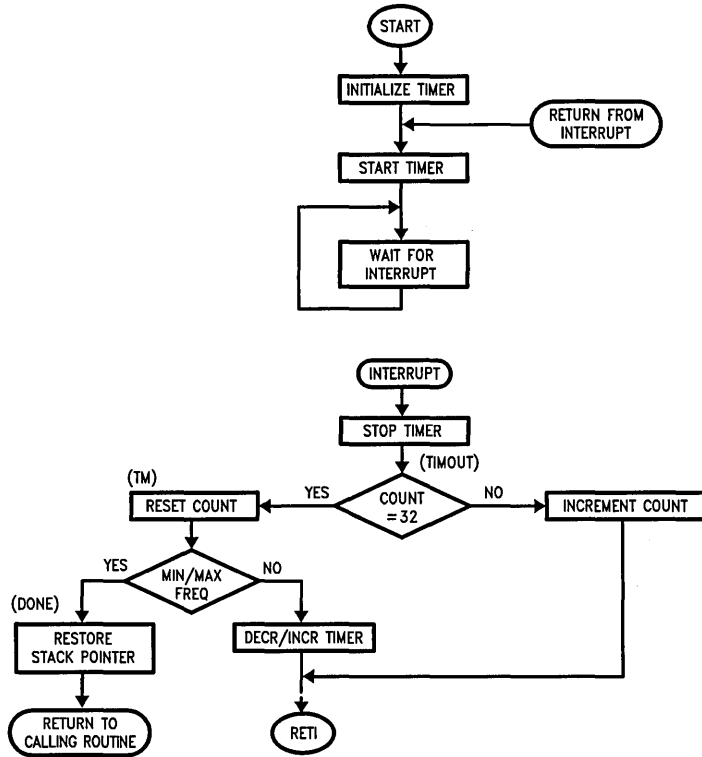


FIGURE 2. Example Showing Possible Cycles from a 4 Stage Ring Counter

Whistle Flow Diagram



TL/DD/10716-2

Descending Whistle

```

1      ;
2      ;
3      ; TIMER INTERRUPT IS USED.
4      ; OUTPUT ON TIMER I/O (TIO) PIN.
5      ; USE 20 MHz XTAL, 1 μs INSTR CYCLE FOR THIS DEMO.
6      ;
7      ; WRITTEN BY: JERRY LEVENTER
8      ; DATE:      OCTOBER 4, 1989
9      ;
10     .TITLE WHISTLE1
11     .CHIP 820
12     ;
13     00D5      PORTGC = OD5      ; PORT G CONFIGURATION
14     00E9      SIOR   = OE9      ; SIO SHIFT REGISTER
15     00EA      TMRLO  = OEA      ; TIMER LOW BYTE
16     00EB      TMRHI  = OEB      ; TIMER HIGH BYTE
17     00EC      TAULO  = OEC      ; TIMER REGISTER LOW BYTE
18     00ED      TAUHI  = OED      ; TIMER REGISTER HIGH BYTE
19     00EE      CNTRL  = OEE      ; CONTROL REGISTER
20     00EF      PSW    = OEF      ; PSW REGISTER
21     0004      TRUN   = 4
22     0005      TPND   = 5
23     0002      BUSY   = 2
24     0000      GIE    = 0
25     ;
26     ; **** SPECIAL REGISTERS AND CONSTANTS ****
27     ;
28     002F      WSLO   = 02F      ; TIMER VALUES
29     0000      WSLHI  = 000
30     00F0      FCNTR  = OF0      ; FREQUENCY COUNT REGISTER
31     0000      FCNT   = 000
32     00FF      MINREQ = OFF      ; MIN FREQUENCY CONSTANT
33     ;
34     ; *****
35     ; *** BEGIN DEMO PROGRAM HERE ***
36     ; *****
37     ;
38     0000 DD2F      MAIN: LD      SP,#02F      DEFAULT INITIALIZATION OF SP
39     0002 3005      JSR      WHISTLE      **CALLING ROUTINE FOR DEMO**
40     0004 FF        JP      .
41     0005 BCD508    WHISTLE:LD      PORTGC,#008  TIQ PIN (G3) AS OUTPUT
42     0008 BCEEA2    LD      CNTRL,#0A2      PWM WITH TIO TOGGLE, 8Tc
43     000B BCEA2F    LD      TMRLO,#WSLO    WHISTLE VALUE FOR TIMER
44     000E BCEB00    LD      TMRHI,#WSLHI
45     0011 BCEC2F    LD      TAULO,#WSLO
46     0014 BCED00    LD      TAUHI,#WSLHI
47     0017 D000      LD      FCNTR,#FCNT    INIT FREQ COUNT
48     0019 BCEF11    LUP:  LD      PSW,#011    ENFI, GIE = 1, TPND = 0
49     001C BDEE7C    SBIT     TRUN,CNTRL    START TIMER
50     001F FF        JP      .              SELF LOOP TIL TIMER INTERRUPT
51     0020 F8        JP      LUP           RUN TIL LAST HISTLE FREQ
52     ;
53     ; **** INTERRUPT ROUTINE ****
54     ;
55     00FF      .=-OFF
56     00FF BDEF75    IFBIT   TPND, PSW      TEST TIMER PENDING FLAG
57     0102 01      JP      TIMOUT
58     0103 FF      JP      .              ERROR

```


Descending Whistle (Continued)

```

59 0104 BDEE6C      TIMEOUT: RBIT   TRUN,CNTRL      ; STOP THE TIMER
60 0107 BDF075      IFBIT    5,FCNTR      ; COUNT CYCLES
61 010A 06          JP        TM
62 010B 9DF0        LD        A,FCNTR      ; INCREMENT COUNT
63 010D 8A          INC        A
64 010E 9CF0        X         A,FCNTR
65 0110 8D          RETSK
66 0111 D000        TM:      LD        FCNTR,#FCNT      ; RESET COUNT
67 0113 DEEC        LD        B,#TAULO
68 0115 AE          LD        A,[B]          ; CHANGE FREQUENCY
69 0116 92FF        IFEQ    A,#MINFREQ      ; TIMER = MIN FREQ?
70 0118 03          JP        DONE          ; YES
71 0119 8A          INC        A
72 011A A6          X         A,[B]          ; STORE FREQ IN AUTO RELOAD
73 011B 8D          RETSK
74 011C 9DFD        DONE:   LD        A,SP          ; *** RESTORE STACK POINTER ***
75 011E 9402        ADD     A,#002          ; *** AND RETURN TO CALLING ***
76 0120 9CFD        X         A,SP          ; *** ROUTINE.          ***
77 0122 8E          RET
78                   .END

```

Ascending Whistle

```

1      ;
2      ;
3      ; OUTPUT ON TIMER I/O (TIO) PIN.
4      ; USES TIMER INTERRUPT.
5      ; USE 20 MHz XTAL, 1 µs INSTR CYCLE FOR THIS DEMO.
6      ;
7      ; WRITTEN BY: JERRY LEVENTER
8      ; DATE:      OCTOBER 4, 1989
9      ;
10     ;
11     ; .TITLE WHISTLE2
12     ; .CHIP 820
13     ;
14     00D5     PORTGC = 0D5      ; PORT G CONFIGURATION
15     00EA     TMRLO  = 0EA      ; TIMER LOW BYTE
16     00EB     TMRHI  = 0EB      ; TIMER HIGH BYTE
17     00EC     TAULO  = 0EC      ; TIMER REGISTER LOW BYTE
18     00ED     TAUHI  = 0ED      ; TIMER REGISTER HIGH BYTE
19     00EE     CNTRL  = 0EE      ; CONTROL REGISTER
20     00EF     PSW    = 0EF      ; PSW REGISTER
21     0004     TRUN   = 4
22     0005     TPNL   = 5
23     0002     BUSY   = 2
24     0000     GIE    = 0
25     ;
26     ; **** SPECIAL REGISTERS AND CONSTANTS ****
27     ;
28     00FF     WSLO   = OFF      ; TIMER VALUES
29     0001     WSLHI  = 001
30     000A     MAXFREQ = 00A     ; LAST FREQUENCY CONSTANT
31     00F0     FCNTR  = 0F0     ; TIMER COUNT REGISTER
32     0010     FCNT   = 010     ; COUNTER CONSTANT
33     ;
34     ; *****
35     ; **** BEGIN PROGRAM HERE ****
36     ; *****
37     ;
38     0000 DD2F   MAIN: LD      SP,#02F      ; DEFAULT INITIALIZATION OF SP
39     0002 3005   JSR      WHISTLE2      ; *** CALLING ROUTINE FOR DEMO ***
40     0004 FF     JP       .
41     ;
42     0005 BCD508 WHISTLE2: LD      PORTGC,#008      ; TIO PIN (G3) AS OUTPUT
43     0008 BCEA0  LD      CNTRL,#0A0      ; PWM WITH TIO TOGGLE,
44     000B BCEAFF LD      TMRLO,#WSLO     ; WHISTLE VALUE FOR TIMER
45     000E BCEB01 LD      TMRHI,#WSLHI
46     0011 BCECFE LD      TAULO,#WSLO
47     0014 BCED01 LD      TAUHI,#WSLHI
48     0017 D010  LD      FCNTR,#FCNTR      ; INITIALIZE COUNTER
49     0019 BCEF11 LUP:  LD      PSW,#011      ; ENTI, GIE = 1, TPNL = 0
50     001C BDEE7C SBIT   TRUN,CNTRL      ; START TIMER
51     001F FF     JP       .              ; SELF LOOP UNTIL TIMER
52     0020 F8     JP      LUP              ; INTERRUPT

```

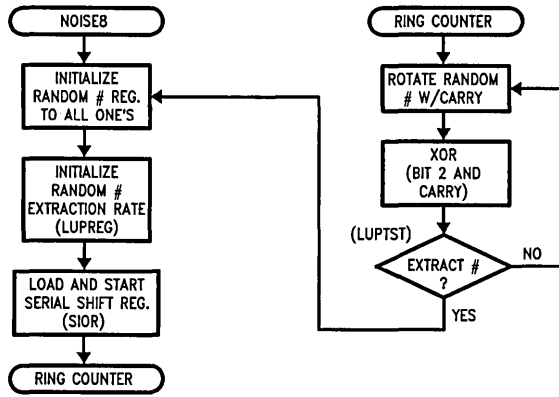
Ascending Whistle (Continued)

```

53          ;
54          ; **** INTERRUPT ROUTINE ****
55          ;
56          00FF          .=OFF
57 00FF BDEF75          IFBIT  TPND,PSW          ; TEST TIMER PENDING FLAG
58 0102 01              JP      TIMEOUT
59 0103 FF              JP      .
60 0104 BDEE6C          TIMEOUT: RBIT  TRUN,CNTRL          ; STOP THE TIMER
61 0107 BDF075          IFBIT  5,FCNTR          ; FREQUENCY TIMED OUT?
62 010A 06              JP      TM              ; YES, CHANGE FREQUENCY
63 010B 9DF0            LD      A,FCNTR          ; NO, KEEP GOING
64 010D 8A              INC     A              ; INCREMENT COUNT
65 010E 9CF0            X        A,FCNTR
66 0110 8D              RETSK
67 0111 D010            TM:    LD      FCNTR,#FCNT          ; RETURN
68 0113 9DEC            LD      A,TAULO          ; RESET COUNTER
69 0115 920A            IFEQ   A,#MAXFREQ          ; CHANGE FREQUENCY
70 0117 05              JP      DONE          ; TIMER = MAX FREQUENCY ?
71 0118 94FF            ADD     A,#OFF          ; YES
72 011A 9CEC            X        A,TAULO          ; INCREMENT FREQUENCY
73 011C 8D              RETSK          ; STORE FREQ IN AUTO RELOAD
74 011D 9DFD            DONE: LD      A,SP          ; *** RESTORE STACK POINTER ***
75 011F 9402            ADD     A,#002          ; *** AND RETURN TO CALLING ***
76 0121 9CFD            X        A,SP          ; *** ROUTINE.          ***
77 0123 8E              RET
78          .END

```

White Noise



TL/DD/10716-3

White Noise (Continued)

```

1      ;
2      ;
3      ;
4      ;
5      ; TIE SERIAL INPUT (SI) PIN TO SERIAL OUTPUT (SO) PIN.
6      ; OUTPUT IS ON THE SERIAL OUTPUT (SO) PIN.
7      ; NO INTERRUPT IS USED.
8      ; USE 20 MHz XTAL, 1 μs INSTR CYCLE FOR THIS DEMO.
9      ;
10     ; WRITTEN BY: JERRY LEVENTER
11     ; DATE:      OCTOBER 4, 1989
12     ;
13     .TITLE NOISE8
14     .CHIP 820
15     ;
16     OOD5      PORTGC = 0D5      ; PORT G CONFIGURATION
17     OOE9      SIOR  = 0E9      ; SERIAL SHIFT REGISTER
18     OOE9      TMRLO = 0EA      ; TIMER LOW BYTE
19     OOE9      TMRHI = 0EB      ; TIMER HIGH BYTE
20     OOE9      TAULO = 0EC      ; TIMER REGISTER LOW BYTE
21     OOE9      TAUHI = 0ED      ; TIMER REGISTER HIGH BYTE
22     OOE9      CNTRL = 0EE      ; CONTROL REGISTER
23     OOE9      PSW   = 0EF      ; PSW REGISTER
24     0002      BUSY  = 2        ; BUSY BIT
25     ;
26     ; **** SPECIAL REGISTERS AND CONSTANTS ****
27     ;
28     0002      RNGVAL = 002      ; RANDOM NUMBER LOCATION
29     00FF      LUPREG = OFF      ; EXTRACTION RATE REGISTER
30     0000      FLAG   = 000      ; RANDOM NUMBER BYTE FLAG
31     0004      COUNT  = 4        ; EXTRACTION RATE CONSTANT
32     ;
33     ; *****
34     ; **** BEGIN PROGRAM HERE ****
35     ; *****
36     ;
37     0000 DD2F      LD      SP,#02F      ; DEFAULT INITIALIZATION OF SP
38     0002 BCD530    NOISE: LD      PORTGC,#030 ; SO AND SK AS OUTPUTS
39     0005 BCEE8B    LD      CNTRL,#08B   ; SK = DIV BY 8, TIMER RELOAD
40     0008 A1       SC              ; INIT STAGE 1
41     0009 5D       LD      B,#RNGVAL    ; POINT TO RANDOM # LOCATION
42     000A 9AFF     LD      [B+],#OFF    ; INIT RING VAL TO ONE'S
43     000C 9EFF     LD      [B],#OFF     ; B POINTS TO UPPER BYTE
44     000E 9CE9    SHIFT: X      A,SIOR  ; PLACE # IN SIOR
45     0010 BDEF7A   SBIT      BUSY,PSW   ; START SHIFTING
46     0013 DF04    LD      LUPREG,#004   ; RESTORE EXTRACTION COUNT
47     ;

```

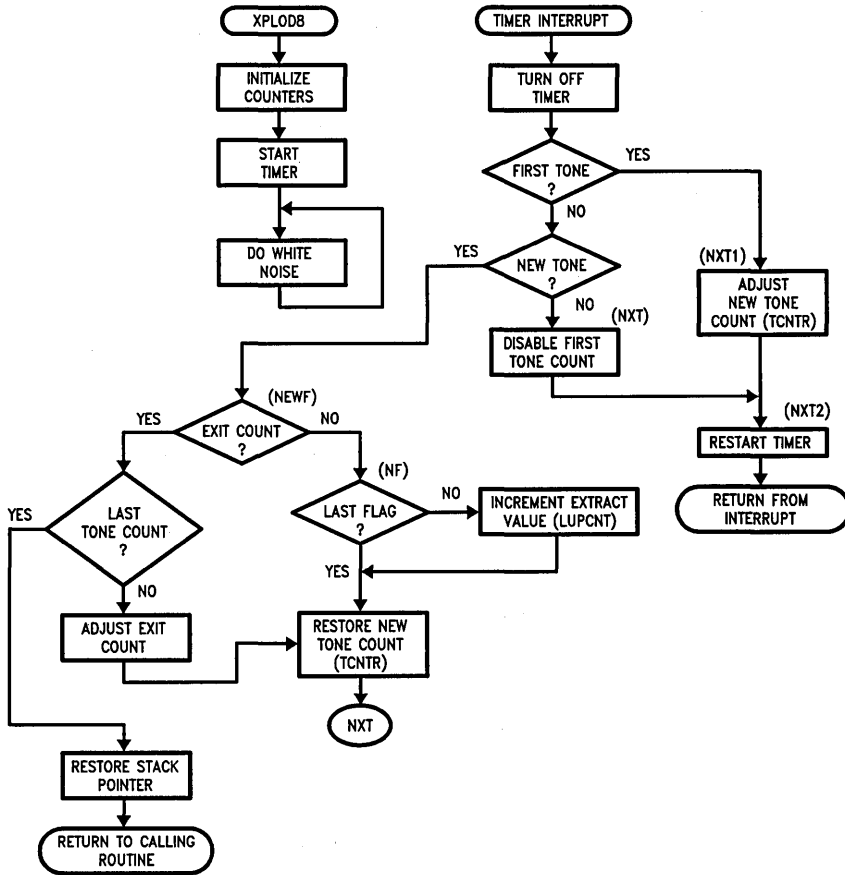
White Noise (Continued)

```

48 ; *****
49 ; RING COUNTER (17 STAGE)
50 ; THIS IS A SEVENTEEN STAGE RING COUNTER (LINEAR
51 ; FEEDBACK SHIFT REGISTER) WITH THE RRC COMMAND.
52 ; THE COUNTER'S 14TH AND 17TH STAGES THROUGH AN
53 ; EXCLUSIVE-OR SERVE AS THE FEEDBACK FUNCTION.
54 ; THIS 14, 17 RING COUNTER BREAKS DOWN INTO
55 ; 1 CYCLE OF [(2 ** 17) - 1] COUNTS. SINCE THE EXCLUSIVE OR
56 ; OCCURS AFTER THE ROTATE, IT IS THE 15TH AND CARRY
57 ; STAGES THAT ARE XOR'D (BIT 2 AND CARRY).
58 ;
59 ;
60 ;
61 ;
62 ;
63 ;
64 ;
65 ;
66 ;
67 ;
68 0015 AE RING: LD A,[B] ; GET RANDOM #
69 0016 B0 RRC A ; ROTATE UPPER BYTE
70 0017 A3 X A,[B-]
71 0018 AE LD A,[B]
72 0019 B0 RRC A ; ROTATE LOWER BYTE
73 001A A6 X A,[B]
74 001B 9804 LD A,#004 ; PERFORM XOR
75 001D 85 AND A,[B]
76 001E 9200 IFEQ A,#000
77 0020 05 JP LUPTST
78 0021 88 IFC
79 0022 02 JP RC
80 0023 A1 SC
81 0024 01 JP LUPTST
82 0025 A0 RC: RC
83 0026 AA LUPTST: LD A,[B+] ; POINT TO UPPER BYTE
84 0027 CF DRSZ LUPREG ; EXTRACT THIS NUMBER ?
85 0028 EC JP RING ; NO, KEEP ROTATING
86 0029 E4 JP SHIFT ; YES, SEND IT
87 .END

```

Explosion



TL/DD/10716-4

Explosion (Continued)

```

1          ;
2          ;
3          ; TIMER INTERRUPT IS USED.
4          ; SI MUST BE TIED TO SO. OUTPUT ON SO.
5          ; USE 20 MHz XTAL, 1 μs INSTR CYCLE FOR THIS DEMO.
6          ;
7          ; WRITTEN BY: JERRY LEVENTER
8          ; DATE:      OCTOBER 4, 1989
9          ;
10         .TITLE XPLD8
11         .CHIP 820
12         ;
13         00D5      PORTGC = 0D5          ; PORT G CONFIGURATION
14         00E9      SIOR   = 0E9          ; SIO SHIFT REGISTER
15         00EA      TMRLO  = 0EA          ; TIMER LOW BYTE
16         00EB      TMRHI  = 0EB          ; TIMER HIGH BYTE
17         00EC      TAULO  = 0EC          ; TIMER REGISTER LOW BYTE
18         00ED      TAUHI  = 0ED          ; TIMER REGISTER HIGH BYTE
19         00EE      CNTRL  = 0EE          ; CONTROL REGISTER
20         00EF      PSW    = 0EF          ; PSW REGISTER
21         0004      TRUN   = 4
22         0005      TPNL   = 5
23         0002      BUSY   = 2
24         ;
25         ; **** SPECIAL REGISTERS AND CONSTANTS ****
26         ;
27         ; ANY REGISTER USED FOR THE DRSZ TEST MUST
28         ; BE INITIALIZED TO AT LEAST "1".
29         ;
30         00F5      FIRSTR = 0F5          ; FIRST TONE CONTROL REGISTER
31         0002      FIRST  = 002          ; FIRST TONE CONSTANT
32         00F6      LASTR  = 0F6          ; LAST TONE CONTROL REGISTER
33         0002      LAST   = 002          ; LAST TONE CONSTANT
34         00F7      EXITR  = 0F7          ; ROUTINE DURATION REGISTER
35         0010      EXIT   = 010          ; EXIT CONSTANT
36         0002      RNGVAL = 002          ; HOLDS CURRENT RANDOM #
37         00F8      TCNTR  = 0F8          ; TONE DURATION REGISTER
38         000A      TCNT   = 0A           ; TONE CONSTANT
39         0020      TCNT1  = 020          ; "FIRST" TONE CONSTANT
40         00F9      LUPREG = 0F9          ; EXTRACTION RATE REGISTER
41         0004      XTRCT  = 004          ; EXTRACT CONSTANT
42         00FA      LUPCNT = 0FA          ; EXTRACTION VARIABLE REGISTER
43         0000      TEMP   = 000          ; LAST TONE FLAG
44         00FF      TVALO  = 0FF          ; TIMER VALUES
45         0010      TVALHI = 010
46         ;
47         ; *****
48         ; **** BEGIN PROGRAM HERE ****
49         ; *****
50         ;
51 0000 DD2F      MAIN: LD      SP,#02F      DEFAULT INITIALIZATION OF SP
52 0002 3005      JSR      XPLD            **** XPLD CALLING ROUTINE ****
53 0004 FF        JP       .              **** SELF LOOP FOR DEMO ****
54 0005 BCD530    XPLD: LD      PORTGC,#030
55 0008 BCEE8A    LD      CNTRL,#08A      SK;= DIV BY 8, PWM ON
56 000B BCEF11    LD      PSW,#011      ENABE TIMER INTERRUPT
57 000E BCEAFF    LD      TMRLO,#TVALO   INITIALIZE TIMER
58 0011 BCEB10    LD      TMRHI,#TVALHI
59 0014 BCECFE    LD      TAULO,#TVALO
60 0017 BCED10    LD      TAUHI,#TVALHI

```


Explosion (Continued)

```

61 001A D502          LD   FIRSTR,#FIRST      ; LENGTHEN FIRST TONE
62 001C D602          LD   LASTR,#LAST        ; LENGTHEN LAST TONE
63 001E D710          LD   EXITR,#EXIT        ; INITIALIZE EXIT COUNT
64 0020 D80A          LD   TCNTR,#TCNT        ; INITIALIZE TONE COUNT
65 0022 DA04          LD   LUPCNT,#XTRCT      ; INITIALIZE EXTRACTION RATE
66 0024 BD0068        RBIT   O,TEMP          ; RESET LAST TONE FLAG
67 0027 BDEE7C        SBIT   TRUN,CNTRL       ; START TIMER
68 002A A1            NOISE: SC          ; INIT. STAGE 1
69 002B 5D            LD   B,#RNGVAL       ; POINT TO RANDOM NUMBER
70 002C 9AFF          LD   [B+],#OFF        ; INIT TO ALL ONE'S
71 002E 9EFF          LD   [B],#OFF
72 0030 9CE9        SHIFT: X          A,SIOR          ; LOAD AND START SIOR
73 0032 BDEF7A        SBIT   BUSY,PSW
74 0035 9DFA          LD   A,LUPCNT        ; RESTORE EXTRACTION COUNT
75 0037 9CF9          X          A,LUPREG
76
77 ; *****
78 ; RING COUNTER (17 STAGE)
79 ;
80 ; THIS IS A SEVENTEEN STAGE RING COUNTER (LINEAR
81 ; FEEDBACK SHIFT REGISTER) WITH THE RRC COMMAND.
82 ; THE COUNTER'S 14th AND 17th STAGES THROUGH AN
83 ; EXCLUSIVE-OR SERVE AS THE FEEDBACK FUNCTION.
84 ; THIS 14, 17 RING COUNTER BREAKS DOWN INTO
85 ; 1 CYCLE OF [(2 ** 17) - 1] COUNTS. SINCE THE EXCLUSIVE OR
86 ; OCCURS AFTER THE ROTATE, IT IS THE 15th AND CARRY
87 ; STAGES THAT ARE XOR'D (BIT 2 AND CARRY).
88 ;
89 ;
90 ;
91 ;
92 ;
93 ;
94 ;
95 ;
96 ;
97 ;
98 0039 AE          RING: LD   A,[B]          ; GET RANDOM #
99 003A B0          RRC   A              ; ROTATE UPPER BYTE
100 003B A3          X      A,[B-]
101 003C AE          LD   A,[B]
102 003D B0          RRC   A              ; ROTATE LOWER BYTE
103 003E A6          X      A,[B]
104 003F 9804        LD   A,#004          ; PERFORM XOR
105 0041 85          AND   A,[B]
106 0042 9200        IFEQ  A,#000
107 0044 05          JP    TSLUP
108 0045 88          IFC
109 0046 02          JP    RC
110 0047 A1          SC
111 0048 01          JP    TSTLUP
112 0049 A0          RC:   RC
113 004A AA          TSTLUP: LD  A,[B+]        ; POINT TO UPPER BYTE
114 004B C9          DRSZ  LUPREG        ; EXTRACT THIS # ?
115 004C EC          JP    RING          ; NO, KEEP ROTATING
116 004D AE          LD   A,[B]          ; YES
117 004E E1          JP    SHIFT

```

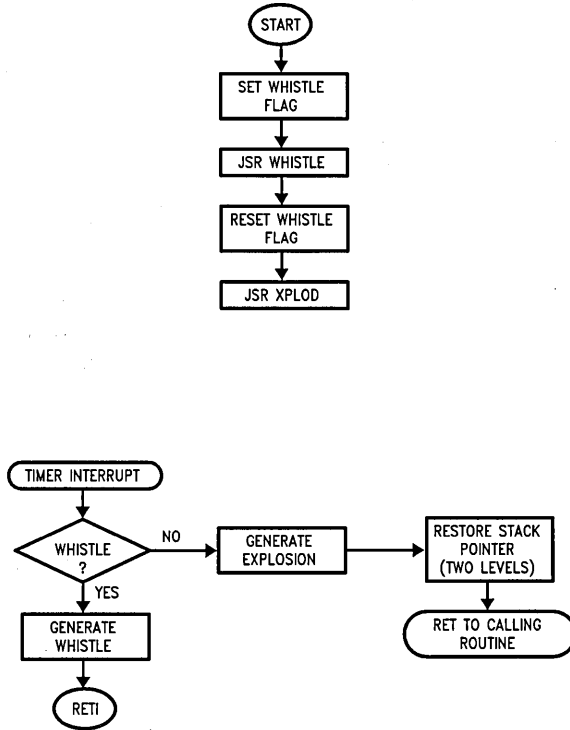
Explosion (Continued)

```

118          ;
119          ; **** TIMER INTERRUPT ROUTINE ****
120          ;
121          00FF          ;      .=      OFF
122 00FF BDEF75          IFBIT TPND,PSW          ; TEST TIMER PND FLAG
123 0102 02              JP      TMOUT
124 0103 2005            JMP      XPLOD
125 0105 BDEE6C          TMOUT: RBIT TRUN,CNTRL          ; STOP TIMER
126 0108 DEFA            LD      B,#LUPCNT
127 010A C5              DRSZ   FIRSTR          ; TEST FOR FIRST TONE
128 010B 213B            JMP      NXT1          ; AND ADJUST
129 010D C8              DRSZ   TCNTR          ; TEST FOR NEW TONE
130 010E 01              JP      NXT          ; NO
131 010F 0D              JP      NEWF
132 0110 D501            NXT:   LD      FIRSTR,#1          ; DISABLE FIRST TONE REG
133 0112 BDEF7C          NXT2:  SBIT   4,PSW          ; ENABLE TIMER INTERRUPT
134 0115 BDEF6D          RBIT   5,PSW          ; RESET TPND FLAG
135 0118 5D              LD      B,#RNGVAL          ; POINT TO RANDOM#
136 0119 BDEE7C          SBIT   TRUN,CNTRL          ; RESTART TIMER
137 011C 8F              RETI          ; RETURN
138 011D C7              NEWF:  DRSZ   EXITR          ; TEST EXIT COUNT
139 011E 10              JP      NF          ; NO
140 011F C6              DRSZ   LASTR          ; ENABLE LAST TONE
141 0120 01              JP      LST
142 0121 06              JP      NLST
143 0122 D709            LST:   LD      EXITR,#09          ; SET LAST TONE LENGTH
144 0124 BD0078          SBIT   0,TEMP          ; SET LAST TONE FLAG
145 0127 0F              JP      NF2
146 0128 9DFD            NLST:  LD      A,SP          ; *** RESTORE STACK POINTER ***
147 012A 9402            ADD    A,#002          ; *** FROM TIMER INTERRUPT ***
148 012C 9CFD            X      A,SP          ; *** AND RETURN TO MAIN ***
149 012E 8E              RET
150 012F BD0070          NF:   IFBIT  0,TEMP          ; LAST TONE ?
151 0132 04              JP      NF2          ; YES
152 0133 AE              LD      A,[B]          ; NEW TONE
153 0134 9404            NF4:  ADD    A,#04          ; INCR EXTRACTION VALUE
154 0136 A6              X      A,[B]
155 0137 D80A            NF2:  LD      TCNTR,#TCNT          ; REINITIALIZE TONE TIME
156 0139 2110            JMP      NXT
157 013B D820            NXT1: LD      TCNTR,#TCNT1          ; ADJUST FIRST TONE LENGTH
158 013D 2112            JMP      NXT2
159          .END

```

Bomb



TL/DD/10716-5

Bomb (Continued)

```

1      ;
2      ;
3      ; THE SERIAL INPUT (SI) AND TIMER I/O (TIO) PINS
4      ; MUST BE TIED TO THE SERIAL OUTPUT (SO) PIN.
5      ; OUTPUT IS ON SO.
6      ; USE 20 MHz XTAL, 1  $\mu$ s INSTR CYCLE FOR THIS DEMO.
7      ;
8      ; WRITTEN BY: JERRY LEVENTER
9      ; DATE:      OCTOBER 4, 1989
10     ;
11     ;
12     ;           .TITLE BOMB8
13     ;           .CHIP 820
14     ;
15     00D5      PORTGC = 0D5      ; PORT G CONFIGURATION
16     00E9      SIOR   = 0E9      ; SIO SHIFT REGISTER
17     00EA      TMRLO  = 0EA      ; TIMER LOW BYTE
18     00EB      TMRHI  = 0EB      ; TIMER HIGH BYTE
19     00EC      TAULO  = 0EC      ; TIMER REGISTER LOW BYTE
20     00ED      TAUHI  = 0ED      ; TIMER REGISTER HIGH BYTE
21     00EE      CNTRL  = 0EE      ; CONTROL REGISTER
22     00EF      PSW    = 0EF      ; PSW REGISTER
23     0004      TRUN   = 4
24     0005      TPND   = 5
25     0002      BUSY   = 2
26     0000      GIE    = 0
27     ;
28     ; **** EXPLOSION REGISTERS AND CONSTANTS ****
29     ;
30     ; SOME OF THE FOLLOWING REGISTERS USE THE DRSZ
31     ; TEST AND MUST THEREFORE BE INITIALIZED TO AT
32     ; LEAST "1".
33     ;
34     00F6      LASTR  = 0F6      ; CONTROL LAST TONE
35     0002      LAST   = 002      ; LAST TONE CONSTANT
36     0004      LAST2  = 004      ; EXIT CONSTANT
37     00F7      EXITR  = 0F7      ; TOTAL TIME TILL EXIT
38     0010      EXIT   = 010      ; EXIT CONSTANT
39     00F3      RNGVAL = 0F3      ; HOLDS CURRENT RING VALUE
40     00F8      TCNTR  = 0F8      ; TIME FOR EACH TONE FREQ
41     000A      TCNT   = 0A       ; CONSTANT VALUE
42     00F9      LUPREG = 0F9      ; TONE COUNT INSIDE RING
43     00FA      LUPCNT = 0FA      ; TONE COUNT OUTSIDE RING (VARIABLE)
44     0000      FLAG   = 000      ; FLAG REGISTER FOR SUBROUTINES
45     ;
46     00FF      TVALO  = 0FF      ;
47     001A      TVALHI = 01A      ;
48     ;
49     ; **** WHISTLE REGISTERS AND CONSTANTS ****
50     ;
51     002F      WSLO   = 02F      ; TIMER VALUES
52     0000      WSLHI  = 000      ;
53     00FF      MINFQ  = 0FF      ; FINAL (LOW FREQ) TIMER VALUE
54     ;
55     00F0      FCNTR  = 0F0      ; FREQUENCY COUNT REGISTER
56     0000      FCNT   = 000

```

Bomb (Continued)

```

57 ;
58 ; *****
59 MAIN:
60 0000 DD2F LD SP,#02F ; DEFAULT INITIALIZATION OF SP
61 0002 BD0078 SBIT 0,FLAG ; SET SUBROUTINE FLAG
62 ; 1 = WHISTLE
63 ; 0 = EXPLOSION
64 0005 3157 JSR WHISTLE
65 0007 BD0068 MAIN2: RBIT 0,FLAG
66 000A 300D JSR BOMB
67 000C FF JP . ; *** STOP HERE OR REPEAT ***
68 ; *****
69 ;
70 000D BCD530 BOMB: LD PORTGC,#030 ; CONFIGURE "SO" AS OUTPUT
71 0010 BCEE8A LD CNTRL,#08A ; SK = DIV BY 8, PWM ON
72 0013 BCEF11 LD PSW,#011 ; ENABLE TIMER INTERRUPT
73 0016 BCEAFF LD TMRLO,#TVALO ; INITIALIZE TIMER
74 0019 BCEB1A LD TMRHI,#TVALHI
75 001C BCECF7 LD TAULO,#TVALO
76 001F BCED1A LD TAUHI,#TVALHI
77 0022 D602 LD LASTR,#LAST ; INITIALIZE LAST TONE FLAG
78 0024 D710 LD EXITR,#EXIT ; INITIALIZE EXIT COUNT
79 0026 D80A LD TCNTR,#TCNT ; INITIALIZE TONE COUNT
80 0028 DA0A LD LUPCNT,#10 ; INITIALIZE FIRST TONE FREQUENCY
81 002A BD0069 RBIT 1,FLAG ; RESET LAST TONE FLAG BIT
82 ;
83 002D A1 NOISE: SC ;
84 002E DEF3 LD B,#RNGVAL ; POINT TO RING VALUE
85 0030 9AFF LD [B+],#OFF ; INIT TO ALL ONE'S
86 0032 9EFF LD [B],#OFF
87 0034 BDEE7C SBIT TRUN,CNTRL ; START THE TIMER
88 0037 BEFGA SHIFT: RBIT BUSY,PSW
89 003A 9CE9 X A,SIOR ; RANDOM # TO SIO
90 003C BDEF7A SBIT BUSY,PSW
91 003F 9DFA LD A,LUPCNT ; RESTORE EXTRACTION COUNT
92 0041 9CF9 X A,LUPREG
93 ;
94 ; *****
95 ; RING COUNTER (17 STAGE)
96 ;
97 ; THIS IS A SEVENTEEN STAGE RING COUNTER (LINEAR
98 ; FEEDBACK SHIFT REGISTER) WITH THE RRC COMMAND.
99 ; THE COUNTER'S 14th AND 17th STAGES THROUGH AN
100 ; EXCLUSIVE-OR SERVE AS THE FEEDBACK FUNCTION.
101 ; THIS 14, 17 RING COUNTER BREAKS DOWN INTO
102 ; 1 CYCLE OF [(2 ** 17) - 1] COUNTS. SINCE THE EXCLUSIVE OR
103 ; OCCURS AFTER THE ROTATE, IT IS THE 15th AND CARRY
104 ; STAGES THAT ARE XOR'D (BIT 2 AND CARRY).
105 ;
106 ; BEFORE ROTATE: 14 17
107 ; AFTER ROTATE: 15 CARRY
108 ;
109 ; CARRY BIT = STAGE ONE
110 ; LOW ORDER BIT = STAGE 17

```

Explosion (Continued)

```

111 ; *****
112 0043 AE RING: LD A,[B] ; GET RANDOM #
113 0044 B0 RRC A ; ROTATE UPPER BYTE
114 0045 A3 X A,[B-]
115 0046 AE LD A,[B]
116 0047 B0 RRC A ; ROTATE LOWER BYTE
117 0048 A2 X A,[B+]
118 0049 9804 LD A,#004 ; PERFORM XOR
119 004B 85 AND A,[B]
120 004C 9200 IFEQ A,#000
121 004E 05 JP TSTLUP
122 004F 88 IFC
123 0050 02 JP RC
124 0051 A1 SC
125 0052 01 JP TSTLUP
126 0053 A0 RC: RC
127 0054 C9 TSLUP: DRSZ LUPREG ; POINT TO UPPER BYTE
128 0055 ED JP RING ; EXTRACT THIS # ?
129 0056 AE LD A,[B] ; NO, KEEP ROTATING
130 0057 2037 JMP SHIFT ; YES
131 ;
132 ; **** INTERRUPT ROUTINE ****
133 ;
134 00FF . = OFF
135 00FF BDEF75 IFBIT TPND,PSW ; TEST FOR EXIT
136 0102 01 JP TMOUT
137 0103 FF JP . ; ERROR
138 ;
139 0104 BDEE6C TMOUT RBIT TRUN,CNTRL ; STOP TIMER
140 0107 BD0070 IFBIT 0,FLAG ; BRANCH TO ROUTINE
141 010A 213B JMP WSINT ; SET = WHISTLE, RESET = EXPLOSION
142 ;
143 010C DEFA LD B,#LUPCNT
144 010E C8 DRSZ TCNTR ; TEST FOR NEW TONE
145 010F 01 JP NXT ; NO, DON'T INCREMENT LUPCNT
146 0110 0C JP NEWF ; YES
147 0111 BDEF7C NXT: SBIT 4,PSW ; ENABLE TIMER INTRRUPT
148 0114 BDEF6D RBIT 5,PSW ; RESET TIMER PENDING FLAG
149 0117 DEF3 LD B,#RNGVAL ; POINT TO RANDOM #
150 0119 BDEE7C SBIT TRUN,CNTRL ; RESTART TIMER
151 011C 8F RETI ; RETURN TO RING COUNTER
152 011D C7 NEWF: DRSZ EXITR ; DO LAST TONE ?
153 011E 10 JP NF ; NO
154 011F C6 DRSZ LASTR ; IS LAST TONE DONE?
155 0120 01 JP LST ; NO
156 0121 06 JP NLST ; YES, RETURN TO MAIN
157 0122 D704 LST: LD EXITR,#LAST2 ; LENGTHEN THE LAST TONE
158 0124 BD0079 SBIT 1,FLAG ; SET LAST TONE FLAG
159 0127 0F JP NF2
160 0128 9DFD NLST: LD A,SP ; ** RESTORE STACK POINTER **
161 012A 9402 ADD A,#002 ; ** AND RETURN TO MAIN **
162 012C 9CFD X A,SP
163 012E 8E RET
164 ;
165 012F BD0071 NF: IFBIT 1,FLAG ; LAST TONE ?
166 0132 04 JP NF2 ; YES, DON'T INCREMENT LUPCNT
167 0133 AE LD A,[B] ; NEW TONE
168 0134 9404 ADD A,#04 ; INCR EXTRACT COUNT (LUPCNT)
169 0136 A6 X A,[B]
170 0137 D80A NF2: LD TCNTR,#TCNT ; REINITIALIZE TONE TIME
171 0139 2111 JMP NXT

```

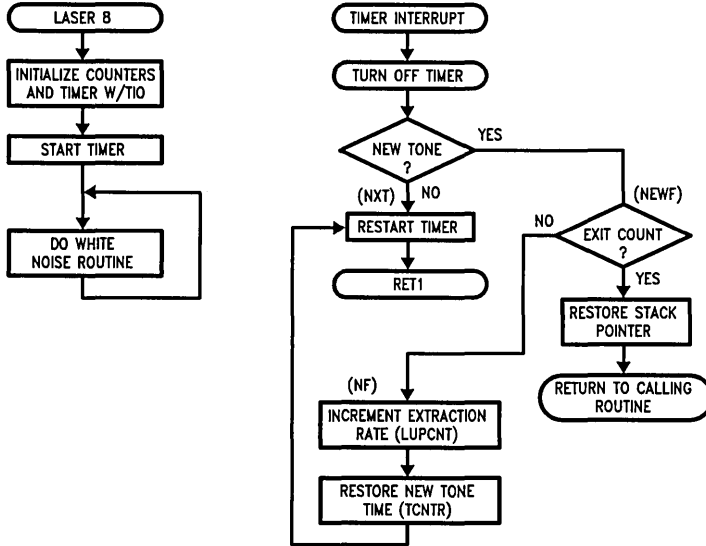
Explosion (Continued)

```

172          ; *****
173 013B BDF075 WSINT:  IFBIT  5,FCNTR  ; READY FOR NEW FREQUENCY ?
174 013E 06      JP      TM          ; YES
175 013F 9DFO    LD      A,FCNTR  ; NO, INCREMENT COUNT
176 0141 8A      INC     A
177 0142 9CFO    X        A,FCNTR
178 0144 8D      RETSK
179 0145 D000    TM:      LD      FCNTR,#FCNT  ; NO, RETURN TO WHISTLE
180 0147 DEEC    LD      B,#TAULO  ; RESET NEW FREQUENCY COUNT
181 0149 AE      LD      A,[B]    ; POINT TO AUTORELOAD REG
182 014A 92FF    IFEQ   A,#MINFQ  ; CHANGE FREQUENCY
183 014C 03      JP      DONE     ; TIMER = MIN FREQ ?
184 014D 8A      INC     A
185 014E A6      X        A,[B]    ; STORE FREQ IN AUTO RELOAD
186 014F 8D      RETSK
187 0150 9DFD    DONE:   LD      A,SP      ; ** RESTORE STACK POINTER **
188 0152 9402    ADD     A,#002    ; ** AND RETURN TO MAIN **
189 0154 9CFD    X        A,SP
190 0156 8E      RET
191          ; *****
192 0157 BCD508 WHISTLE: LD     PORTGC,#008 ; TIO PIN (G3) AS OUTPUT
193 015A BCEEA2 LD     CNTRL,#0A2  ; PWM WITH TIO TIGGLE, 8Tc
194 015D D000    LD     FCNTR,#FCNT ; INIT FREQ COUNTER
195 015F BCEA2F LD     TMRLO,#WSLO ; WHISTLE VALUE FOR TIMER
196 0162 BCEB00 LD     TMRHI,#WSLHI
197 0165 BCEC2F LD     TAULO,#WSLO
198 0168 BCED00 LD     TAUHI,#WSLHI
199
200 016B BCEF11 ; BEGIN LD     PSW,#011  ; ENTI, GIE = 1, TPND = 0
201 016E BDEE7C SBIT  TRUN,CNTRL  ; START TIMER
202 0171 FF      JP      .         ; LOOP UNTIL TIMER INTERRUPT
203 0172 F8      JP      BEGIN    ; RETURN HERE FROM INTERRUPT
204          .END

```

Laser Gun



TL/DD/10716-6

Laser Gun (Continued)

```

1      ;
2      ; TIMER INTERRUPT IS USED.
3      ; THE SERIAL OUTPUT PIN (SO) AND THE TIO PIN MUST BE
4      ; TIED TOGETHER.
5      ; OUTPUT IS ON SO AND TIO.
6      ;
7      ; TO ALTER THE DURATION OF THE LASER SHOT CHANGE THE
8      ; "EXIT" VALUE, HOWEVER, DO NOT EXCEED 03F HEX.
9      ; THE TIMER VALUES (TVALO, TVALHI) COMBINED WITH THE
10     ; TONE COUNT (TNCTR) CAN BE ADJUSTED TO ACHIEVE A
11     ; VARIETY OF SOUNDS.
12     ;
13     ; USE 20 MHz XTAL, 1 μs INSTR CYCLE TIME FOR THIS DEMO.
14     ;
15     ;
16     ; WRITTEN BY: JERRY LEVENTER
17     ; DATE:      OCTOBER 4, 1989
18     ;
19     ;
20     .TITLE LASERS
21     .CHIP 820
22     ;
23     OOD5      PORTGC = OD5      ; PORT G CONFIGURATION
24     OOE9      SIOR   = OE9      ; SIO SHIFT REGISTER
25     OOE A     TMRLO  = OEA      ; TIMER LOW BYTE
26     OOE B     TMRHI  = OEB      ; TIMER HIGH BYTE
27     OOE C     TAULO  = OEC      ; TIMER REGISTER LOW BYTE
28     OOE D     TAUHI  = OED      ; TIMER REGISTER HIGH BYTE
29     OOE E     CNTRL  = OEE      ; CONTROL REGISTER
30     OOE F     PSW    = OEF      ; PSW REGISTER
31     0004      TRUN   = 4
32     0005      TPNL   = 5
33     0002      BUSY   = 2
34     ;
35     ; **** SPECIAL REGISTERS AND COUNTERS ****
36     ; ANY REGISTER THAT IS USED FOR THE DRSZ TEST,
37     ; MUST BE INITIALIZED TO AT LEAST "1".
38     ;
39     00F7      EXITR  = 0F7      ; ROUTINE DURATION REGISTER
40     003F      EXIT   = 03F      ; EXIT CONSTANT
41     0002      RNGVAL = 002      ; HOLDS CURRENT RANDOM #
42     00F8      TCNTR  = 0F8      ; TONE DURATION REGISTER
43     0020      TCNT   = 020      ; TONE CONSTANT
44     00F9      LUPREG = 0F9      ; EXTRACTION RATE REGISTER
45     0003      XTRCT  = 003      ; EXTRACT CONSTANT
46     00FA      LUPCNT = 0FA      ; EXTRACT VARIABLE REGISTER
47     00FF      TVALO  = 0FF      ; TIMER VALUES
48     0000      TVALHI = 000
49     ;
50     ;*****
51     ;**** BEGIN PROGRAM HERE ****
52     ;*****
53     ;
54 0000      MAIN: LD     SP,#02F    DD2F; DEFAULT INITIALIZATION OF SP
55 0002      LUP:  LD     EXITR,#EXIT D73F; INITIALIZE SHOT DURATION
56 0004      JSR   LASERS 3018; *** LASER CALLING ROUTINE ***
57 0006      LD     EXITR,#EXIT D73F
58 0008      JSR   LASERS 3018

```

Laser Gun (Continued)

```

59 000A D73F          LD      EXITR,#EXIT
60 000C 3018          JSR     LASER8
61 000E D715          LD      EXITR,#015      ; EXIT COUNT CAN BE INITIALIZED
62 0010 3018          JSR     LASER8          ; INSIDE PROGRAM IF SHOT RATE
63 0012 D715          LD      EXITR,#015      ; DOES NOT CHANGE.
64 0014 3018          JSR     LASER8
65 0016 B8             NOP
66 0017 EA             JP      LUP              ; **** LOOP FOR DEMO ****
67
;
68 0018 BCD530         LASER8: LD      PORTGC,#030
69 001B BCEEAA          LD      CNTRL,#OAA      ; SK = DIV BY 8, PWM/TIO TIMER
70 001E BCEF11          LD      PSW,#011       ; ENABLE TIMER INTERRUPT
71 0021 BCEAFF          LD      TMRLO,#TVALO   ; INITIALIZE TIMER
72 0024 BCEB00          LD      TMRHI,#TVALHI
73 0027 BCECFE          LD      TAULO,#TVALO
74 002A BCED00          LD      TAUHI,#TVALHI
75
;
76 002D D820           LD      EXITR,#EXIT    ; INITIALIZE EXIT COUNT
77 002F DA03           LD      TCNTR,#TCNT    ; INITIALIZE TONE COUNT
78 0031 BDEE7C          LD      LUPCNT,#XTRCT ; INITIALIZE EXTRACTION RATE
79 0034 A1             SBIT     TRUN,CNTRL    ; START TIMER
80 0035 5D             NOISE:  SC            ; INIT. STAGE 1
81 0036 9EFF           LD      B,#RNGVAL     ; POINT TO RANDOM NUMBER
82 0038 9CE9           LD      [B],#OFF      ; INIT RANDOM #
83 003A BDEF7A          SHIFT: X             A,SIOR                ; LOAD AND START SIOR
84 003D 9DFA           SBIT     BUSY,PSW
85 003F 9CF9           LD      A,LUPCNT     ; RESTORE EXTRACTION COUNT
86
;
87
; *****
88
; RING COUNTER
89
;
90
; THIS IS A NINE STAGE RING COUNTER (LINEAR
91
; FEEDBACK SHIFT REGISTER) WITH THE RRC COMMAND.
92
; THE COUNTER'S 8th AND 9th STAGES, THROUGH AN
93
; EXCLUSIVE-OR SERVE AS THE FEEDBACK FUNCTION.
94
; SINCE THE EXCLUSIVE OR OCCURS AFTER THE ROTATE,
95
; IT IS THE 1st AND 9th STAGES THAT ARE XOR'D,
96
; (THE CARRY FLAG AND BIT 0).
97
;
98
; CARRY BIT = STAGE 1
99
; LOW ORDER BIT = STAGE 9
100
; *****
101 0041 AE            RING:  LD      A,[B]      ; GET RANDOM #
102 0042 B0            RRC      A              ; ROTATE UPPER BYTE
103 0043 A6            X        A,[B]
104 0044 9800          LD      A,#000        ; PERFORM XOR
105 0046 85            AND     A,[B]
106 0047 9200          IFEQ   A,#000
107 0049 05            JP      TSTLUP
108 004A 88            IFC
109 004B 02            JP      RC
110 004C A1            SC
111 004D 01            JP      TSTLUP
112 004E A0            RC:    RC

```

Laser Gun (Continued)

```

113 004F C9          TSTLUP: DRSZ  LUPREG          ; EXTRACT THIS # ?
114 0050 F0          JP      RING              ; NO, KEEP ROTATING
115 0051 AE          LD      A,[B]            ; YES
116 0052 E5          JP      SHIFT
117                  ;
118                  ; **** TIMER INTERRUPT ROUTINE ****
119                  ;
120          00FF          . =      OFF
121 00FF BDEF75      IFBIT  TPND,PSW          ; TEST TIMER PND FLAG
122 0102 01          JP      TMOUT
123 0103 FF          JP      .              ; ERROR
124                  ;
125 0104 BDEE6C      TMOUT: RBIT  TRUN,CNTRL        ; STOP TIMER
126 0107 DEFA          LD      B,#LUPCNT
127 0109 C8          DRSZ   TCNTR              ; TEST FOR NEW TONE
128 010A 01          JP      NXT              ; NO
129 010B 0B          JP      NEWF
130 010C BDEF7C      NXT:   SBIT  4,PSW          ; ENABLE TIMER INTERRUPT
131 010F BDEF6D      RBIT  5,PSW          ; RESET TPND FLAG
132 0112 5D          LD      B,#RNGVAL        ; POINT TO RANDOM #
133 0113 BDEE7C      SBIT  TRUN,CNTRL        ; RESTART TIMER
134 0116 8F          RETI
135 0117 C7          NEWF:  DRSZ   EXITR          ; EXIT COUNT = 0 ?
136 0118 07          JP      NF              ; NO
137 0119 9DFD      NLST:  LD      A,SP          ; *** RESTORE STACK POINTER ***
138 011B 9402      ADD     A,#002          ; *** FROM TIMER INTERRUPT ***
139 011D 9CFD      X      A,SP          ; *** AND RETURN TO MAIN ***
140 011F 8E          RET
141 0120 AE          NF:    LD      A,[B]          ; NEW TONE
142 0121 9404      ADD     A,#04          ; INCR EXTRACTION VALUE
143 0123 A6          X      A,[B]
144 0124 D820      LD      TCNTR,#TCNT        ; REINITIALIZE TONE TIME
145 0126 E5          JP      NXT
146                  .END

```

DTMF Generation with a 3.58 MHz Crystal

National Semiconductor
Application Note 666
Verne H. Wilson



AN-666

DTMF (Dual Tone Multiple Frequency) is associated with digital telephony, and provides two selected output frequencies (one high band, one low band) for a duration of 100 ms. DTMF generation consists of selecting and combining two audio tone frequencies associated with the rows (low band frequency) and columns (high band frequency) of a push-button touch tone telephone keypad.

This application note outlines two different methods of DTMF generation using a COP820C/840C microcontroller clocked with a 3.58 MHz crystal in the divide by 10 mode. This yields an instruction cycle time of 2.79 μ s. The application note also provides a low true row/column decoder for the DTMF keyboard.

The first method of DTMF generation provides two PWM (Pulse Width Modulation) outputs on pins G3 and G2 of the G port for 100 ms. These two PWM outputs represent the selected high band and low band frequencies respectively, and must be combined externally with an LM324 op amp or equivalent feed back circuit to produce the DTMF signal.

The second method of DTMF generation uses ROM lookup tables to simulate the two selected DTMF frequencies. These table lookup values for the selected high band and low band frequencies are then combined arithmetically. The high band frequencies contain a higher bias value to compensate for the DTMF requirement that the high band frequency component be 2 dB above the low band frequency component to compensate for losses in transmission. The resultant value from the arithmetic combination of sine wave values is output on L port pins L0 to L5, and must be combined externally with a six input resistor ladder network to produce the DTMF signal. This resultant value is updated every 118 μ s. The COP820C/840C timer is used to time out the 100 ms duration of the DTMF. A timer interrupt at the end of the 100 ms is used to terminate the DTMF output. The external ladder network need not contain any active components, unlike the first method of DTMF generation with the two PWM outputs into the LM324 op amp.

The associated COP820C/840C program for the DTMF generation is organized as three subroutines. The first subroutine (KBRDEC) converts the low true column/row input from the DTMF keyboard into the associated DTMF hexadecimal digit. In turn, this hex digit provides the input for the other two subroutines (DTMFGP and DTMFLP), which represent the two different methods of DTMF generation. These three subroutines contain 35, 94, and 301 bytes of COP820C/840C code respectively, including all associated ROM tables. The Program Code/ROM table breakdowns are 19/16, 78/16, and 88/213 bytes respectively.

DTMF KEYBOARD MATRIX

The matrix for selecting the high and low band frequencies associated with each key is shown in *Figure 1*. Each key is uniquely referenced by selecting one of the four low band frequencies associated with the matrix rows, coupled with selecting one of the four high band frequencies associated with the matrix columns. The low band frequencies are

697 Hz, 770 Hz, 852 Hz, and 941 Hz, while the high band frequencies are 1209 Hz, 1336 Hz, 1477 Hz, and 1633 Hz. The DTMF keyboard input decode subroutine assumes that the keyboard is encoded in a low true row/column format, where the keyboard is strobed sequentially with four low true column selects with each returning a low true row select. The low true column and row selects are encoded in the upper and lower nibbles respectively of the accumulator, which serves as the input to the DTMF keyboard input decode subroutine. The subroutine will then generate the DTMF hexadecimal digit associated with the DTMF keyboard input digit.

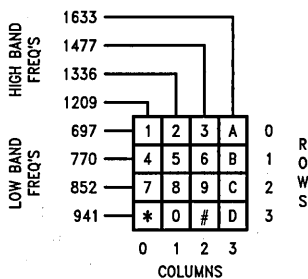
The DTMF keyboard decode subroutine (KBRDEC) utilizes a common ROM table lookup for each of the two nibbles representing the low true column and row encodings for the keyboard. The only legal low true nibbles for a single key input are E, D, B, and 7. All other low true nibble values represent multiple keys, no key, or no column strobe. Results from two legal nibble table lookups (from the same 16 byte ROM table) are combined to form a hex digit with the binary format of 0000RRCC, where RR represents the four row values and CC represents the four column values. The illegal nibbles are trapped, and the subroutine is exited with a RET (return) command to indicate multiple keys or no key. A pair of legal nibble table lookups result in the subroutine being exited with a RETSK (return and skip) command to indicate a single key input. This KBRDEC subroutine uses 35 bytes of code, consisting of 19 bytes of program code and 16 bytes of ROM table.

DTMF GENERATION USING PWM AND AN OP AMP

The first DTMF generation method (using the DTMFGP subroutine) generates the selected high band and low band frequencies as PWM (Pulse Width Modulation) outputs on pins G3 and G2 respectively of the G port. The COP820C/840C microcontrollers each contain only one timer, and three times must be generated to satisfy the DTMF application. These three times are the half periods of the two selected frequencies and the 100 ms duration period. Obviously the single timer can only generate one of the required times, while the program must generate the two remaining times. The solution lies in dividing the 100 ms duration time by the half periods for each of the eight DTMF frequencies, and then examining the respective high band and low band quotients and remainders. Naturally these divisions must be normalized to the instruction cycle time (t_c). 100 ms represents 35796 t_c 's. The results of these divisions are detailed in Table I.

The four high band frequencies are produced by running the COP820C/840C timer in PWM (Pulse Width Modulation) mode, while the program produces the four low band frequencies and the 100 ms duration timeout. The programmed times are achieved by using three programmed register counters R0, R2 and R3, with a backup register R1 to reload the counter R0. These three counters represent the half period, the 100 ms quotient, and the 100 ms remainder associated with each of the four low band frequencies.

3



TL/DD/10740-22

FIGURE 1. DTMF Keyboard Matrix

TABLE I. Frequency Half Periods, Quotients and Remainders

	Freq. Hz	Half Period in μ s	Half Period in t_c 's	100 ms/0.5P in t_c 's	
				Quotient	Remainder
Low Band Frequencies	697	717.36	257	139	73
	770	649.35	232	154	68
	852	586.85	210	170	96
	941	531.35	190	188	76
High Band Frequencies	1209	413.56	148	241	128
	1336	374.25	134	267	18
	1477	338.53	121	295	101
	1633	306.18	110	325	46

Note: 100 ms represents 35796 t_c 's.

The DTMFGP subroutine starts by transforming the DTMF hex digit in the accumulator (with binary format 0000RRCC) into low and high frequency vectors with binary formats 0011RR11 and 0011CC00 respectively. The transformation of the hex digit 0000RRCC (where RR is the row select and CC is the column select) into the frequency vectors is shown in Table II. The conversion produces a timer vector 0011CC00 (T), and three programmed counter vectors for R1, R2, and R3. The formats for the three counter vectors are 0011RR11 (F), 0011RR10 (Q), and 0011RR01 (R). These four vectors created from the core vector are used as

inputs for a 16 byte ROM table using the LAID (Load Accumulator InDirect) instruction. One of these four vectors (the T vector) is a function of the column bits (CC), while the other three vectors (F, Q, R) are a function of the row bits (RR). This correlates to only one parameter being needed for the timer (representing the selected high band frequency), while three parameters are needed for the three counters (half period, 100 ms quotient, 100 ms remainder) associated with the low band frequency and 100 ms duration. The frequency parameter ROM translation table, accessed by the T, F, Q, and R vectors, is shown in Table III.

TABLE II. DTMF Hex Digit Translation

DTMF Hex Digit— 0000RRCC -----

Timer Vector	Timer	T	0011CC00
Half Period Vector	R1	F	0011RR11
100 ms Quotient Vector	R2	Q	0011RR10
100 ms Remainder Vector	R3	R	0011RR01

TABLE III. Frequency Parameter ROM Translation Table

T— Timer	F— Frequency	Q— Quotient	R— Remainder
Address	Data (Decimal)	Vector	
0x30	147	T	
0x31	10	R	
0x32	140	Q	
0x33	38	F	
0x34	133	T	
0x35	9	R	
0x36	155	Q	
0x37	33	F	
0x38	120	T	
0x39	14	R	
0x3A	171	Q	
0x3B	31	F	
0x3C	109	T	
0x3D	10	R	
0x3E	189	Q	
0x3F	26	F	

The theory of operation in producing the selected low band frequency starts with loading the three counters with values obtained from a ROM table. The half period for the selected frequency is counted out, after which the G2 output bit is toggled. During this half period countout, the quotient counter is decremented. This procedure is repeated until the quotient counter counts out, after which the program branches to the remainder loop. During the remainder loop, the remainder counter counts out to terminate the 100 ms. Following the remainder countout, the G2 and G3 bits are both reset, after which the DTMF subroutine is exited. Great care must be taken in time balancing the half period loop for

the selected low band frequency. Furthermore, the toggling of the G2 output bit (achieved with either a set or reset bit instruction) must also be exactly time balanced to maintain the half period time integrity. Local stall loops (consisting of a DRSZ instruction followed by a JP jump back to the DRSZ for a two byte, six instruction cycle loop) are embedded in both the half period and remainder loops. Consequently, the ROM table parameters for the half period and remainder counters are approximately only one-sixth of what otherwise might be expected. The program for the half period loop, along with the detailed time balancing of the loop for each of the low band frequencies, is shown in *Figure 2*.

	Program	Bytes/ Cycles	Conditional Cycles	Cycles	Total Cycles
	LD B, #PORTGD	2/3			
	LD X, #R1	2/3			
LUP1:	LD A, [X-]	1/3		3	
	IFBIT 2, [B]	1/1		1	
	JP BYP1	1/3	3	1	
	X A, [X+]	1/3		3	
	SBIT 2, [B]	1/1		1	
	JP BYP2	1/3		3	
BYP1:	NOP	1/1	1		
	RBIT 2, [B]	1/1	1		
	X A, [X+]	1/3	3		
BYP2:	DRSZ R2	1/3		3	
	JP LUP2	1/3		3	
	JP FINI	1/3			
LUP2:	DRSZ R0	1/3		3	
	JP LUP2	1/3		3	1
	LD A, [X]	1/3			3
	IFEQ A, #31	2/2			2
	JP LUP1	1/3		1	3
	NOP	1/1		1	
	NOP	1/1		1	
	IFEQ A, #38	2/2		2	
	JP LUP1	1/3	1	3	35
	LAI	1/3	3		
	NOP	1/1	1		
	JP LUP1	1/3	3		40

Table III Frequency	Stall Loop	Total Cycles	Half Period
$[(38 - 1) \times 6]$	$\times 6$	+ 35	= 257
$[(33 - 1) \times 6]$	$\times 6$	+ 40	= 232
$[(31 - 1) \times 6]$	$\times 6$	+ 30	= 210
$[(26 - 1) \times 6]$	$\times 6$	+ 40	= 190

FIGURE 2. Time Balancing for Half Period Loop

TABLE IV. Time Balancing for Remainder Loop

Table III Remainder	Stall Loop	R Loop Overhead	Total Cycles	Table I Remainder
$[(10 - 1)]$	$\times 6]$	+ 20	= 74	73
$[(9 - 1)]$	$\times 6]$	+ 20	= 68	68
$[(14 - 1)]$	$\times 6]$	+ 20	= 98	96
$[(10 - 1)]$	$\times 6]$	+ 20	= 74	76

Note that the Q value in Table III is one greater than the quotient in Table I to compensate for the fact that the quotient count down to zero test is performed early in the half period loop. The overhead in the remainder loop is 20 instruction cycles. The detailed time balancing for the remainder loop is shown in Table IV.

The selected high band frequency is achieved by loading the half period count in t_c 's minus one (from Table III) into the timer autoreload register and running the timer in PWM output mode. The minus one is necessary since the timer toggles the G3 output bit when it underflows (counts down through zero), at which time the contents of the autoreload register are transferred into the timer.

In summary, the input digit from the keyboard (encoded in low true column/row format) is translated into a digit matrix vector XXXRRCC which is checked for 1001RRCC to indicate a single key entry. No key or multiple key entries will set a flag and terminate the DTMF subroutine. The digit matrix vector for a single key is transformed into the core vector 0000RRCC. The core vector is then translated into four other vectors (T, F, Q, R) which in turn are used to select four parameters from a 16 byte ROM table. These four parameters are used to load the timer, and the respective half period, quotient, and remainder counters. The 16 byte ROM table must be located starting at ROM location 0030 (or 0X30) in order to minimize program size, and has reference setups with the "OR A, #033" instruction for the F vector and the "OR A, #030" instruction for the T vector.

The three parameters associated with the two R bits of the core vector require a multi-level table lookup capability with the LAID instruction. This is achieved with the following section of code in the DTMF subroutine:

```

LD      B, #R1
LUP:   X      A, [B]
LD      A, [B, ]
LAID
X      A, [B+]
DEC     A
IFBNE  #4
JP      LUP

```

This program loads the F frequency vector into R1, and then decrements the vector each time around the loop. The vector is successively moved with the exchange commands from R1 to R2 to R3 as one of the same exchange commands loads the data from the ROM table into R1, R2, and R3. This successive decrementation of the F vector changes the F vector into the Q vector, and then changes the Q vector into the R vector. These vectors are used to access the ROM table with the LAID instruction. The B pointer is incremented each time around the loop after it has been used to store away the three selected ROM table parameters (one per loop). These three parameters are stored in sequential RAM locations R1, R2, and R3. The IFBNE test instruction is used to skip out of the loop once the three selected ROM table parameters have been accessed and stored away.

The timer is initialized to a count of 15 so that the first timer underflow and toggling of the G3 output bit (with timer PWM mode and G3 toggle output selected) will occur at the same time as the first toggling of the G2 output bit. The half period counts for the high band frequencies minus one are stored in the timer section of the ROM table. The selected value from this frequency ROM table is stored in the timer autoreload register. The timer is selected for PWM output mode and started with the instruction LD [B], #0B0 where the B pointer is selecting the CNTRL register at memory location 0EE.

This first DTMF generation subroutine for the COP820C/840C uses 94 bytes of code, consisting of 78 bytes of program code and 16 bytes of ROM table. A program test routine to sequentially call the DTMFGP subroutine for each of the 16 keyboard input digits is supplied with the listing for the DTMF35 program. This test routine uses a 16 byte ROM table to supply the low true encoded column/row keyboard input to the accumulator. An input from the I0 input pin of the I port is used to select which DTMF generation subroutine is to be used. The DTMFGP subroutine is selected with I0 = 0.

A TYPICAL OP AMP CONFIGURATION FOR MIXING THE TWO DTMF PWM OUTPUTS IS SHOWN IN FIGURE 3.

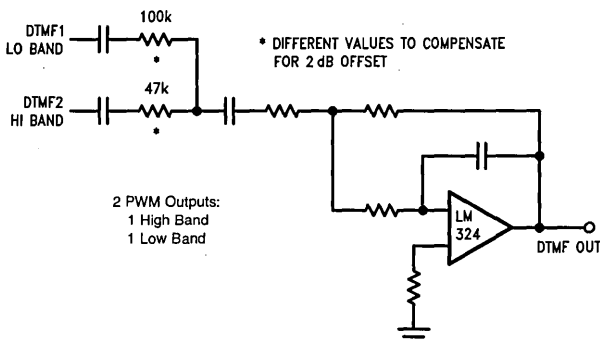


FIGURE 3. Typical Op Amp Configuration for Mixing DTMF PWM Outputs

TL/DD/10740-23

DTMF GENERATION USING A RESISTOR LADDER NETWORK

The second DTMF generation method (using the DTMFLP subroutine) generates and combines values from two table lookups simulating the two selected sine waves. The high band frequency table values have a higher base line value (16 versus 13) than the low band frequency table values. This higher bias for the high frequency values is necessary to satisfy the DTMF requirement that the high band DTMF frequencies need a value 2 dB greater than the low band DTMF frequencies to compensate for losses in transmission.

The resultant value from arithmetically combining the table lookup low band and high band frequency values is output on pins L0 to L5 of the L port in order to feed into a six input external resistor ladder network. The resultant value is updated every $117\frac{1}{3}$ μ s (one cycle of the LUP42 program loop). The LUP42 program loop contains 42 instruction cycles (t_C 's) of 2.7936511 μ s each for a total loop time of $117\frac{1}{3}$ μ s. The COP820C/840C timer is used to count out the 100 ms DTMF duration time.

An interrupt from the timer terminates the 100 ms DTMF output. Note that the Stack Pointer (SP) must be adjusted following the timer interrupt before returning from the DTMFLP subroutine.

The DTMFLP subroutine starts by quadrupling the value of the DTMF hex digit value in the accumulator, and then adding an offset value to reach the first value in the telephone key table. The telephone key ROM table contains four values associated with each of the 16 DTMF hex keys. These four values represent the low and high frequency table sizes and table starting addresses associated with the pair of frequencies (one low band, one high band) associated with each DTMF key. The FRLUP section of the program loads the four associated telephone key table values from the ROM table into the registers LFTBSZ (Low Freq Table Size), LFTADR (Low Freq Table Address), HFTBSZ (High Freq Table Size), and HFTADR (High Freq Table Address). The program then initializes the timer and autoreload register, starts the timer, and then jumps to LUP42. Note that the timer value in t_C 's is 100 ms plus one LUP42 time, since the initial DTMF output is not until the end of the LUP42 program.

Multiples of the magic number 118 μ s (approximately) are close approximations to all eight of the DTMF frequencies. The LUP42 program uses 42 instruction cycles (of 2.7936511 μ s each) to yield a LUP42 time of $117\frac{1}{3}$ μ s. The purpose of the LUP42 program is to update the six L port outputs by accessing and then combining the next set of

values from the selected low band and high band sine wave frequency tables in the ROM. The ROM table offset frequency pointers (LFPTR and HFPTR) must increment each time and then wrap around from top to bottom of the two selected ROM tables. The ROM table size parameters (LFTBSZ and HFTBSZ) for the selected frequencies are tested during each LUP42 to determine if the wrap around from ROM table top to bottom is necessary. The wrap around is implemented by clearing the frequency pointer in question. Note that the ROM tables are mapped from a reference of 0 to table size minus one, so that the table size is used in a direct comparison with the frequency offset pointer to test for the need for a wrap around. Also note that the offset pointer incremented value is used during the following LUP42 cycle, while the pre-incremented value of the pointer is used during the current cycle. However, it is the incremented value that is tested versus the table size for the need to wrap around.

After the low band and high band ROM table sine wave frequency values are accessed in each cycle of the LUP42 program, they are added together and then output to pins L0-L5 of the L port. As stated previously, the low band frequency values have a lower bias than the high band frequency values to compensate for the required 2 dB offset. Specifically, the base line and maximum values for the low frequency values are 13 and 26 respectively, while the base line and maximum values for the high frequency values are 16 and 32 respectively. Thus the combined base line value is 29, while the combined maximum value is 58. This gives a range of values on the L port output (L0-L5) from 0 to 58.

The minimum time necessary for the LUP42 update program loop is 36 instruction cycles including the jump back to the start of the loop. Consequently, two LAID instructions are inserted just prior to the jump back instruction at the end of LUP42 to supply the six extra NOP instruction cycles needed to increase the LUP42 instruction cycles from 36 to 42. A three cycle LAID instruction can always be used to simulate three single cycle NOP instructions if the accumulator data is not needed.

Table V shows the multiple LUP42 approximation to the eight DTMF frequencies, including the number of sine wave cycles and data points in the approximation. As an example, three cycles of a sine wave with a total of 19 data points across the three cycles is used to approximate the 1336 Hz DTMF frequency. The 19 cycles of LUP42 times the LUP42 time of $117\frac{1}{3}$ μ s is divided into the three cycles to yield a value of 1345.69 Hz. This gives an error of +0.73% when compared with the DTMF value of 1336 Hz. This is well within the 1.5% North American DTMF error range.

TABLE V. DTMF Frequency Approximation Table

DTMF Freq.	# of Sine Wave Cycles	# of Data Points	Calculation	Approx. Freq.	% Error
697	4	49	$4/(49 \times 117\frac{1}{3})$	= 695.73	-0.18
770	1	11	$1/(11 \times 117\frac{1}{3})$	= 774.79	+0.62
852	1	10	$1/(10 \times 117\frac{1}{3})$	= 852.27	+0.03
941	1	9	$1/(9 \times 117\frac{1}{3})$	= 946.97	+0.63
1209	1	7	$1/(7 \times 117\frac{1}{3})$	= 1217.53	+0.71
1336	3	19	$3/(19 \times 117\frac{1}{3})$	= 1345.69	+0.73
1477	4	23	$4/(23 \times 117\frac{1}{3})$	= 1482.21	+0.35
1633	4	21	$4/(21 \times 117\frac{1}{3})$	= 1623.38	-0.59

The frequency approximation is equal to the number of cycles of sine wave divided by the time in the total number of LUP42 cycles before the ROM table repeats.

The values in the DTMF sine wave ROM tables are calculated by computing the sine value at the appropriate points, scaling the sine value up to the base line value, and then adding the result to the base line value. The following example will help to clarify this calculation.

Consider the three cycles of sine wave across 19 data points for the 1336 Hz high band frequency. The first value in the table is the base line value of 16. With 2π radians per sine wave cycle, the succeeding values in the table represent the sine values of $1 \times (6\pi/19)$, $2 \times (6\pi/19)$, $3 \times (6\pi/19)$, . . . , up to $18 \times (6\pi/19)$. Consider the seventh and eighth values in the table, representing the sine values of $6 \times (6\pi/19)$ and $7 \times (6\pi/19)$ respectively. The respective calculators of $16 \times \sin[6 \times (6\pi/19)]$ and $16 \times \sin[7 \times (6\pi/19)]$ yield values of -5.20 and 9.83 . Rounding to the nearest integer gives values of -5 and 10 . When added to the base line value of 16, these values yield the results 11 and 26 for the seventh and eighth values in the 1336 Hz DTMF ROM table. Symmetry in the loop of 19 values in the DTMF table dictates that the fourteenth and thirteenth values in the table are 21 and 6, representing values of 5 and -10 from the calculations.

The area under a half cycle of sine wave relative to the area of the surrounding rectangle is $2/\pi$, where π radians represent the sine wave half cycle. This surrounding rectangle has a length of π and a height of 1, with the height representing the maximum sine value. Consequently, the area of the surrounding rectangle is π . The integral of the area under the half sine wave from 0 to π is equal to 2. The ratio of $2/\pi$ is equal to 63.66%, so that the total of the values for each half sine wave should approximate 63.66% of the sum of the max values. The maximum values (relative to the base line) are 13 and 16 respectively for the low and high band DTMF frequencies.

For the previous 1336 Hz example, the total of the absolute values for the 19 sine values from the 1336 Hz ROM

table is equal to 196. The surrounding rectangle for the three cycles of sine wave is 19 by 16 for a total area of 304. The ratio of 196/304 is 64.47% compared with the $2/\pi$ ratio of 63.66%. Thus the sine wave approximation gives an area abundance of 0.81% (equal to 64.47 - 63.66).

An application of the sine wave area criteria is shown in the generation of the DTMF 852 Hz frequency. The ten sine values calculated are 0, 7.64, 12.36, 12.36, 7.64, 0, -7.64 , -12.36 , -12.36 , and -7.64 . Rounding off to the nearest integer yields values of 0, 8, 12, 12, 8, 0, -8 , -12 , -12 and -8 . The total of these values (absolute numbers) is 80, while the area of the surrounding rectangle is 130 (10×13). The ratio of 80/130 is 61.54% compared with the $2/\pi$ ratio of 63.66%. Thus the sine wave approximation gives an area deficiency of 2.12% (equal to 63.66 - 61.54), which is overly deficient. Consequently, two of the ten sine values are augmented to yield sine values of 0, 8, 12, 13*, 8, 0, -8 , -12 , -13^* , and -8 . This gives an absolute total of 82 and a ratio of 82/130, which equals 63.08% and serves as a much better approximation to the $2/\pi$ ratio of 63.66%.

The sine wave area criteria is also used to modify two values in the DTMF 941 Hz frequency. The nine sine values calculated are 0, 8.36, 12.80, 11.26, 4.45, -4.45 , -11.26 , -12.80 , and -8.36 . Rounding off to the nearest integer yields values of 0, 8, 13, 11, 4, -4 , -11 , -13 , and -8 . The total of these values (absolute numbers) is 72, while the area of the surrounding rectangle is 117 (9×13). The ratio of 72/117 is 61.54% compared to the $2/\pi$ ratio of 63.66%. Thus the sine wave approximation gives an area deficiency of 2.12% (equal to 63.66 - 61.54), which is overly deficient. Rounding up the two values of 4.45 and -4.45 to 5 and -5 , rather than down to 4 and -4 , yields values of 0, 8, 13, 11, 5, -5 , -11 , -13 and -8 . This gives an absolute total of 74 and a ratio of 74/117, which equals 63.25% and serves as a much better approximation to the $2/\pi$ ratio of 63.66%.

With these modified values for the 852 and 941 DTMF frequencies, the area criteria ratio of $2/\pi$ = 63.66% for the sine wave compared to the surrounding rectangle has the following values:

DTMF Freq.	Sum of Values	Rectangle Area	Percentage	Diff.
697 Hz	406	$49 \times 13 = 637$	63.74%	+0.08%
770 Hz	92	$11 \times 13 = 143$	64.34%	+0.68%
852 Hz	82	$10 \times 13 = 130$	63.08%	-0.58%
941 Hz	74	$9 \times 13 = 117$	63.25%	-0.41%
1209 Hz	72	$7 \times 16 = 112$	64.29%	+0.63%
1336 Hz	196	$19 \times 16 = 304$	64.47%	+0.81%
1477 Hz	232	$23 \times 16 = 368$	63.04%	-0.62%
1633 Hz	216	$21 \times 16 = 336$	64.29%	+0.63%

The LUP42 program loop is interrupted by the COP820C/840C timer after 100 ms of DTMF output. As stated previously, the Stack Pointer (SP) must be adjusted (incremented by 2) following the timer interrupt before returning from the DTMFLP subroutine.

This second DTMF generation subroutine for the COP820C/840C uses 301 bytes of code, consisting of 88 bytes of program code and 213 bytes of ROM table. The following is a summary of the DTMFLP subroutine code allocation.

DTMFLP Code Allocation	# of Bytes
1. Subroutine Header Code	42
2. Interrupt Code	16
3. LUP42 Code	30
4. Telephone Key Table	64
5. Sine Value Tables	149
Total	301

A program test routine to sequentially call the DTMFLP subroutine for each of the 16 DTMF keyboard input digits is supplied with the listing for the DTMF35 program. This test routine uses a 16 byte ROM table to supply the low true encoded column/row keyboard input to the accumulator. An input from the I0 pin of the I port is used to select which DTMF generation subroutine is to be used. The DTMFLP subroutine is selected with I0 = 1.

A TYPICAL RESISTOR LADDER NETWORK IS SHOWN IN FIGURE 4.

SUMMARY

In summary, the DTMF35 program assumes a COP820C/840C clocked with a 3.58 MHz crystal in divide by 10 mode. The DTMF35 program contains three subroutines, KBRDEC, DTMFGP, and DTMFLP. The KBRDEC subroutine is a low true DTMF keyboard decoder, while the DTMFGP and DTMFLP subroutines represent the alternative methods of DTMF generation.

The KBRDEC subroutine provides a low true decoding of the DTMF keyboard input and assumes that the keyboard input has been encoded in a low true column/row format, with the columns of the keyboard being sequentially strobed.

The DTMFGP subroutine produces two PWM (Pulse Width Modulation) outputs (representing the selected high and low band DTMF frequencies) for combination with an external op amp network (LM324 or equivalent).

The DTMFLP subroutine produces six bits of combined high band and low band DTMF frequency output for combination in an external resistor ladder network. This output represents a combined sine wave simulation of the two selected DTMF frequencies by combining values from two selected ROM tables, and updating these values every 118 μ s.

The three DTMF35 subroutines contain the following number of bytes of program and ROM table memory:

Subroutine	# of Bytes of Program	# of Bytes of ROM Table	Total # of Bytes
KBRDEC	19	16	35
DTMFGP	78	16	94
DTMFLP	88	213	301

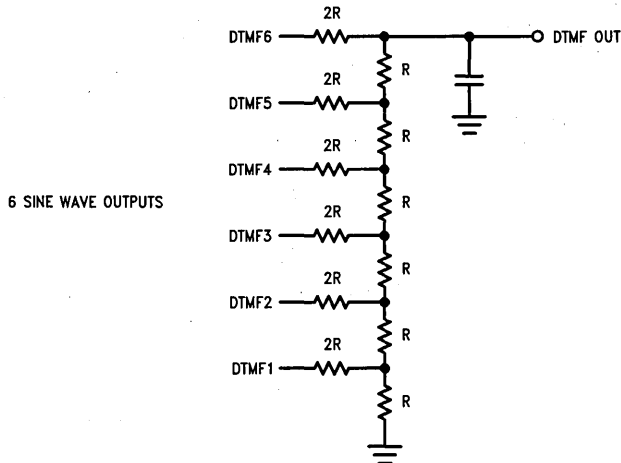


FIGURE 4. Typical Resistor Ladder Network

TL/DD/10740-24

```

1      ;
2      ; DTMF GENERATION WITH A 3.58 MHZ          VERNE H. WILSON
3      ;                                CRYSTAL FOR COP820C/840C          10/28/89
4      ;
5      ; DTMF - DUAL TONE MULTIPLE FREQUENCY
6      ;
7      ; PROGRAM NAME: DTMF35.MAC
8      ;
9      ;           .TITLE DTMF35
10     ;           .CHIP 840
11     ;
12     ;
13     ; THIS DTMF PROGRAM IS BASED ON A COP820C/840C RUNNING
14     ; WITH A CKI CLOCK OF 3.579545 MHZ (TV COLOR CRYSTAL
15     ; FREQUENCY) IN DIVIDE BY 10 MODE, FOR AN INSTRUCTION
16     ; CYCLE TIME OF 2.7936511 MICROSECONDS.
17     ;
18     ; THIS PROGRAM CONTAINS THREE SUBROUTINES, ONE FOR A
19     ; LOW TRUE ROW/COLUMN DTMF KEYBOARD DECODING (KBRDEC),
20     ; AND THE OTHER TWO (DTMFGP, DTMFLP) FOR ALTERNATE
21     ; METHODS OF DTMF GENERATION.
22     ;
23     ; KEYBOARD INPUT DATA IS IN ACCUMULATOR WITH A
24     ; LOW TRUE FORMAT AS FOLLOWS:
25     ;     BITS 7 TO 4 : LOW TRUE COLUMN VALUE (E,D,B,7)
26     ;     BITS 3 TO 0 : LOW TRUE ROW VALUE (E,D,B,7)
27     ;
28     ; ASSUMPTION MADE THAT COLUMN STROBES (LOW TRUE) ARE
29     ; OUTPUT, WHILE ROW VALUES (LOW TRUE) ARE INPUT.
30     ;
31     ; THE FIRST METHOD OF DTMF GENERATION CONSISTS OF
32     ; GENERATING TWO PWM OUTPUTS ON THE G PORT G2 AND G3
33     ; OUTPUT PINS. THESE TWO OUTPUTS NEED TO BE MIXED
34     ; EXTERNALLY WITH AN APPROPRIATE LM324 OP AMP FEEDBACK
35     ; CIRCUIT TO GENERATE THE DTMF.
36     ;
37     ; THE SECOND METHOD OF DTMF GENERATION USES ROM LOOKUP
38     ; TABLES TO SIMULATE THE TWO DTMF SINE WAVES AND
39     ; COMBINES THEM ARITHMETICALLY. THE RESULT IS OUTPUT ON
40     ; THE LOWER SIX BITS OF THE L PORT (L0 - L5). THESE SIX
41     ; OUTPUTS ARE COMBINED EXTERNALLY WITH A LADDER NETWORK
42     ; TO GENERATE THE DTMF.
43     ;
44     ; THE SECOND DTMF GENERATION METHOD USES APPROXIMATELY
45     ; THREE TIMES AS MUCH ROM CODE (INCLUDING PROGRAM CODE
46     ; AND ROM TABLES) AS THE FIRST METHOD, BUT HAS THE
47     ; ADVANTAGE OF ELIMINATING THE COST OF THE EXTERNAL
48     ; ACTIVE COMPONENT (LM324 OR EQUIVALENT).
49     ;
50     ; BOTH OF THE DTMF SUBROUTINES GENERATE THEIR OUTPUTS
51     ; FOR A PERIOD OF 100 MILLISECONDS.

```

TL/DD/10740-1

```

52      ;
53      ; DECLARATIONS:
54      ;
55      0000      KDATA = 0      ; *** KEYBOARD DATA ***
56      00D0      PORTLD = 0D0    ; PORTL DATA REG
57      00D1      PORTLC = 0D1    ; PORTL CONFIG REG
58      00D4      PORTGD = 0D4    ; PORTG DATA REG
59      00D5      PORTGC = 0D5    ; PORTG CONFIG REG
60      00D7      PORTI = 0D7     ; PORTI INPUT PINS
61      00DC      PORTD = 0DC     ; PORTD REG
62      00EA      TMRLO = 0EA     ; TIMER LOW COUNTER
63      00EB      TMRHI = 0EB     ; TIMER HIGH COUNTER
64      00EC      TAULO = 0EC     ; TMR AUTORELOAD REG LO
65      00ED      TAUHI = 0ED     ; TMR AUTORELOAD REG HI
66      00EE      CNTRL = 0EE     ; CONTROL REG
67      00EF      PSW = 0EF       ; PROC STATUS WORD
68      00F0      R0 = 0F0        ; LB FREQ LOOP COUNTER
69      00F1      R1 = 0F1        ; LB FREQ LOOP COUNT
70      00F2      R2 = 0F2        ; LB FREQ Q COUNT
71      00F3      R3 = 0F3        ; LB FREQ R COUNT
72      ;
73      0000 DD2F      START: LD      SP,#02F      ; INITIALIZE STACK PTR
74      ;
75      ;
76      ;
77      0002 DEDC      LD      B,#PORTD      ;
78      0004 9E00      LD      [B],#0      ;
79      0006 A0        RC      ;
80      0007 AE        LD      A,[B]      ; DTMF TEST LOOP
81      0008 9405      ADD     A,#5      ; SEQUENCE IS 1,5,9,D,4,
82      000A A6        X      A,[B]      ; 8,#,A,7,0,3,B,*,2,6,C
83      000B 6C        RBIT   4,[B]      ; HEX MATRIX TO LOOKUP
84      000C 9420      ADD     A,#020     ; TABLE FOR LOW TRUE
85      000E A4        LAID   ;
86      000F 3210      JSR    KBRDEC      ; KBRDEC SUBROUTINE
87      0011 A1        SC      ; SET C IF NOT SINGLE KEY
88      0012 DED7      LD      B,#PORTI    ; TEST BIT 0 OF PORTI TO
89      0014 70        IFBIT  0,[B]      ; DETERMINE WHICH
90      0015 03        JP      BYPA      ; DTMF SUBROUTINE
91      0016 3040      JSR    DTMFGP     ; TWO PWM OUTPUTS ON
92      0018 02        JP      BYPB      ; G PORT PINS G2,G3
93      0019 308E      BYPA: JSR    DTMFLP ; SIX LADDER OUTPUTS ON
94      ;
95      001B DEDC      BYPB: LD      B,#PORTD ; L PORT PINS L0 - L5
96      001D E8        JP      LOOP      ; DO WILL TOGGLE FOR EACH
97      ;
98      ;
99      ;

```

TL/DD/10740-2

```

100 .FORM
101 ;
102 ; KEYBOARD DIGIT MATRIX TABLE
103 ;
104 0020 . = 020
105 ;
106 ;
107 0020 EE .BYTE 1 5 9 D 4 8 # A
    0021 DD OEE,ODD,ODD,077,0ED,0DB,0B7,07E
    0022 BB
    0023 77
    0024 ED
    0025 DB
    0026 B7
    0027 7E
108 ;
109 0028 EB .BYTE 7 0 3 B * 2 6 C
    0029 D7 OEB,OD7,0BE,07D,0E7,0DE,0BD,07B
    002A BE
    002B 7D
    002C E7
    002D DE
    002E BD
    002F 7B
110 ;
111 ;
112 ;
113 ;
114 ;
115 ; FIRST DTMF SUBROUTINE (DTMF GP) PRODUCES TWO PWM
116 ; (PULSE WIDTH MODULATION) OUTPUTS ON PINS G3, G2
117 ;
118 ;
119 ; G PORT IS USED FOR THE TWO OUTPUTS
120 ; - HIGH BAND (HB) FREQUENCY OUTPUT ON G3
121 ; - LOW BAND (LB) FREQUENCY OUTPUT ON G2
122 ;
123 ; TIMER COUNTS OUT
124 ; - HB FREQUENCIES
125 ;
126 ; PROGRAM COUNTS OUT
127 ; - LB FREQUENCIES
128 ; - 100 MSEC DIVIDED BY LB HALF PERIOD QUOTIENT
129 ; - 100 MSEC DIVIDED BY LB HALF PERIOD REMAINDER
130 ;
131 ; NOTE THAT ALL COUNTS MUST BE NORMALIZED TO THE
132 ; 2.7936511 MICROSECOND INSTRUCTION CYCLE Tc
133 ;
134 ; 100 MSEC REPRESENTS 35796 Tc's
135 ;
136 ;

```

TL/DD/10740-3

```

137 ;
138 ;
139 ; HALF PERIODS FOR THE 8 DTMF FREQUENCIES (697,770,852,
140 ; 941,1209,1336,1477, AND 1633 KHZ) ARE 257,232,
141 ; 210,190,148,134,121, AND 110 Tc's RESPECTIVELY
142 ;
143 ; THE 100 MSEC DIVIDED BY HALF PERIOD QUOTIENTS ARE
144 ; 139,154,170,188,241,267,295, AND 325 RESPECTIVELY
145 ;
146 ; THE 100 MSEC DIVIDED BY HALF PERIOD REMAINDERS ARE
147 ; 72,67,95,75,127,17,100, AND 45 RESPECTIVELY
148 ;
149 ;
150 ;
151 ;
152 ; BINARY FORMAT FOR THE HEX DIGIT KEY VALUE FROM THE
153 ; KBRDEC SUBROUTINE IS 000ORRCC,
154 ; WHERE - RR IS ROW SELECT (LB FREQUENCIES)
155 ; - CC IS COLUMN SELECT (HB FREQUENCIES)
156 ;
157 ; FREQUENCY VECTORS (HB & LB) FOR FREQ PARAMETER TABLE
158 ; MADE FROM KEY VALUE
159 ;
160 ; HB FREQ VECTORS (4) END WITH 00 FOR TIMER COUNTS,
161 ; WHERE VECTOR FORMAT IS 0011CC00
162 ;
163 ; LB FREQUENCY VECTORS (12) END WITH:
164 ; 11 FOR HALF PERIOD LOOP COUNTS,
165 ; WHERE VECTOR FORMAT IS 0011RR11
166 ; 10 FOR 100 MSEC DIVIDED BY HALF PERIOD QUOTIENTS,
167 ; WHERE VECTOR FORMAT IS 0011RR10
168 ; 01 FOR 100 MSEC DIVIDED BY HALF PERIOD REMAINDERS,
169 ; WHERE VECTOR FORMAT IS 0011RR01
170 ;
171 ; FREQ PARAMETER TABLE AT HEX 003* (REQUIRED LOCATION)
172 ;
173 ;
174 ;
175 ; KEY VALUE
176 ; 000ORRCC
177 ;
178 ; TIMER T CC00
179 ; R1 F RR11
180 ; R2 Q RR10
181 ; R3 R RR01
182 ;
183 ;
184 ;

```

TL/DD/10740-4

.FORM

185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208

;FREQUENCY AND 100 MSEC PARAMETER TABLE

190 0030 93	.BYTE	147	; T
191 0031 0A	.BYTE	10	; R
192 0032 8C	.BYTE	140	; Q
193 0033 26	.BYTE	38	; F
194 0034 85	.BYTE	133	; T
195 0035 09	.BYTE	9	; R
196 0036 9B	.BYTE	155	; Q
197 0037 21	.BYTE	33	; F
198 0038 78	.BYTE	120	; T
199 0039 0E	.BYTE	14	; R
200 003A AB	.BYTE	171	; Q
201 003B 1F	.BYTE	31	; F
202 003C 6D	.BYTE	109	; T
203 003D 0A	.BYTE	10	; R
204 003E BD	.BYTE	189	; Q
205 003F 1A	.BYTE	26	; F

209 0040 DED5
210 0042 9B3F
211 0044 6B
212 0045 6A
213 0046 5F
214 0047 A6
215 0048 AE
216 0049 9733
217 004B DEF1
218 004D A6
219 004E AE
220 004F A4
221 0050 A2
222 0051 8B
223 0052 44
224 0053 F9
225 0054 5F
226 0055 AE
227 0056 65
228 0057 A0
229 0058 B0
230 0059 B0
231 005A 9730
232 005C A4
233 005D DEEA
234 005F 9A0F
235 0061 9A00

```
DTMFGP: LD B,#PORTGC ; CONFIGURE G PORT
          LD [B-],#03F ; FOR OUTPUTS
          RBIT 3,[B] ; OPTIONAL HB RESET
          RBIT 2,[B] ; OPTIONAL LB RESET
          LD B,#KDATA
          X A,[B] ; STORE KEY VALUE
          LD A,[B] ; KEY VALUE TO ACC
          OR A,#033 ; CREATE LB FREQ VECTOR
          LD B,#R1 ; FROM KEY VALUE
LUP: X A,[B]
     LD A,[B] ; THREE PARAMETERS
     LAID ; FROM LOW BAND
     X A,[B+] ; FREQ ROM TABLE
     DEC A ; TO R1,R2,R3
     IFBNE #4
     JP LUP
     LD B,#KDATA
     LD A,[B] ; KEY VALUE TO ACC
     SWAP A ; CREATE HB FREQ VECTOR
     RC ; FROM KEY VALUE
     RRC A
     RRC A
     OR A,#030
     LAID ; HB FREQ TABLE
     LD B,#TMRLO ; (1 PARAMETER)
     LD [B+],#15 ; INSTRUCTION CYCLE
     LD [B+],#0 ; TIME UNTIL TOGGLE
```

TL/DD/10740-5


```

236 0063 A2          X          A,[B+]      ; HB FREQ PARAMETER TO
237 0064 9A00       LD          [B+],#0    ;  AUTORELOAD REGISTER
238 0066 9EB0       LD          [B],#0B0    ;  START TIMER PWM
239 0068 DED4       LD          B,#PORTGD
240 006A DCF1       LD          X,#R1
241 006C BB         LUP1:      LD          A,[X-]
242 006D 72         IFBIT      2,[B]      ; TEST LB OUTPUT
243 006E 03         JP          BYP1
244 006F B2         X          A,[X+]
245 0070 7A         SBIT      2,[B]      ; SET LB OUTPUT
246 0071 03         JP          BYP2
247 0072 B8         BYP1:      NOP
248 0073 6A         RBIT      2,[B]      ; RESET LB OUTPUT
249 0074 B2         X          A,[X+]
250 0075 C2         BYP2:      DRSZ      R2          ; DECR. QUOT. COUNT
251 0076 01         JP          LUP2
252 0077 0E         JP          FINI      ; Q COUNT FINISHED
253 0078 C0         LUP2:      R0          ; DECR. F COUNT
254 0079 FE         JP          LUP2      ; LB (HALF PERIOD)
255                ;
256 007A BE         LD          A,[X]      ; *****
257 007B 921F       IFEQ      A,#31    ; BALANCE      ***
258 007D EE         JP          LUP1      ; LOW BAND     ***
259 007E B8         NOP          ; FREQUENCY    ***
260 007F B8         NOP          ; RESIDUE      ***
261 0080 9226       IFEQ      A,#38    ; DELAY FOR    ***
262 0082 E9         JP          LUP1      ; EACH OF THE  ***
263 0083 A4         LAID      ; FOUR LOW BAND ***
264 0084 B8         NOP          ; FREQUENCIES  ***
265 0085 E6         JP          LUP1      ; *****
266 0086 C3         FINI:      DRSZ      R3          ; DECR. REMAINDER COUNT
267 0087 FE         JP          FINI      ; REM. COUNT NOT FINISHED
268 0088 BDEE6C     RBIT      4,CNTRL  ; STOP TIMER
269 008B 6B         RBIT      3,[B]    ; OPTIONAL CLR HB OUTPUT
270 008C 6A         RBIT      2,[B]    ; OPTIONAL CLR LB OUTPUT
271 008D 8E         RET          ; RETURN FROM SUBROUTINE
272                ;
273                ;
274                ;

```

TL/DD/10740-6

```

275             .FORM
276             ;
277             ; SECOND DTMF SUBROUTINE (DTMFLP) PRODUCES SIX
278             ; COMBINED LOW BAND AND HIGH BAND FREQUENCY
279             ; SINE WAVE OUTPUTS ON PINS LO - L5
280             ;
281             ; SIX L PORT OUTPUTS (LO - L5) FEED INTO AN EXTERNAL
282             ; RESISTOR LADDER NETWORK TO CREATE THE DTMF OUTPUT.
283             ;
284             ; FOUR VALUES FROM A KEYBOARD ROM TABLE ARE LOADED
285             ; INTO LFTBSZ (LOW FREQ TABLE SIZE), LFTADR (LOW
286             ; FREQ TABLE ADDRESS), HFTBSZ (HIGH FREQ TABLE SIZE),
287             ; AND HFTADR (HIGH FREQ TABLE ADDRESS).
288             ;
289             ; LUP42 USES THE LFPTR (LOW FREQ POINTER) AND HFPTR
290             ; (HIGH FREQ POINTER) TO ACCESS THE SINE DATA TABLES
291             ; FOR THE SELECTED FREQUENCIES ONCE PER LOOP. THESE
292             ; POINTERS ARE BOTH INCREMENTED ONCE PER LUP42.
293             ;
294             ; LUP42 PROGRAM LOOP UPDATES THE OUTPUT VALUE EVERY
295             ; 117 1/3 uSEC BY SELECTING AND THEN COMBINING NEW
296             ; VALUES FROM THE SELECTED LOW BAND AND HIGH BAND
297             ; FREQUENCY ROM TABLES WHICH SIMULATE THE SINE WAVES
298             ; FOR THE TWO FREQUENCIES.
299             ;
300             ; MULTIPLES OF THE MAGIC NUMBER OF APPROXIMATELY
301             ; 118 uSEC ARE CLOSE APPROXIMATIONS TO ALL EIGHT OF
302             ; THE DTMF FREQUENCIES.
303             ;
304             ; COP820C/840C TIMER USED TO INTERRUPT THE DTMF LUP42
305             ; PROGRAM LOOP AFTER 100 MSEC TO FINISH THE DTMF
306             ; OUTPUT AND RETURN FROM THE DTMFLP SUBROUTINE. NOTE
307             ; THAT THE STACK POINTER (SP) MUST BE ADJUSTED AFTER
308             ; THE INTERRUPT BEFORE RETURNING FROM THE SUBROUTINE.
309             ;
310             ;
311             ;
312             ;
313             ;
314             ; DECLARATIONS:
315             ;
316             0005             LFPTR   = 05             ; LOW FREQ POINTER
317             0006             TEMP    = 06             ; TEMPORARY
318             0007             HFPTR   = 07             ; HIGH FREQ POINTER
319             0008             LFTBSZ  = 08             ; LO FREQ TABLE SIZE
320             0009             LFTADR  = 09             ; LO FREQ TABLE ADDR
321             000A             HFTBSZ  = 0A             ; HI FREQ TABLE SIZE
322             000B             HFTADR  = 0B             ; HI FREQ TABLE ADDR
323             ;
324             0004             TRUN    = 04
325             ;

```

TL/DD/10740-7

```

326 ;
327 008E BCD1FF      ; DTMFLP: LD PORTLC,#OFF ; INITIALIZE PORT L
328 0091 BCD01D      ; LD PORTLD,#29 ; FOR NO TONE OUT
329 0094 BC0500      ; LD LFPTR,#0 ; INITIALIZE OFFSET
330 0097 58          ; LD B,#HFPTR ; POINTERS FOR
331 0098 9A00        ; LD [B+],#0 ; DTMF SINE WAVE
332 009A A0          ; RC ; TABLE LOOKUP
333 009B 65          ; SWAP A ; QUADRUPLE KEY
334 009C B0          ; RRC A ; VALUE AND ADD
335 009D B0          ; RRC A ; OFFSET FOR KEY
336 009E 94B8        ; ADD A,#0B8 ; TABLE LOOKUP
337 00A0 A6          ; FRLUP: X A,[B] ; LOAD FOUR VALUES
338 00A1 AE          ; LD A,[B] ; FROM ROM KEY
339 00A2 A4          ; LAID ; TABLE INTO LOW
340 00A3 A2          ; X A,[B+] ; FREQ LFTBSZ,
341 00A4 8A          ; INC A ; LFTADR, AND HI
342 00A5 4C          ; IFBNE #0C ; FREQ HFTBSZ,
343 00A6 F9          ; JP FRLUP ; HFTADR
344 00A7 DEEA        ; LD B,#TMRLO ; INITIALIZE TIMER
345 00A9 9A00        ; LD [B+],#0 ; WITH A tC COUNT
346 00AB 9A8C        ; LD [B+],#140 ; EQUIVALENT TO
347 00AD 9A00        ; LD [B+],#0 ; 100 MSEC PLUS
348 00AF 9A8C        ; LD [B+],#140 ; A LUP42 TIME
349 00B1 9A80        ; LD [B+],#080 ; TIMER PWM, NO OUT
350 00B3 9B11        ; LD [B-],#011 ; ENABLE TMR INTRPT
351 00B5 7C          ; SBIT TRUN,[B] ; START TIMER
352 00B6 210F        ; JMP LUP42
353 ;
354 ;
355 ;
356 ;
357 ; TELEPHONE KEY TABLE:
358 ;
359 ; TABLE FORMAT:
360 ; PARAMETER 1: # OF LOW FREQ TABLE VALUES
361 ; PARAMETER 2: BASE ADDR. OF LOW FREQ VALUES
362 ; PARAMETER 3: # OF HIGH FREQ TABLE VALUES
363 ; PARAMETER 4: BASE ADDR. OF HIGH FREQ VALUES
364 ;
365 ; KEY 1
366 00B8 31          ; .BYTE 49,02D,7,07C
367 00B9 2D
368 00BA 07
369 00BB 7C
370 ;
371 ; KEY 2
372 00BC 31          ; .BYTE 49,02D,19,083
373 00BD 2D
374 00BE 13
375 00BF 83
376 ;

```

371		; KEY 3		
372	00C0 31		.BYTE	49,02D,23,096
	00C1 2D			
	00C2 17			
	00C3 96			
373		;		
374		; KEY A		
375	00C4 31		.BYTE	49,02D,21,0AD
	00C5 2D			
	00C6 15			
	00C7 AD			
376		;		
377		; KEY 4		
378	00C8 0B		.BYTE	11,05E,7,07C
	00C9 5E			
	00CA 07			
	00CB 7C			
379		;		
380		; KEY 5		
381	00CC 0B		.BYTE	11,05E,19,083
	00CD 5E			
	00CE 13			
	00CF 83			
382		;		
383		; KEY 6		
384	00D0 0B		.BYTE	11,05E,23,096
	00D1 5E			
	00D2 17			
	00D3 96			
385		;		
386		; KEY B		
387	00D4 0B		.BYTE	11,05E,21,0AD
	00D5 5E			
	00D6 15			
	00D7 AD			
388		;		
389		; KEY 7		
390	00D8 0A		.BYTE	10,069,7,07C
	00D9 69			
	00DA 07			
	00DB 7C			
391		;		
392		; KEY 8		
393	00DC 0A		.BYTE	10,069,19,083
	00DD 69			
	00DE 13			
	00DF 83			
394		;		
395		; KEY 9		
396	00E0 0A		.BYTE	10,069,23,096
	00E1 69			

TL/DD/10740-9

```

00E2 17
00E3 96
397
398 ; KEY C
399 00E4 0A .BYTE 10,069,21,0AD
00E5 69
00E6 15
00E7 AD
400
401 ; KEY *
402 00E8 09 .BYTE 9,073,7,083
00E9 73
00EA 07
00EB 83
403
404 ; KEY 0
405 00EC 09 .BYTE 9,073,19,07C
00ED 73
00EE 13
00EF 7C
406
407 ; KEY #
408 00F0 09 .BYTE 9,073,23,096
00F1 73
00F2 17
00F3 96
409
410 ; KEY D
411 00F4 09 .BYTE 9,073,21,0AD
00F5 73
00F6 15
00F7 AD
412
413 ;
414 ;
415 ;
416 00FF .=-00FF
417 ;
418 00FF BCD01D INTRPT: LD PORTLD,#29 ; BASE LINE VALUE
419 0102 DEEF LD B,#PSW ; 100 MSEC INTERRUPT
420 0104 9B00 LD [B-],#0 ; FROM TIMER
421 0106 9E00 LD [B],#0 ; CLR PSW AND CNTRL
422 0108 DEFD LD B,#SP ; RESTORE STACK
423 010A AE LD A,[B] ; POINTER (SP)
424 010B 8A INC A ; TO ITS VALUE
425 010C 8A INC A ; BEFORE THE
426 010D A6 X A,[B] ; INTERRUPT
427 010E 8E RET ; RETURN FROM
428 ; SUBROUTINE
429 ;
430 ;

```

TL/DD/10740-10

```

431                .FORM
432                ;
433                ; LUP42 CONSISTS OF 42 COP840C INSTRUCTION CYCLE TIMES
434                ; LUP42 TIMING LOOP IS 42 / 0.3579545 = 117 1/3 uSEC
435                ;
436                ;
437 010F 5A        LUP42:  LD      B,#LFPTR
438 0110 AE        LD      A,[B]          ; INCREMENT LOW FREQ
439 0111 8A        INC      A            ; OFFSET POINTER
440 0112 57        LD      B,#LFTBSZ    ; TEST IF LFPTR
441 0113 82        IFEQ    A,[B]        ; BEYOND LIMIT
442 0114 64        CLR      A            ; REINITIALIZE LFPTR
443 0115 5A        LD      B,#LFPTR    ; FOR NEXT TIME
444 0116 A6        X        A,[B]
445 0117 56        LD      B,#LFTADR    ; ADD PTR TO LO FREQ
446 0118 84        ADD      A,[B]        ; TABLE ADDRESS
447 0119 A4        LAID    A,[B]        ; LOW FREQ COMPONENT
448 011A 59        LD      B,#TEMP     ; RESULT TO TEMP
449 011B A2        X        A,[B+]
450 011C AE        LD      A,[B]        ; INCREMENT HI FREQ
451 011D 8A        INC      A            ; OFFSET POINTER
452 011E 55        LD      B,#HFTBSZ   ; TEST IF HFPTR
453 011F 82        IFEQ    A,[B]        ; BEYOND LIMIT
454 0120 64        CLR      A            ; REINITIALIZE HFPTR
455 0121 58        LD      B,#HFPTR    ; FOR NEXT TIME
456 0122 A6        X        A,[B]
457 0123 54        LD      B,#HFTADR   ; ADD PTR TO HI FREQ
458 0124 84        ADD      A,[B]        ; TABLE ADDRESS
459 0125 A4        LAID    A,[B]        ; HI FREQ COMPONENT
460 0126 59        LD      B,#TEMP     ; ADD LOW FREQ VALUE
461 0127 84        ADD      A,[B]        ; TO HI FREQ VALUE
462 0128 9CDO     X        A,PORTLD    ; RESULT TO PORT L
463 012A A4        LAID    A,[B]        ; EQUIVALENT OF
464 012B A4        LAID    A,[B]        ; SIX NOP'S
465 012C E2        JP      LUP42       ; TIMING LOOP OF
466                ; 117 1/3 uSEC
467                ;
468                ;
469                ;
470                ;

```

TL/DD/10740-11

```

471          .FORM
472          ;
473          ; THE FREQUENCY APPROXIMATION IS EQUAL TO THE NUMBER OF
474          ; CYCLES OF SINE WAVE DIVIDED BY THE TIME IN THE TOTAL
475          ; NUMBER OF LUP42 CYCLES BEFORE THE REPETITION OF THE
476          ; ROM TABLE. AS AN EXAMPLE, CONSIDER THE THREE CYCLES
477          ; OF SINE WAVE AND 19 VALUES IN THE ASSOCIATED 1336 HZ
478          ; ROM TABLE. THE 19 CYCLES OF LUP42 TIMES THE LUP42
479          ; TIME OF 117 1/3 USEC IS DIVIDED INTO THE THREE CYCLES
480          ; OF SINE WAVE TO YIELD A VALUE OF 1345.69 HZ AS THE
481          ; 1336 HZ APPROXIMATION.
482          ;
483          ; THE VALUES IN THE ROM TABLES FOR THE DTMF SINE WAVES
484          ; SHOULD WRAP AROUND END TO END IN EITHER DIRECTION TO
485          ; FORM A SYMETRICAL LOOP. THE FIRST VALUE IN THE ROM
486          ; TABLE REPRESENTS THE BASE LINE FOR THAT FREQUENCY.
487          ;
488          ; THE HIGH BAND DTMF FREQUENCIES HAVE A BASE LINE VALUE
489          ; OF 16 AND A MAXIMUM VALUE OF 32. THE LOW BAND DTMF
490          ; FREQUENCIES HAVE A BASE LINE VALUE OF 13 AND A
491          ; MAXIMUM VALUE OF 26. THIS DIFFERENCE IN BASE LINE
492          ; VALUES IS NECESSARY TO SATISFY THE REQUIREMENT OF THE
493          ; HIGH BAND FREQUENCIES NEEDING A LEVEL 2 DB ABOVE THE
494          ; LEVEL OF THE LOW BAND FREQUENCIES TO COMPENSATE FOR
495          ; LOSSES IN TRANSMISSION. THE SUM OF THE TWO BASE LINE
496          ; VALUES YIELDS A BASE LINE VALUE OF 29, WHILE THE SUM
497          ; OF THE TWO MAXIMUM VALUES YIELDS A MAXIMUM VALUE OF
498          ; 58. THUS THE SIX BIT DTMF OUTPUT FROM THE L PORT TO
499          ; THE LADDER NETWORK RANGES FROM 0 TO 58, WITH A BASE
500          ; LINE VALUE OF 29.
501          ;
502          ; THE VALUES IN THE DTMF SINE WAVE TABLES ARE
503          ; CALCULATED BY COMPUTING THE SINE VALUE AT THE
504          ; APPROPRIATE POINTS, SCALING THE SINE VALUE UP TO THE
505          ; BASE LINE VALUE, AND THEN ADDING THE RESULT TO THE
506          ; BASE LINE VALUE. THE FOLLOWING EXAMPLE WILL HELP TO
507          ; CLARIFY THIS CALCULATION.
508          ;
509          ; CONSIDER THE THREE CYCLES OF SINE WAVE ACROSS 19
510          ; DATA POINTS FOR THE 1336 HZ DTMF HIGH BAND FREQUENCY.
511          ; THE FIRST VALUE IN THE TABLE IS THE BASE LINE VALUE
512          ; OF 16. WITH 2 PI RADIANS PER SINE WAVE CYCLE,
513          ; THE SUCCEEDING VALUES IN THE TABLE REPRESENT THE
514          ; SINE VALUES OF 1 X (6 PI / 19), 2 X (6 PI / 19),
515          ; 3 X (6 PI / 19), . . . . , UP TO 18 X (6 PI / 19).
516          ; LET US NOW CONSIDER THE SEVENTH AND EIGHTH VALUES
517          ; IN THE TABLE, REPRESENTING THE SINE VALUES OF
518          ; 6 X (6 PI / 19) AND 7 X (6 PI / 19) RESPECTIVELY.
519          ; THE CALCULATIONS OF 16 X SIN [6 X (6 PI / 19)] AND
520          ; 16 X SIN [7 X (6 PI / 19)] YIELD VALUES OF - 5.20 AND
521          ; 9.83 RESPECTIVELY. ROUNDED TO THE NEAREST INTEGER

```

TL/DD/10740-12

```

522 ; GIVES VALUES OF - 5 AND 10. WHEN ADDED TO THE BASE
523 ; LINE VALUE OF 16, THESE VALUES YIELD THE RESULTS
524 ; 11 AND 26 FOR THE SEVENTH AND EIGHTH VALUES IN THE
525 ; 1336 HZ DTMF TABLE. SYMMETRY IN THE LOOP OF 19 VALUES
526 ; IN THE DTMF TABLE DICTATES THAT THE FOURTEENTH AND
527 ; THIRTEENTH VALUES IN THE TABLE ARE 21 AND 6,
528 ; REPRESENTING VALUES OF 5 AND - 10 FROM THE
529 ; CALCULATIONS.
530 ;
531 ; THE AREA UNDER A HALF CYCLE OF SINE WAVE RELATIVE TO
532 ; THE AREA OF THE SURROUNDING RECTANGLE IS 2/PI, WHERE
533 ; PI RADIANS REPRESENT THE SINE WAVE HALF CYCLE. THIS
534 ; SURROUNDING RECTANGLE HAS A LENGTH OF PI AND A HEIGHT
535 ; OF 1, WITH THE HEIGHT REPRESENTING THE MAXIMUM SINE
536 ; VALUE. CONSEQUENTLY, THE AREA OF THIS SURROUNDING
537 ; RECTANGLE IS PI. THE INTEGRAL OF THE AREA UNDER THE
538 ; HALF SINE WAVE FROM 0 TO PI IS EQUAL TO 2. THE RATIO
539 ; OF 2/PI IS EQUAL TO 63.66 % , SO THAT THE TOTAL OF
540 ; THE VALUES FOR EACH HALF SINE WAVE SHOULD APPROXIMATE
541 ; 63.66 % OF THE SUM OF THE MAX VALUES. THE MAXIMUM
542 ; VALUES (RELATIVE TO THE BASE LINE) ARE 13 AND 16
543 ; RESPECTIVELY, FOR THE LOW AND HIGH BAND FREQUENCIES.
544 ;
545 ;
546 ;
547 ;
548 ;
549 ; LF697: 4 CYCLES OF SINE WAVE SPREAD
550 ; ACROSS 49 TIMING LOOP (LUP42) CYCLES
551 ;
552 ;
553 ;     FREQ. = 4 / (49 X 117 1/3) = 695.73 HZ
554 ;     ERROR = (697 - 695.73) / 697 = - 0.18 %
555 .BYTE 13,19,24,26,25,20,14,7,2,0
012D 0D
012E 13
012F 18
0130 1A
0131 19
0132 14
0133 0E
0134 07
0135 02
0136 00
556 0137 01 .BYTE 1,5,11,18,23,26,25,21,15,9
0138 05
0139 0B
013A 12
013B 17
013C 1A
013D 19
013E 15

```

TL/DD/10740-13


```

013F 0F
0140 09
557 0141 03      .BYTE      3,0,1,4,10,16,22,25,26,23
0142 00
0143 01
0144 04
0145 0A
0146 10
0147 16
0148 19
0149 1A
014A 17
558 014B 11      .BYTE      17,11,5,1,0,3,8,15,21,25
014C 0B
014D 05
014E 01
014F 00
0150 03
0151 08
0152 0F
0153 15
0154 19
559 0155 1A      .BYTE      26,24,19,12,6,1,0,2,7
0156 18
0157 13
0158 0C
0159 06
015A 01
015B 00
015C 02
015D 07

560 ;
561 ;
562 ; LF770:  1 CYCLE OF SINE WAVE SPREAD
563 ;          ACROSS 11 TIMING LOOP (LUP42) CYCLES
564 ;
565 ;          FREQ. = 1 / (11 X 117 1/3) = 774.79 HZ
566 ;          ERROR = (774.79 - 770) / 770 = + 0.62 %
567 ;
568 015E 0D      .BYTE      13,20,25,26,23,17,9,3,0,1
015F 14
0160 19
0161 1A
0162 17
0163 11
0164 09
0165 03
0166 00
0167 01
569 0168 06      .BYTE      6
570 ;

```

TL/DD/10740-14

```

571      ;
572      ; LF852:  1 CYCLE OF SINE WAVE SPREAD
573      ;                ACROSS 10 TIMING LOOP (LUP42) CYCLES
574      ;
575      ;                FREQ. = 1 / (10 X 117 1/3) = 852.27 HZ
576      ;                ERROR = (852.27 - 852) / 852 = + 0.03 %
577      ;
578 0169 0D      .BYTE      13,21,25,26,21,13,5,1,0,5
      016A 15
      016B 19
      016C 1A
      016D 15
      016E 0D
      016F 05
      0170 01
      0171 00
      0172 05

579      ;
580      ;
581      ; LF941:  1 CYCLE OF SINE WAVE SPREAD
582      ;                ACROSS 9 TIMING LOOP (LUP42) CYCLES
583      ;
584      ;                FREQ. = 1 / (9 X 117 1/3) = 946.97 HZ
585      ;                ERROR = (946.97 - 941) / 941 = + 0.63 %
586      ;
587 0173 0D      .BYTE      13,21,26,24,18,8,2,0,5
      0174 15
      0175 1A
      0176 18
      0177 12
      0178 08
      0179 02
      017A 00
      017B 05

588      ;
589      ;
590      ;
591      ; HF1209: 1 CYCLE OF SINE WAVE SPREAD
592      ;                ACROSS 7 TIMING LOOP (LUP42) CYCLES
593      ;
594      ;                FREQ. = 1 / (7 X 117 1/3) = 1217.53 HZ
595      ;                ERROR = (1217.53 - 1209) / 1209 = + 0.71 %
596      ;
597 017C 10      .BYTE      16,29,32,23,9,0,3
      017D 1D
      017E 20
      017F 17
      0180 09
      0181 00
      0182 03

598      ;

```

TL/DD/10740-15

```

599      ;
600      ; HF1336:  3 CYCLES OF SINE WAVE SPREAD
601      ;                ACROSS 19 TIMING LOOP (LUP42) CYCLES
602      ;
603      ;                FREQ. = 3 / (19 X 117 1/3) = 1345.69 HZ
604      ;                ERROR = (1345.69 - 1336) / 1336 = + 0.73 %
605      ;
606 0183 10      .BYTE      16,29,31,19,4,0,11,26,32,24
      0184 1D
      0185 1F
      0186 13
      0187 04
      0188 00
      0189 0B
      018A 1A
      018B 20
      018C 18
607 018D 08      .BYTE      8,0,6,21,32,28,13,1,3
      018E 00
      018F 06
      0190 15
      0191 20
      0192 1C
      0193 0D
      0194 01
      0195 03

608      ;
609      ;
610      ; HF1477:  4 CYCLES OF SINE WAVE SPREAD
611      ;                ACROSS 23 TIMING LOOP (LUP42) CYCLES
612      ;
613      ;                FREQ. = 4 / (23 X 117 1/3) = 1482.21 HZ
614      ;                ERROR = (1482.21 - 1477) / 1477 = + 0.35 %
615      ;
616 0196 10      .BYTE      16,30,29,14,1,4,20,32,26,10
      0197 1E
      0198 1D
      0199 0E
      019A 01
      019B 04
      019C 14
      019D 20
      019E 1A
      019F 0A
617 01A0 00      .BYTE      0,8,24,32,22,6,0,12,28,31
      01A1 08
      01A2 18
      01A3 20
      01A4 16
      01A5 06
      01A6 00

```

TL/DD/10740-16

```

01A7 0C
01A8 1C
01A9 1F
618 01AA 12      .BYTE      18,3,2
01AB 03
01AC 02
619           ;
620           ;
621           ; HF1633:  4 CYCLES OF SINE WAVE SPREAD
622           ;          ACROSS 21 TIMING LOOP (LUP42) CYCLES
623           ;
624           ;
625           ;          FREQ. = 4 / (21 X 117 1/3) = 1623.38 HZ
626           ;          ERROR = (1633 - 1623.38) / 1633 = - 0.59 %
627 01AD 10      .BYTE      16,31,27,9,0,11,29,30,14,0
01AE 1F
01AF 1B
01B0 09
01B1 00
01B2 0B
01B3 1D
01B4 1E
01B5 0E
01B6 00
628 01B7 07      .BYTE      7,25,32,18,2,3,21,32,23,5
01B8 19
01B9 20
01BA 12
01BB 02
01BC 03
01BD 15
01BE 20
01BF 17
01C0 05
629 01C1 01      .BYTE      1
630           ;
631           ;
632           ;

```

TL/DD/10740-17

```

        .FORM
633      ;
634      ;
635      ; DTMF KEYBOARD DECODE SUBROUTINE (KBRDEC)
636      ;
637      ; KEYBOARD INPUT DATA IS IN ACCUMULATOR WITH A
638      ;   LOW TRUE FORMAT AS FOLLOWS:
639      ;   BITS 7 TO 4 : LOW TRUE COLUMN VALUE (E,D,B,7)
640      ;   BITS 3 TO 0 : LOW TRUE ROW VALUE (E,D,B,7)
641      ;
642      ; ASSUMPTION MADE THAT COLUMN STROBES (LOW TRUE) ARE
643      ;   OUTPUT, WHILE ROW VALUES (LOW TRUE) ARE INPUT.
644      ;
645      ; LOW TRUE COLUMN/ROW INPUT DIGIT IN ACCUMULATOR IS
646      ;   TRANSFORMED INTO A DTMF HEX DIGIT KEY VALUE
647      ;
648      ; TABLE LOOKUP TRANSFORMATION CHECKS FOR MULTIPLE KEYS,
649      ;   NO KEY, OR NO COLUMN SELECT, AND THEN PRODUCES
650      ;   A DTMF HEX DIGIT KEY VALUE WITH A BINARY FORMAT
651      ;   OF 000ORRCC FOR A SINGLE KEY INPUT,
652      ;   WHERE   - RR IS LOW BAND (LB) FREQUENCY SELECT
653      ;             - CC IS HIGH BAND (HB) FREQUENCY SELECT
654      ;
655      ; KBRDEC SUBROUTINE IS EXITED WITH A RETURN (RET)
656      ;   COMMAND TO INDICATE MULTIPLE KEYS, NO KEY,
657      ;   OR NO COLUMN SELECT
658      ;
659      ; KBRDEC SUBROUTINE IS EXITED WITH A RETURN AND SKIP
660      ;   (RETSK) COMMAND TO INDICATE A SINGLE KEY ENTRY
661      ;
662      ;
663      0200      . =0200
664      ;
665      ; LOW TRUE TRANSLATION TABLE - ONLY E,D,B,7 ACCEPTABLE
666      ;
667      0200 C0      .BYTE  0C0,0C0,0C0,0C0,0C0,0C0,0C0,0C
        0201 C0
        0202 C0
        0203 C0
        0204 C0
        0205 C0
        0206 C0
        0207 0C
668      0208 C0      .BYTE  0C0,0C0,0C0,8,0C0,4,0,0C0
        0209 C0
        020A C0
        020B 08
        020C C0
        020D 04
        020E 00
        020F C0
669      ;

```

TL/DD/10740-18

```

670      ;
671 0210 5F      KBRDEC: LD      B,#KDATA
672 0211 A6      X      A,[B] ; STORE LOW TRUE
673 0212 AE      LD      A,[B] ; COLUMN/ROW VALUE
674 0213 95F0    AND     A,#0F0 ; EXTRACT LOW TRUE COLUMN
675 0215 65      SWAP   A      ; & PUT IN LOWER NIBBLE
676 0216 A4      LAID   ; 0000CC00 FROM TABLE
677 0217 A0      RC      ; SHIFT TABLE VALUE DOWN
678 0218 B0      RRC    A      ; TWO BITS TO PRODUCE
679 0219 B0      RRC    A      ; 000000CC
680 021A A6      X      A,[B] ; STORE RESULT
681 021B 950F    AND     A,#0F ; EXTRACT LOW TRUE ROW
682 021D A4      LAID   ; 0000RR00 FROM TABLE
683 021E 84      ADD     A,[B] ; ADD TO PRODUCE 0000RRCC
684 021F 930F    IFGT   A,#0F ; RETURN IF MULTIPLE KEYS,
685 0221 8E      RET     ; NO KEYS, OR NO COLUMN
686 0222 8D      RETSK  ; RETURN AND SKIP
687      ;           ; IF SINGLE KEY
688      ;
689      ;
690      ;
691      .END

```

TL/DD/10740-19

B	00FE	BYP1	0072	BYP2	0075	BYPA	0019
BYPB	001B	CNTRL	00EE	DTMFGP	0040	DTMFLP	008E
FINI	0086	FRLUP	00A0	HFPTR	0007	HFTADR	000B
HFTBSZ	000A	INTRPT	00FF *	KBRDEC	0210	KDATA	0000
LFPTR	0005	LFTADR	0009	LFTBSZ	0008	LOOP	0006
LUP	004D	LUP1	006C	LUP2	0078	LUP42	010F
PORTD	00DC	PORTGC	00D5	PORTGD	00D4	PORTI	00D7
PORTLC	00D1	PORTLD	00D0	PSW	00EF	RO	00F0
R1	00F1	R2	00F2	R3	00F3	SP	00FD
START	0000 *	TAUHI	00ED *	TAULO	00EC *	TEMP	0006
TMRHI	00EB *	TMRLO	00EA	TRUN	0004	X	00FC

TL/DD/10740-20

2-Way Multiplexed LCD Drive and Low Cost A/D Converter Using V/F Techniques with COP8 Microcontrollers

National Semiconductor
Application Note 673
Volker Soffel



ABSTRACT

This application note is intended to show a general solution for implementing a low cost A/D and a 2-way multiplexed LCD drive using National Semiconductor's COP840C 8-bit microcontroller. The implementation is demonstrated by means of a digital personal scale. Details and function of the weight sensor itself are not covered in this note. Also the algorithms used to calculate the weight from the measured frequency are not included, as they are too specific and depend on the kind of sensor used.

Typical Applications

- Weighing scales
- Sensors with voltage output
- Capacitive or resistive sensors
- All kinds of measuring equipment
- Automotive test and control systems

Features

- 2-way multiplexed LCD drive capability up to 30 segments (4 digit and 2 dot points)
- Precision frequency measurement
- Low current consumption
- Current saving HALT mode
- Additional computing power for application specific tasks

INTRODUCTION

Today's most popular digital scales all have the following characteristics:

They are battery powered and use a LCD to display the weight. Instead of using a discrete A/D-converter, in many cases a V/F converter is used, which converts an output voltage change of the weight sensor to a frequency change. This frequency is measured by a microcontroller and is used to calculate the weight. The advantages of a V/F over an A/D converter are multifold. Only one line from the V/F to the microcontroller is needed, whereas a parallel A/D needs at least 8 lines or even more (National also offers A/Ds with serial output). A V/F can be constructed very simply using National Semiconductor's low cost, precision voltage to frequency converters LM331 or LM331A. Other possibilities are using Op-amps or a 555-timer in astable mode.

V/F-CONVERSION

Hardware

The basic configuration of the scale described in this application note is shown in *Figure 1*.

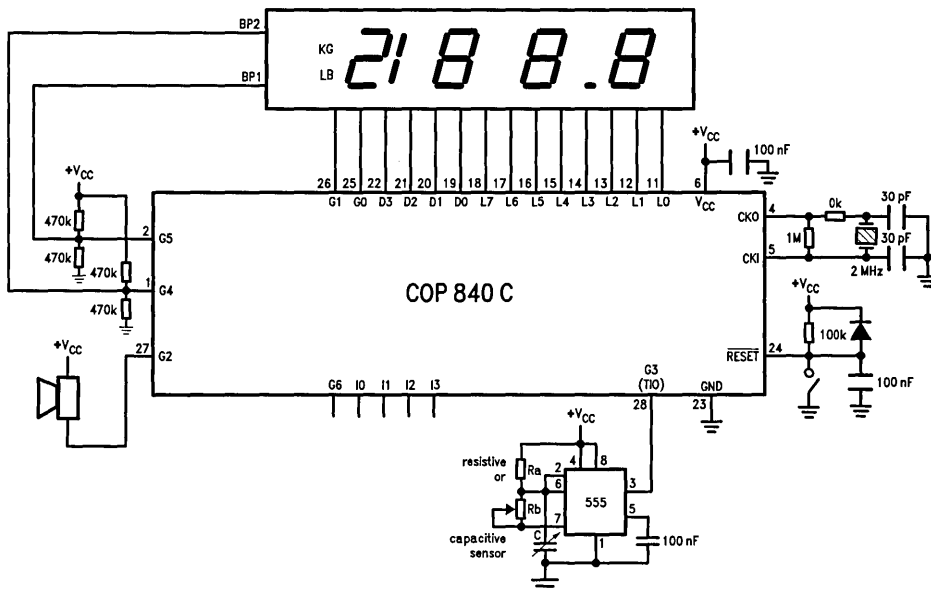


FIGURE 1. System Diagram

TL/DD/10788-1

A capacitive or resistive sensor's weight related capacitance or resistance change is transformed by a 555 timer (in astable mode) to a change of frequency. The output frequency f is determined by the formula:

$$f = 1.44 / ((Ra + 2Rb) * C)$$

The output high time is given by:

$$t1 = 0.693 * (Ra + Rb) * C$$

The output low time is given by:

$$t2 = 0.693 * Rb * C$$

This frequency is measured using the COP800 16-bit timer in the "input capture" mode. After calculation, the weight is displayed on a 2-way multiplexed LCD. Using this configuration a complete scale can be built using only two ICs and a few external passive components.

For more information on V/F converters generally used with voltage output sensors, refer to the literature listed in the reference section.

Frequency Measurement

The COP 16-bit timer is ideally suited for precise frequency measurements with minimum software overhead. This timer has three programmable operating modes, of which the "input capture" mode is used for the frequency measurement. Allocated with the timer is a 16-bit "autoload/capture register". The G3-I/O-pin serves as the timer capture input (TIO). In the "input capture" mode the timer is decremented with the instruction cycle frequency (t_c). Each positive going edge at TIO (also neg. edge programmable) causes the timer value to be copied automatically to the autoload/capture register without stopping the timer or destroying its

contents. The "timer pending" flag (TPND) in the PSW-register is set to indicate a capture has occurred, and if the timer-interrupt is enabled, an interrupt is generated. The frequency measurement routine listed below executes the following operations (refer to the RAM/register definition file listed at the beginning for symbolic names used in the routines):

The timer is preset with FFFF Hex and is started by setting the TRUN bit, after which the software checks the TPND-flag in a loop (timer interrupt is disabled). When the TPND flag is set the first time, the contents of the capture register is saved in RAM locations STALO and STAHI (start value). The TPND pending flag now must be reset by the software. Then, another 255 positive going edges are counted (equal to 255 pulses) before the capture register is saved in RAM locations ENDLO, ENDHI (end value). The shortest time period that can be measured depends on the number of instruction cycles needed to save the capture register, because with the next positive going edge on TIO the contents of the capture register is overwritten (worst case is 18 instruction cycles, which equals a max. frequency of 55.5 kHz at $t_c = 1 \mu s$).

The end-value is subtracted from the start-value and the result is restored in RAM locations STALO, STAHI. This value can then be used to calculate the time period of the frequency applied to TIO (G3) by multiplying it with the t_c -time and dividing the result by the number of pulses measured ($N = 255$).

$$T = (\text{startvalue} - \text{endvalue}) * t_c / N$$

```
;THE FOLLOWING "INCLUDE FILE" IS USED
;AS PART OF THE DEFINITION- AND INITIALIZATION PHASE
;IN COP800 PROGRAMS.
;REGISTER NAMES, CONTROL BITS ETC ARE NAMED IN THE
;SAME WAY IN THE COP800 DATA-SHEETS.
```

```
;      --- COP800 MEMORY MAPPED ---
```

```
; *****
;* PORT -, CONFIGURATION - AND CONTROL REGISTERS *
; *****
```

```
PORTLD = 0D0      ; L-PORT DATA REGISTER
PORTLC = 0D1      ; L-PORT CONFIGURATION
```

TL/DD/10788-2

```

PORTLP = 0D2 ; L-PORT INPUT REGISTER
PORTGD = 0D4 ; G-PORT DATA REGISTER
PORTGC = 0D5 ; G-PORT CONFIGURATION
PORTGP = 0D6 ; G-PORT INPUT REGISTER

PORTD = 0DC ; D-PORT (OUTPUT)
PORTI = 0D7 ; I-PORT (INPUT)

SIOR = 0E9 ; MWIRE SHIFT REGISTER
TMRLO = 0EA ; TIMER LOW-BYTE
TMRHI = 0EB ; TIMER HIGH-BYTE
TAULO = 0EC ; T.-AUTO REG.LOW BYTE
TAUHI = 0ED ; T.-AUTO REG.HIGH BYTE

CNTRL = 0EE ; CONTROL REGISTER
PSW = 0EF ; PSW-REGISTER

```

```
.FORM
```

```
*****
```

```
* CONSTANT DECLARE *
```

```
*****
```

```
--- CONTROL REGISTER BITS ---
```

```

S0 = 00 ; MICROWIRE CLOCK DIVIDE BY
; --- BIT 0 ---
S1 = 01 ; MICROWIRE CLOCK DIVIDE BY
; --- BIT 1 ---
IEDG = 02 ; EXTERNAL INTERRUPT EDGE
; POLARITY SELECT (0=RISING
; EDGE,1=FALLING EDGE)
MSEL = 03 ; ENABLE MICROWIRE FUNCTION
; --- SO AND SK ---
TRUN = 04 ; START/STOP THE TIM/COUNT.
; (1=RUN;0=STOP)
TEDG = 05 ; TIMER INPUT EDGE POL.SEL.
; (0=RIS. EDGE;1=FAL. EDGE)
CSEL = 06 ; SELECTS THE CAPTURE MODE
;
TSEL = 07 ; SELECTS THE TIMER MODE

```

```
--- P S W REGISTER ---
```

```
GIE = 00 ; GLOBAL INTERRUPT ENABLE
```

```
TL/DD/10788-3
```

```

ENI      = 01      ; EXTERNAL INTERRUPT ENABLE
BUSY     = 02      ; MICROWIRE BUSY SHIFTING
IPND     = 03      ; EXTERNAL INTERRUPT PENDING
ENTI     = 04      ; TIMER INTERRUPT ENABLE
TPND     = 05      ; TIMER INTERRUPT PENDING
C        = 06      ; CARRY FLAG
HC       = 07      ; HALF CARRY FLAG

```

```

;****          RAM-DEFINITIONS          ****

```

```

BCDLO    = 000    ;CALCULATED WEIGHT IN BCD
           ;LOW BYTE
BCDHI    = 001    ;CALCULATED WEIGHT IN BCD
           ;HIGH BYTE
MWBUFF0  = 003    ;7SEGMENT DATA FOR LCD DISPL
           ;L-PORT
MWBUFF1  = 004    ;D-PORT
MWBUFF2  = 005    ;G-PORT
OFF1     = 006    ;OFFSET REGISTERS FOR
OFF2     = 007    ;7 SEGMENT CODE TABLE
OFF3     = 008    ;

STALO    = 009    ;START VALUE,LOW BYTE
STAHI    = 00A    ;START VALUE,HIGH BYTE
ENDLO    = 00B    ;END VALUE LOW BYTE
ENDHI    = 00C    ;END VALUE HIGH BYTE

DIV0     = 00D    ;DIVISOR FOR DINBI248 ROUTINE

;022..02F RESERVED FOR STACK WITH COP820
;062..06F RESERVED FOR STACK WITH COP840

```

```

;****          REGISTER DEFINITIONS          ****

```

```

COUNT   = 0F0
COUNT2  = 0F1
COUNT3  = 0F2
FLAG     = 0FF      ;FLAG REGISTER

```

```

;****          BIT DEFINITIONS FLAG REGISTER          ****

```

```

POUND    = 04      ;POUND=1:DISPLAY POUND SEGMENT
           ;POUND=0:DISPLAY kg SEGMENT

```

```

;*****          G-PORT BIT DEFINITIONS          *****

```

```

BP1     = 05      ;BACKPLANE 1

```

BP2 = 04 ;BACKPLANE 2

;TIME OF 255 PULSES, USING TIMER INPUT CAPTURE MODE

FMEAS:

```
                                ;PERIOD TIME=
                                ;(START-ENDVALUE)*tc/255
                                ;DIFFERENCE START-ENDVALUE
                                ;IS STORED IN ENDLO,ENDHI
LD      COUNT,#000              ;LOAD PULSE COUNTER (255 PULSES)
LD      X,#TAULO                ;POINT TO AUTO REG. LOW B.
LD      B,#TMRLO               ;PRESET TIMER
LD      [B+],#0FF              ;REG. WITH FFFFh
LD      [B],#0FF
LD      B,#CNTRL
LD      [B+],#0D0              ;CNTRL-REG.: TIMER CAPTURE
                                ;MODE,TIO POS. TRIGGERED,
                                ;START TIMER
```

```
L1:   RBIT   #TPND,[B]         ;RESET TIMER PENDING FLAG
      IFBIT  #TPND,[B]
      JP     SSTORE
      JP     L1
```

SSTORE: ;STORE START VALUE

```
RBIT   #TPND,[B]
LD      A,[X+]                ;LOAD TIMER CAPTURE REG.
                                ;LOW BYTE
X       A,STALO                ;STORE IN RAM
LD      A,[X-]                ;LOAD HIGH BYTE CAPTURE,
                                ;POINT TO LOW BYTE CAPTURE
X       A,STAH1                ;STORE IN RAM
LD      B,#PSW
```

L256:

```
IFBIT  #TPND,[B]
JP     DCOU
JP     L256
```

```
DCOU:  RBIT   #TPND,[B]         ;RESET TIMER PENDING FLAG
      DRSZ   COUNT              ;DECREMENT PULSE COUNTER
                                ;COUNTER = 0 ?
      JP     L256              ;NO,LOOP 'TIL 255 PULSES
                                ;HAVE BEEN MEASURED
```

ESTORE: ;STORE END VALUE

```
LD      CNTRL,#00             ;STOP TIMER
LD      B,#STALO              ;POINT TO START VALUE LOW BYTE
LD      A,[X+]                ;LOAD END VALUE LOW BYTE
X       A,[B]                 ;LOAD ACCU WITH STARTVALUE LOW BYTE
                                ;& STALO WITH END VALUE LOW BYTE
```

TL/DD/10788-5

```
SC
SUBC   A,[B]                  ;SUBTRACT ENDVALUE LOW BYTE
                                ;FROM STARTVALUE LOW BYTE
X       A,[B+]                ;STORE RESULT IN STALO,
                                ;POINT TO STAH1
LD      A,[X]                 ;LOAD ACCU WITH ENDVALUE HIGH BYTE
X       A,[B]                 ;LOAD ACCU WITH STARTVALUE HIGH BYTE
                                ;& STAH1 WITH ENDVALUE HIGH BYTE
SUBC   A,[B]                  ;SUBTRACT ENDVALUE HIGH BYTE FROM
                                ;STARTVALUE HIGH BYTE
X       A,[B]                 ;STORE RESULT IN STAH1
RET
.END
```

TL/DD/10788-6

2-WAY MULTIPLEXED LCD DRIVE

Today a wide variety of LCDs, ranging from static to multiplex rates of 1:64 are available on the market. The multiplex rate of a LCD can be determined by the number of its backplanes (segment-common plate). The higher the multiplex rate the more individual segments can be controlled using only one line. e.g. a static LCD only has one backplane; only one segment can be controlled with one line. A two-way multiplexed LCD has two backplanes and two segments can be controlled with one line, etc.

Common to all LCDs is the fact that the drive voltage applied to the backplane(s) and segments has to be alternating. DC-components higher than 100 mV can cause electrochemical reactions (refer to manufacturer's spec), which reduce reliability and lifetime of the display.

If the multiplex ratio of the LCD is N and the amount of available outputs is M, the number of segments that can be driven is:

$$S = (M - N) * N$$

So the maximum number of a 2-way mux LCD's segments that can be driven with a COP800 in 28-pin package (if all outputs can be used to drive the LCD) is:

$$S = (18 - 2) * 2 = 32$$

During one LCD refresh cycle tx (typical values for 1/tx = fx are in the range 30 Hz . . . 60 Hz), three different voltages levels: Vop, 0.5*Vop and 0V have to be generated. The "off" voltage across a segment is not 0V as with static LCDs and also the "on" voltage is not Vop, but only a fraction of it. The ratio of "on" to "off" r.m.s.-voltage (discrimination) is determined by the multiplex ratio and the number of voltage levels involved. The most desirable discrimination ratio is one that maximizes the ratio of VON to VOFF, allowing the maximum voltage difference between activated and non-activated states. In general the maximum achievable ratio for any particular value of N is given by:

$$(V_{ON}/V_{OFF})_{max} = \text{SQR}((\text{SQR}(N) + 1)/(\text{SQR}(N) - 1))$$

$$\text{SQR} = \text{square root}$$

Using this formula the maximum achievable discrimination ratio for a 2-way multiplex LCD is 2.41, however, it is also possible to order a customized display with a smaller ratio. For ease of operation, most LCD drivers use equal voltage steps (0V, 0.5 *Vop, Vop). Thus a discrimination ratio of 2.24 is achieved. When using the COP800 to drive a 2-way multiplexed LCD the only external hardware required to achieve the three voltage steps are 4 equal resistors that form two voltage dividers—one for each backplane

(Figure 1). The procedure is to set G4 and G5 to "0" for 0V, to HI-Z (TRI-STATE®) for 0.5*Vop and to "1" in order to establish Vop at the backplane electrodes.

With the COP800 each I/O pin can be set individually to TRI-STATE, "1" or "0", so this procedure can be implemented very easily.

The current consumption of typical LCDs is in the range of 3 µA to 4 µA (at Vop = 4.5V, refresh rate 60 Hz) per square centimeter of activated area. Thus the backplane and segment terminals can be treated as HI-Z loads. At high refresh rates the LCD's current consumption increases dramatically, which is the reason why many LCD manufacturers recommend not using a refresh frequency higher than 60 Hz.

Timing Considerations

As shown in Figures 2 and 3, one LCD refresh cycle tx is subdivided into four equally distant time sections ta, tb, tc and td during which the backplane and segment terminals have to be updated in order to switch a specific segment on or off. Considering a refresh frequency of 50 Hz (tx = 20 ms) ta, tb, tc and td are equal to 5 ms; a COP800 running from an external clock of 2 MHz has an internal instruction cycle time of 5 µs and a typical current consumption of less than 350 µA (at VCC = 3V and room temperature), thus meeting both the requirements of low current consumption and additional computing power between LCD refreshes.

The timing is done using the COP800's 16-bit timer in the PWM autoloader mode. The timer and the assigned 16-bit autoloader register are preset with proper values. By setting the TRUN-flag in the CNTRL-register the timer is decremented each instruction cycle. A flag (TPND) is set at underflow and the timer is automatically reloaded with the value stored in the autoloader-register. Timer underflow can also be programmed to generate an interrupt.

Segment Control

Figure 2 shows the voltage-waveforms applied to the two backplane-electrodes (a) and the waveform at a segment-electrode (b), which is needed to switch segment A on and segment B off. The resulting voltage over the segments (c and d) is achieved by subtracting waveform (b) from BP1 (segment A) and waveform (b) from BP2 (segment B).

Figure 3 shows the four different waveforms which must be generated to meet all possible combinations of two segments connected to the same driving terminal (off-off, on-off, off-on, on-on).

Figure 4 shows the internal segment and backplane connections for a typical 2-way mux LCD.

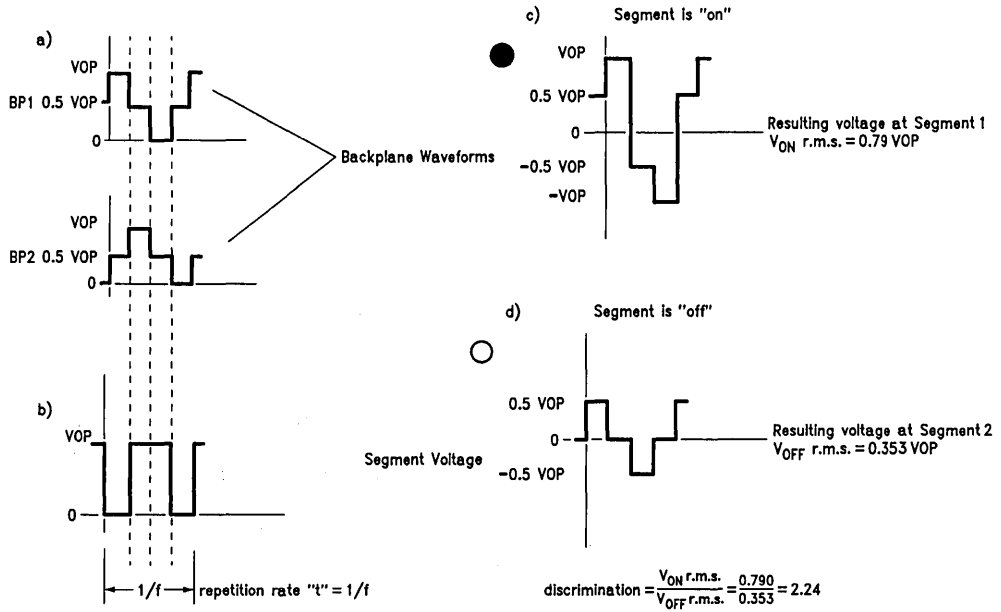


FIGURE 2. LCD Waveforms

TL/DD/10788-7

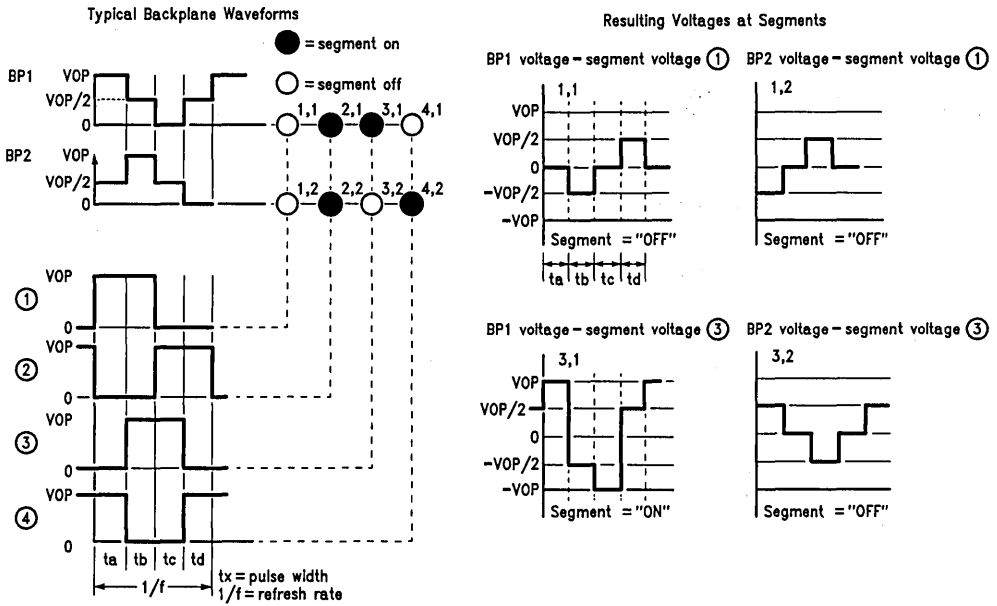


FIGURE 3. Backplane and Segment Voltage Scheme for 1:2 Mux LCD-Drive

TL/DD/10788-8

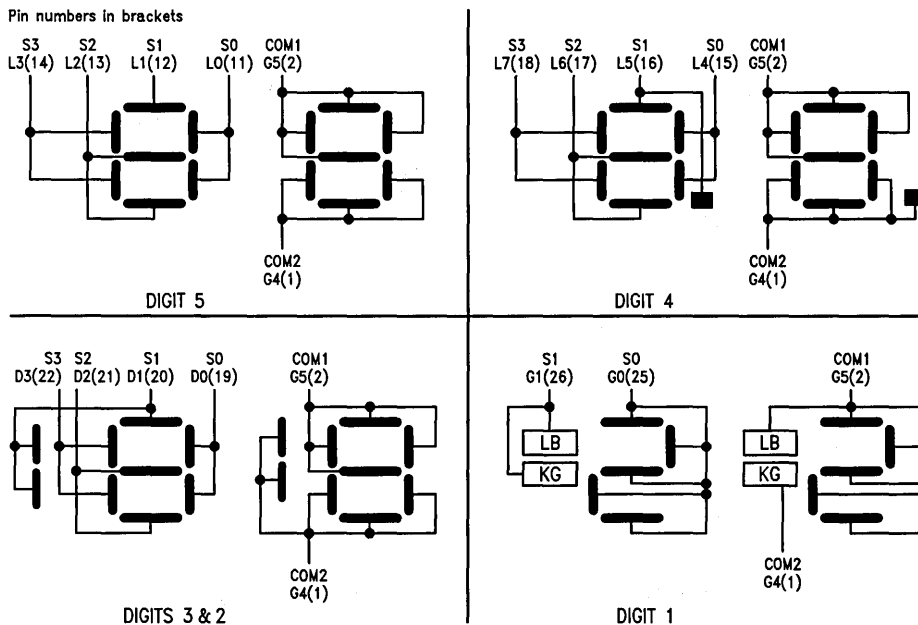


FIGURE 4. Customized LCD Display (Backplane and Segment Organization)

TL/DD/10788-9

LCD Drive Subroutine

The LCD drive subroutine DISPL converts a 16-bit binary value to a 24-bit BCD-value for easier display data fetch. The drive subroutine itself is built up of a main routine doing the backplane refresh and 7 subroutines (SEG0, SEG1, SEG2, SEG3, SEGOUT, TTPND, DISPD). The subroutines SEG0 to SEG4 are used to get the LCD segment data from a look-up table in ROM for time phases ta, tb, tc and td respectively. Subroutine SEGOUT writes the segment data for each time phase to the corresponding output ports. One time phase takes 5 ms, giving a total refresh cycle time of 20 ms (50 Hz). The exact timing is done by using the COP800 16-bit timer in the PWM autoload mode. In that mode the timer is reloaded with the value stored in the autoload register on every timer underflow. At the same time the timer pending flag is set. The subroutine TTPND checks this flag in a loop. If the timer pending flag is set, this subroutine resets it and returns to the calling program. Thus a 5 ms time delay is created before the segment and backplane data for the next time phase is written to the output ports. Finally the subroutine DISPD switches off the LCD by setting the backplane and segment connections to "0". In this digital scale application a frequency measurement is made while the LCD is off. Then the weight is calculated from this frequency and is displayed for 10s. After this 10s the LCD is switched off again and the COP800 is programmed to enter the current saving HALT mode ($I_{DD} < 10 \mu A$). A new weight cycle on the digital scale is initiated by pressing a push button, which causes a reset of the microcontroller.

CONCLUSIONS

National Semiconductor's COP800 Microcontroller family is ideally suited for use with V/F converters and 2-way multiplexed LCDs, as they offer features, which are essential for these types of applications. The high resolution, 3-mode programmable 16-bit timer allows precise frequency measurement in the input capture mode with minimum software overhead. The timer's PWM autoreload mode offers an easy way to implement a precise timebase for the LCD refresh. The COP800's programmable I/O ports provide flexibility in driving 2-way multiplexed LCDs directly. The COP800 family, fabricated using M2CMOS technology, offers both low voltage (min V_{CC} of 2.5V) and low current drain.

REFERENCES

1. National Semiconductor, "Linear Databook 2, Rev. 1" LM331, LM331A datasheets pages 3-285 ff.
2. National Semiconductor, "Linear Applications Databook, 1986", "Versatile monolithic V/Fs can compute as well as convert with high accuracy", pages 1213 ff.
3. National Semiconductor, "Microcontrollers Databook, Rev. 1", COP820C/COP840C datasheets pages 2-7 ff.
4. U. Tietze, Ch. Schenk, "Halbleiter-Schaltungstechnik" 8.Auflage 1986, Springer Verlag, ISBN 0-387-16720-X, "Funktionsgeneratoren mit steuerbarer Frequenz", pages 465 ff, "Multivibratoren", pages 183 ff.
5. Lucid Displays, "LCD design guide", English Electric Valve Company Ltd., Chelmsford, Essex, Great Britain.

APPENDIX—Software Routines

```
;LOOKUP TABLE FOR CUSTOMIZED 2-WAY MUX LCD
```

```
. = X'200 ;START LOOK-UP TABLE AT ROM ADDRESS 200
```

```
;TIMEPHASE Ta 7 SEGMENT DATA
.BYTE 004 ;"0" AND ".0"
.BYTE 00E ;"1" AND ".1"
.BYTE 008 ;"2" AND ".2"
.BYTE 008 ;"3" AND ".3"
.BYTE 002 ;"4" AND ".4"
.BYTE 001 ;"5" AND ".5"
.BYTE 001 ;"6" AND ".6"
.BYTE 00C ;"7" AND ".7"
.BYTE 000 ;"8" AND ".8"
.BYTE 000 ;"9" AND ".9"
.BYTE 00F ;" " AND ". "
```

```
;SPECIAL SEGMENTS TIMPHASE Ta
```

```
.BYTE 001 ;"LB"
.BYTE 000 ;"LB 2"
.BYTE 003 ;"KG"
.BYTE 002 ;"KG 2"
```

```
. = .+ 1
```

```
;TIMEPHASE Tb 7 SEGMENT DATA
```

```
.BYTE 002 ;"0"
.BYTE 00E ;"1"
.BYTE 003 ;"2"
.BYTE 00A ;"3"
.BYTE 00E ;"4"
.BYTE 00A ;"5"
.BYTE 002 ;"6"
.BYTE 00E ;"7"
```

TL/DD/10788-10


```

.BYTE 002 ;"8"
.BYTE 00A ;"9"
.BYTE 00F ;" "
.BYTE 000 ;".0"
.BYTE 00C ;".1"
.BYTE 001 ;".2"
.BYTE 008 ;".3"
.BYTE 00C ;".4"
.BYTE 008 ;".5"
.BYTE 000 ;".6"
.BYTE 00C ;".7"
.BYTE 000 ;".8"
.BYTE 008 ;".9"
.BYTE 00D ;". "

```

```

.LOCAL

```

```

TTPND:

```

```

LD B, #PSW

```

```

$LOOP:

```

```

IFBIT #TPND, [B]

```

```

JP $END

```

```

JP $LOOP

```

```

$END:

```

```

RBIT #TPND, [B]

```

```

LD B, #PORTGD

```

```

RET

```

```

.LOCAL

```

```

. = .+1

```

```

;TIMEPHASE Tc 7 SEGMENT DATA

```

```

.BYTE 00B ;"0" AND ".0"

```

```

.BYTE 001 ;"1" AND ".1"

```

```

.BYTE 007 ;"2" AND ".2"

```

```

.BYTE 007 ;"3" AND ".3"

```

```

.BYTE 00D ;"4" AND ".4"

```

```

.BYTE 00E ;"5" AND ".5"

```

```

.BYTE 00E ;"6" AND ".6"

```

```

.BYTE 003 ;"7" AND ".7"

```

```

.BYTE 00F ;"8" AND ".8"

```

```

.BYTE 00F ;"9" AND ".9"

```

```

.BYTE 000 ;" " AND ". "

```

```

COPY:

```

```

;COPY 2BYTES POINTED TO

```

```

;BY B AND B+1 TO RAM

```

```

;POINTED TO BY X AND X+1

```

```

LD A, [B+]

```

```

X A, [X+]

```

```

LD A, [B+]

```

```

X A, [X+]

```

```

RET

```

```

.LOCAL

```

;TIMEPHASE Td 7 SEGMENT DATA

```
.BYTE 00D ;"0"
.BYTE 001 ;"1"
.BYTE 00C ;"2"
.BYTE 005 ;"3"
.BYTE 001 ;"4"
.BYTE 005 ;"5"
.BYTE 00D ;"6"
.BYTE 001 ;"7"
.BYTE 00D ;"8"
.BYTE 005 ;"9"
.BYTE 000 ;" "
.BYTE 00F ;".0"
.BYTE 003 ;".1"
.BYTE 00E ;".2"
.BYTE 007 ;".3"
.BYTE 003 ;".4"
.BYTE 007 ;".5"
.BYTE 00F ;".6"
.BYTE 003 ;".7"
.BYTE 00F ;".8"
.BYTE 007 ;".9"
.BYTE 002 ;". "
```

;SPECIAL SEGMENTS TIMEPHASE Tb

```
.BYTE 003 ;"LB"
.BYTE 003 ;"LB 2 "
.BYTE 001 ;"KG"
.BYTE 001 ;"KG 2"
```

;SPECIAL SEGMENTS TIMPHASE Tc

```
.BYTE 002 ;"LB"
.BYTE 003 ;"LB 2"
.BYTE 000 ;"KG"
.BYTE 001 ;"KG 2"
```

;SPECIAL SEGMENTS TIMEPHASE Td

```
.BYTE 000 ;"LB"
.BYTE 000 ;"LB 2"
.BYTE 002 ;"KG"
.BYTE 002 ;"KG 2"
```

.END

;DISPL:

;INPUT PARAMETER: COUNT2 =RAM REGISTER, WHICH CONTAINS

;THE DISPLAY TIME IN SEC.

;EXAMPLE COUNT2= 1-> DISPLAY TIME IS 1SEC.

;LCD DRIVE ROUTINE FOR CUSTOMIZED 2 WAY MULTIPLEX

;LCD

```

;ROUTINE CONVERTS BCD DATA STORED IN RAM LOCATIONS
;BCDLO, BCDHI INTO LCD OUTPUT DATA STORED AT
;MWBUF0 = LPORT DATA
;MWBUF1 = DPORT DATA
;MWBUF2 = G-PORT DATA (G0,G1 ONLY, OTHER BITS
;          STAY UNCHANGED)
;SUBROUTINES INCLUDED:
;SEG0: GETS LCD SEGMENT DATA FOR TIMEPHASE TA
;SEG1: GETS LCD SEGMENT DATA FOR TIMEPHASE TB
;SEG2: GETS LCD SEGMENT DATA FOR TIMEPHASE TC
;SEG3: GETS LCD SEGMENT DATA FOR TIMEPHASE TD
;DISPD: SWITCHES THE DISPLAY OFF AND
;        CONFIGURES G-,L- AND D-PORTS
;TTPND: CHECKS TIMER PENDING FLAG (REFRESH
;        RATE GENERATION)
;SEGOUT: OUTPUTS LCD SEGMENT AND BACKPLANE DATA
;SUBROUTINES SEG0... SEG1 MUST FOLLOW DIRECTLY AFTER LOOK-UP
;TABLE, BECAUSE OF THE USE OF THE LAID-INSTRUCTION

```

```
.LOCAL
```

```

SEG0:
    LD     B,#OFF1 ;POINT TO OFFSET 1 REG.
    LD     [B+],#000
    LD     [B+],#000
    LD     A,#00B

$TWO:
    IFBIT  #05,BCDHI ;WEIGHT >= 200 POUNDS?
    INCA   ;YES DISPLAY DIGIT5 ("2")

$POUND:
    IFBIT  #POUND,FLAG
    JP     $LPORT
    ADD    A,#002

$LPORT:
    X      A,[B]
    LD     X,#BCDLO
    LD     B,#MWBUF0
    LD     A,[X]
    AND    A,#00F ;ELIMINATE DIGIT1 BITS
    ADD    A,OFF2
    LAID   ;GET DIGIT1 DATA
    X      A,[B] ;SAVE DIGIT1 DATA
           ;IN MWBUF0
    LD     A,[X+]
    AND    A,#0F0 ;ELIMINATE DIGIT1 BITS
    SWAP   A
    ADD    A,OFF1 ;ALWAYS DISPLAY DECIMAL POINT
    LAID   ;GET DIGIT1 DATA
    SWAP   A
    OR     A,[B] ;STORE DIGIT1 AND
    X      A,[B+] ;DIGIT2 DATA IN MWBUF0

```

```

$DPORT:
  LD      A, [X]
  IFBIT  #04,BCDHI
  JP      $ADD1
  AND    A,#00F
  ADD    A,OFF2 ;DISPLAY NO LEADING ZERO
  JP      $GET

$ADD1:
  AND    A,#00F
  ADD    A,OFF1 ;DISPLAY "1" (DIGIT4)

$GET:
  LAID   ;GET DIGIT3 DATA
  X      A, [B+] ;STORE DIGIT3 DATA IN
          ;MWBUF1

$GPORT:
  LD      A,OFF3
  LAID   ;GET DIGIT5 ("2") AND SPECIAL
          ;SEGMENT DATA
  OR     A,#0FC ;SET BITS 2...7 TO 1
  X      A, [B] ;SAVE DATA IN MWBUF2
  RET

SEG1:
  LD      B,#OFF1
  LD      [B+],#01B
  LD      [B+],#010
  LD      A,#056
  JP      $TWO

SEG2:
  LD      B,#OFF1
  LD      [B+],#030
  LD      [B+],#030
  LD      A,#05A
  JP      $TWO

SEG3:
  LD      B,#OFF1
  LD      [B+],#04B
  LD      [B+],#040
  LD      A,#05E
  JP      $TWO
  .LOCAL

DISPL:
  IFBIT  #POUND,FLAG
  JP      MULT2
  JP      LDT

MULT2:
  LD      B,#BUF12LO ;CALCULATE WEIGHT IN POUNDS
  LD      [B+],#22 ;(Multiplication of kg *2.2)

```

TL/DD/10788-14

```

LD      X, #STALO
JSR     MULBI168
LD      B, #BUF12LO
JSR     COPY
LD      STAHI+1, #00
LD      DIV0, #10
JSR     DIVBI248

LDT:
JSR     BINBCD16      ;CONVERT BINARY TO BCD WEIGHT
LD      COUNT, #50   ;REPEAT DISPLAY LOOP 50 TIMES
                        ; (=1 SEC DISPLAY TIME)

LD      B, #TMRLO
LD      [B+], #0E8   ;LOAD TIMER WITH 1000(03E8h)
LD      [B+], #003   ; (=50 Hz LCD REFRESH AT tc=5us)
LD      [B+], #0E8   ;LOAD AUTOREG. WITH 1000
LD      [B+], #003
LD      [B+], #090   ;CNTRL-REG.: "TIMER WITH AUTO-
                        ;LOAD"- MODE, START TIMER
LD      [B+], #010   ;PSW-REG.: RESET TPND FLAG

DISP1:
JSR     SEG0          ;GET 7-SEGM. DATA FOR REFRESH
                        ;TIMEPHASE Ta
JSR     TTPND         ;TEST TIMER PENDING FLAG
                        ;BACKPLANE REFRESH Ta

TP0:
SBIT    #BP1, [B]
LD      A, [B+]      ;POINT TO G-CONFIG.-REG.
RBIT    #BP2, [B]
SBIT    #BP1, [B]
LD      A, [B-]      ;POINT TO G-DATA REG.
RBIT    #BP2, [B]
JSR     SEGOUT       ;SEGMENT DATA OUT
JSR     SEG1         ;GET 7-SEG. DATA FOR Tb
JSR     TTPND

TP1:
SBIT    #BP2, [B]
LD      A, [B+]      ;POINT TO G-CONF.-REG.
RBIT    #BP1, [B]
SBIT    #BP2, [B]
LD      A, [B-]      ;POINT TO G-DATA REG.
RBIT    #BP1, [B]
JSR     SEGOUT
JSR     SEG2         ;GET 7-SEGM. DATA FOR Tc
JSR     TTPND

TP2:
RBIT    #BP1, [B]
LD      A, [B+]      ;POINT TO G-CONFIG.-REG.
RBIT    #BP2, [B]
SBIT    #BP1, [B]
LD      A, [B-]      ;POINT TO G-DATA-REG.
RBIT    #BP2, [B]
JSR     SEGOUT

```

```

        JSR      SEG3
        JSR      TTPND

TP3:
        RBIT    #BP1, [B]
        RBIT    #BP2, [B]
        LD      A, [B+]
        RBIT    #BP1, [B]
        SBIT    #BP2, [B]
        JSR      SEGOUT
        DRSZ    COUNT
        JP      DISP1
        LD      COUNT, #50
        DRSZ    COUNT2      ;10SEC OVER?
        JP      DISP1      ;NO, DISPLAY WEIGHT
        JSR      DISPD
        RET                    ;YES ROUTINE FINISHED

DISPD:
                                ;SWITCH DISPLAY OFF
        LD      B, #PORTLD
        LD      [B+], #000    ;OUTPUT 0 TO L PORT
        LD      [B+], #0FF    ;L-PORT = OUTPUT PORT
        LD      B, #PORTGD
        LD      [B+], #000    ;OUTPUT 0 TO G OUTPUTS
        LD      [B+], #037    ;G0..G2, G4, G5=OUTPUTS
        LD      PORTD, #000   ;OUTPUT 0 TO D-PORT
        RET

SEGOUT:
        LD      B, #MWBUFF0
        LD      A, [B+]      ;POINT TO MWBUF1
        X      A, PORTLD    ;OUTPUT 7 SEG. DATA IN
                                ;MWBUFF0 TO L-PORT
        LD      A, [B+]      ;POINT TO MWBUF2
        X      A, PORTD     ;OUTPUT MWBUF1 TO D-PORT
        LD      X, #PORTGD
        LD      A, [X]
        AND    A, [B]       ;AND MWBUF2 WITH PORTGD
                                ;LEAVE BITS 2...7 UNCHANGED
        X      A, [B]       ;STORE RESULT (A') IN
                                ;MWBUFF2, LOAD A WITH
                                ;ORIGINAL MWBUF2 VALUE
        AND    A, #003      ;AND 007 WITH ORIGINAL
                                ;MWBUFF2 (A''), SET BITS 0, 1 TO
                                ;CORRECT VALUE
        OR     A, [B]       ;OR A' WITH A'', RESTORE ORIGINAL
                                ;G2...G7 BITS
        X      A, [X]       ;OUTPUT RESULT TO G-PORT
        RET

```

TL/DD/10788-16

;16 BIT BINARY TO BCD CONVERSION
 ;THE MEMORY ASSIGNMENTS ARE AS FOLLOWS:

;BINLO: RAM ADDRESS BINARY LOW BYTE
 ;BCDLO: RAM ADDRESS BCD LOW BYTE
 ;COUNT: RAM ADDRESS SHIFT COUNTER (0F0...0FB,0FF)

;BCD NUMBER IN BCDLO,BCDLO+1,BCDLO+2

```

;
;MEMORY ADDRESS      M(BINLO+1)  M(BINLO)
;DATA                BINARY HB   BINARY LOW BYTE
;
;MEMORY ADDRESS      M(BCDLO+2)  M(BCDLO+1)  M(BCDLO)
;DATA                BCD HB      BCD          BCD LOW BYTE
;

```

```

BINLO = STALO
.LOCAL
$BCDT = (BCDLO + 3) & 0F
$BINT = (BINLO + 2) & 0F

```

```

BINBCD:
LD      COUNT, #16 ;LOAD CONTROL REGISTER WITH
           ;NUMBER OF LEFTSHIFTS TO
           ;EXECUTE
LD      B, #BCDLO ;LOAD BCD-NUMBER LOWEST BYTE
           ;ADDRESS

```

```

$CBCD:
LD      [B+], #00 ;CLEAR BCD RAM-REGISTERS
IFBNE  #$BCDT
JP      $CBCD

```

```

$LSH:
LD      B, #BINLO ;LEFTSHIFT BINARY NUMBER
RC

```

```

$LSHFT:
LD      A, [B]
ADC     A, [B] ;IF MSB IS SET, SET CARRY
X      A, [B+]
IFBNE  #$BINT
JP      $LSHFT
LD      B, #BCDLO

```

```

$BCDADD:
LD      A, [B]
ADD     A, #066 ;ADD CORRECTION FACTOR
ADC     A, [B] ;LEFTSHIFT BCD NUMBER
           ;(BCD=2**WEIGHT OF
           ;BINARY BIT(=CARRY BIT))
DCOR   A ;DECIMAL CORRECT ADDITION
X      A, [B+]
IFBNE  #$BCDT

```

```

JP      $BCDADD
DRSZ   COUNT ;DECREMENT SHIFT COUNTER
JP      $LSH
RET
.LOCAL

```

TL/DD/10788-17

TL/DD/10788-18

```

;BINARY DIVIDE 24BIT BY 8BIT (Q=Y/Z)
;YL: LOW BYTE RAM ADDRESS DIVIDEND
;ZL: LOW BYTE RAM ADDRESS DIVISOR
;CNTR: RAM ADDRESS SHIFT COUNTER (0F0...0FB,0FF)

```

```

;QUOTIENT AT RAM LOCATIONS YL..YL+2
;REMAINDER AT YL+3
;QUOTIENT IS ALL '1's IF DIVIDE BY ZERO, REMAINDER
;THEN CONTAINS YL

```

```

;THE MEMORY ASSIGNMENTS ARE AS FOLLOWS:

```

```

;
;      M(YH+1)  M(YH)           M(YL+1)  M(YL)
;      0        Y(HIGH BYTE)    Y        Y(LOW BYTE)
;-----
;      M(ZL)
;      Z
;

```

```

;ROUTINE NEEDS 1.21ms FOR EXECUTION AT tc=1us

```

```

ZL      =  DIV0
YL      =  STALO
CNTR    =  COUNT
.LOCAL
$YH     =  YL+2
$BTY    =  ($YH&00F)+2 ;PARAMETER FOR "IFBNE"-INSTR.

```

```

DIVBI248:

```

```

LD      CNTR,#018 ;INITIALIZE SHIFT COUNTER
LD      B,$YH+1  ;FOR 24 COUNTS
LD      [B],#000 ;PUT 0 IN M(YH+1)
LD      X,$YH+1

```

```

$LSHFT:

```

```

LD      B,$YL   ;LEFT SHIFT DIVIDEND
RC

```

```

$LUP:

```

```

LD      A,[B]
ADC     A,[B]
X       A,[B+]
IFBNE  #$BTY
JP      $LUP
LD      B,$ZL
IFC
JP      $SUBT

```

```

$TSUBT:

```

```

SC      ;SUBTRACT AND TEST
LD      A,[X] ;SUBTRACT Z FROM M(YH+1,YH+2)
SUBC   A,[B]
IFNC
JP      $TEST

```



```

$SUBT:                                ;SUBTRACT Z FROM M(YH+1,YH+2)
      LD      A, [X]
      SUBC   A, [B]
      X      A, [X]
      LD      B, #YL
      SBIT   #0, [B]
$TEST:
      DRSZ   CNTR      ;24 SHIFTS EXECUTED?
      JP     $LSHFT   ;NO, LEFT SHIFT DIVIDEND
      RET
      .LOCAL

```

```

;BINARY MULTIPLIES A 16BIT VALUE (X1)
;WITH A 8BIT VALUE (X2): M = X1 * X2

```

```

;X1L: RAM ADDRESS X1 LOW BYTE
;X2L: RAM ADDRESS X2
;COUNT RAM ADDRESS SHIFT COUNTER

```

```

;M IS STORED AT RAM ADDRESSES X2L...X2L+2

```

```

;THE MEMORY ASSIGNMENTS ARE AS FOLLOWS:
;MEMORY      M(X2L+2)  M(X2L+1)      M(X2L)
;DATA        0         0              X2

```

```

-----
;MEMORY      M(X1L+1)  M(X1L)
;DATA        X1(H.B.)  X1(LOW BYTE)

```

```

;THE EXECUTION TIME FOR THE ROUTINE AT tc=1us IS 240us
;

```

```

      .LOCAL

```

```

MULBI168:

```

```

      LD      COUNT, #9 ;PRESET SHIFT COUNTER
      LD      [B+], #00 ;PRESET X2L+1, X2L+2 WITH '0'
      LD      [B], #00
      RC
$LOOP:
      LD      A, [B]    ;RIGHT SHIFT
      RRCA
      X      A, [B-]
      LD      A, [B]
      RRCA
      X      A, [B-]
      LD      A, [B]
      RRCA
      X      A, [B+]

```

```
LD      A, [B+] ; INCREMENT B POINTER
IFNC    ; MOST SIGN. BIT OF X2 SET?
JP      $TEST  ; NO, TEST SHIFT COUNTER
RC      ; YES, RESET CARRY
LD      A, [B-] ; POINT TO 2nd HIGHEST BYTE
        ; OF RESULT
LD      A, [X+] ; DO WEIGHTED ADD
ADC     A, [B]
X       A, [B+]
LD      A, [X-]
ADC     A, [B]
X       A, [B]
```

\$TEST:

```
DRSZ    COUNT ; 8 RIGHT SHIFTS EXECUTED?
JP      $LOOP  ; NO, SHIFT
RET     ; YES, MULTIPLICATION FINISHED
.LOCAL
.END
```

TL/DD/10788-21

PC[®] MOUSE Implementation Using COP800

National Semiconductor
Application Note 681
Alvin Chan



ABSTRACT

The mouse is a very convenient and popular device used in data entry in desktop computers and workstations. For desktop publishing, CAD, paint or drawing programs, using the mouse is inevitable. This application note will describe how to use the COP822C microcontroller to implement a mouse controller.

INTRODUCTION

Mouse Systems was the first company to introduce a mouse for PCs. Together with Microsoft and Logitech, they are the most popular vendors in the PC mouse market. Most mainstream PC programs that use pointing devices are able to support the communication protocols laid down by Mouse Systems and Microsoft.

A typical mouse consists of a microcontroller and its associated circuitry, which are a few capacitors, resistors and transistors. Accompanying the electronics are the mechanical parts, consisting of buttons, roller ball and two disks with slots. Together they perform several major functions: motion detection, host communication, power supply, and button status detection.

MOTION DETECTION

Motion detection with a mouse consists of four commonly known mechanisms. They are the mechanical mouse, the opto-mechanical mouse, the optical mouse and the wheel mouse.

The optical mouse differs from the rest as it requires no mechanical parts. It uses a special pad with a reflective surface and grid lines. Light emitted from the LEDs at the bottom of the mouse is reflected by the surface and movement is detected with photo-transistors.

The mechanical and the opto-mechanical mouse use a roller ball. The ball presses against two rollers which are connected to two disks for the encoding of horizontal and vertical motion. The mechanical mouse has contact points on the disks. As the disks move they touch the contact bars,

which in turn generates signals to the microcontroller. The opto-mechanical mouse uses disks that contain evenly spaced slots. Each disk has a pair of LEDs on one side and a pair of photo-transistors on the other side.

The wheel mouse has the same operation as the mechanical mouse except that the ball is eliminated and the rollers are rotated against the outside surface on which the mouse is placed.

HOST COMMUNICATION

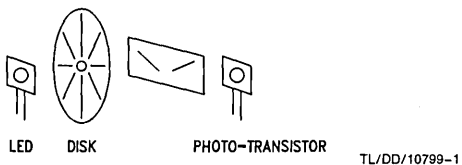
Besides having different operating mechanisms, the mouse also has different modes of communication with the host. It can be done through the system bus, the serial port or a special connector. The bus mouse takes up an expansion slot in the PC. The serial mouse uses one of the COM ports.

Although the rest of this report will be based on the opto-mechanical mouse using the serial port connection, the same principle applies to the mechanical and the wheel mouse.

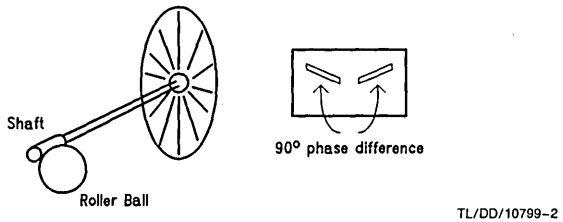
MOTION DETECTION FOR THE OPTO-MECHANICAL MOUSE

The mechanical parts of the opto-mechanical mouse actually consist of one roller ball, two rollers connected to the disks and two pieces of plastic with two slots on each one for LED light to pass through. The two slots are cut so that they form a 90 degree phase difference. The LEDs and the photo-transistors are separated by the disks and the plastic. As the disks move, light pulses are received by the photo-transistors. The microcontroller can then use these quadrature signals to decode the movement of the mouse.

Figure 1a shows the arrangement of the LEDs, disks, plastic and photo-transistors. The shaft connecting the disk and the ball is shown separately on *Figure 1b*. *Figure 2* shows the signals obtained from the photo-transistors when the mouse moves. The signals will not be exactly square waves because of unstable hand movements.

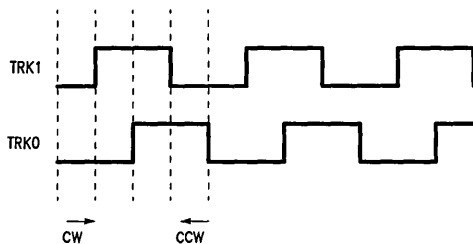
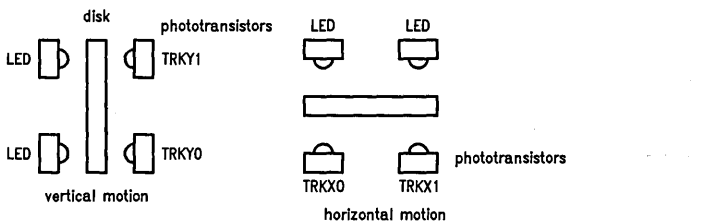


a



b

FIGURE 1



TL/DD/10799-4

Signals at phototransistors are similar for vertical and horizontal motion.
Track 1 leads track 0 by 90 degrees

FIGURE 2

RESOLUTION, TRACKING SPEED AND BAUD RATE

The resolution of the mouse is defined as the number of movement counts the mouse can provide for each fixed distance travelled. It is dependent on the physical dimension of the ball and the rollers. It can be calculated by measuring the sizes of the mechanical parts.

An example for the calculation can be shown by making the following assumptions:

- The disks have 40 slots and 40 spokes
- Each spoke has two data counts
(This will be explained in the section "An Algorithm for Detecting Movements")
- Each slot also has two data counts
- The roller has a diameter of 5mm

For each revolution of the roller, there will be $40 \times 2 \times 2 = 160$ counts of data movement. At the same time, the mouse would have travelled a distance of $\pi \times 5 = 15.7\text{mm}$. Therefore the resolution of the mouse is $15.7/160 = 0.098\text{mm}$ per count. This is equivalent to 259 counts or dots per inch (dpi).

The tracking speed is defined as the fastest speed that the mouse can move without the microcontroller losing track of the movement. This depends on how fast the microcontroller can sample the pulses from the photo-transistors. The effect of a slow tracking speed will contribute to jerking movements of the cursor on the screen.

The baud rate is fixed by the software and the protocol of the mouse type that is being emulated. For mouse systems and microsoft mouse, they are both 1200. Baud rate will affect both the resolution and the tracking speed. The internal movement counter may overflow while the mouse is still sending the last report with a slow baud rate. With a fast baud rate, more reports can be sent for a certain distance moved and the cursor should appear to be smoother.

POWER SUPPLY FOR THE SERIAL MOUSE

Since the serial port of the PC has no power supply lines, the RTS, CTS, DTR and DSR RS232 interface lines are

utilized. Therefore the microcontroller and the mouse hardware should have very little power consumption. National Semiconductor's COP822C fits into this category perfectly. The voltage level in the RS232 lines can be either positive or negative. When they are positive, the power supply can be obtained by clamping down with diodes. When they are negative, a 555 timer is used as an oscillator to transform the voltage level to positive. The 1988 National Semiconductor Linear 3 Databook has an example of how to generate a variable duty cycle oscillator using the LMC555 in page 5-282.

While the RTS and DTR lines are used to provide the voltage for the mouse hardware, the TXD line of the host is utilized as the source for the communication signals. When idle, the TXD line is in the mark state, which is the most negative voltage. A prp transistor can be used to drive the voltage of the RXD pin to a voltage level that is compatible with the RS232 interface standard.

AN ALGORITHM FOR DETECTING MOVEMENTS

The input signal of the photo-transistors is similar to that shown in *Figure 2*. Track 1 leads track 0 by 90 degrees. Movement is recorded as either of the tracks changes state. State tables can be generated for clockwise and counter-clockwise motions.

With the two tracks being 90 degrees out of phase, there could be a total of four possible track states. It can be observed that the binary values formed by combining the present and previous states are unique for clockwise and counter-clockwise motion. A sixteen entry jump table can be formed to increment or decrement the position of the cursor. If the value obtained does not correspond to either the clockwise or counter-clockwise movement, it could be treated as noise. In that case either there is noise on the microcontroller input pins or the microcontroller is tracking motions faster than the movement of the mouse. A possible algorithm can be generated as follows. The number of instruction cycles for some instructions are shown on the left.

(TRK1, TRK0) _t (TRK1, TRK0) _{t-1} Binary Value				(TRK1, TRK0) _t (TRK1, TRK0) _{t-1} Binary Value					
CCW				CW					
0	1	0	0	4	1	0	0	0	8
1	1	0	1	D	0	0	0	1	1
1	0	1	1	B	0	1	1	1	7
0	0	1	0	2	1	1	1	0	E

```

CYCLES      ;*****
;           SAMPLE SENSOR INPUT
;           INC OR DEC THE POSITION
;*****
;
;
SENSOR:
1           LD           B, #GTEMP
3           LD           A, PORTGP
1           RRC          A
2           AND          A, #03C           ; G6, G5, G4, G3
1           X           A, [B]           ; (GTEMP)
;
2           LD           A, [B+]           ; (GTEMP) X IN 3, 2
1           RRC          A
1           RRC          A
2           AND          A, #03
1           OR           A, [B]           ; (TRACKS)
2           OR           A, #0B0           ; X MOVEMENT TABLE
3           JID
;
NOISEX: JP   YDIR
;
3           INCX: LD      A, XINC
1           INC          A
3           JP           COMX
;
DECX: LD      A, XINC
DEC          A
COMX:
2           IFEQ        A, #080
1           JP          YDIR
3           X           A, XINC
1           LD          B, #CHANGE
1           SBIT        RPT, [B]
1           LD          B, #TRACKS
;
YDIR:
2           LD          A, [B-]           ; (TRACKS) Y IN 5, 4
1           SWAP        A
1           RRC          A
1           RRC          A
1           RRC          A
2           AND          A, #0C0
1           OR           A, [B]           ; (GTEMP)

```

```

1          SWAP      A
2          OR        A, #0CO          ; Y MOVEMENT TABLE
3          JID
;
; NOISEY: JP        ESENS
;
3          INCY:    LD        A, YINC
1          INC      A
3          JP        COMY
;
; DECY:          LD        A, YINC
;              DEC      A
;
; COMY:          IFEQ     A, #080
2          JP        ESENS
1          X         A, YINC
3          LD        B, #CHANGE
1          SBIT     RPT, [B]
1          LD        B, #GTEMP
;
; ESENS:        LD        A, [B+]          ; (GTEMP) IN5, 4, 1, 0
2          X         A, [B]          ; (TRACKS) NEW TRACK STATUS
1          RET
;
;
; . = OBO
; MOVEMX:
; .ADDR        NOISEX          ; 0
; .ADDR        INCX           ; 1
; .ADDR        DECX           ; 2
; .ADDR        NOISEX          ; 3
; .ADDR        DECX           ; 4
; .ADDR        NOISEX          ; 5
; .ADDR        NOISEX          ; 6
; .ADDR        INCX           ; 7
; .ADDR        INCX           ; 8
; .ADDR        NOISEX          ; 9
; .ADDR        NOISEX          ; A
; .ADDR        DECX           ; B
; .ADDR        NOISEX          ; C
; .ADDR        DECX           ; D
; .ADDR        INCX           ; E
; .ADDR        NOISEX          ; F
;
;
; . = OCO
; MOVEMY:
; .ADDR        NOISEY          ; 0
; .ADDR        INCY           ; 1
; .ADDR        DECY           ; 2
; .ADDR        NOISEY          ; 3
; .ADDR        DECY           ; 4
; .ADDR        NOISEY          ; 5
; .ADDR        NOISEY          ; 6
; .ADDR        INCY           ; 7
; .ADDR        INCY           ; 8
; .ADDR        NOISEY          ; 9
; .ADDR        NOISEY          ; A
; .ADDR        DECY           ; B
; .ADDR        NOISEY          ; C
; .ADDR        DECY           ; D
; .ADDR        INCY           ; E
; .ADDR        NOISEY          ; F

```

Going through the longest route in the sensor routine takes 75 instruction cycles. So at 5 MHz the microcontroller can track movement changes within 150 μ s by using this algorithm.

MOUSE PROTOCOLS

Since most programs in the PC support the mouse systems and microsoft mouse, these two protocols will be discussed here. The protocols are byte-oriented and each byte is framed by one start-bit and two stop-bits. The most commonly used reporting mode is that a report will be sent if there is any change in the status of the position or of the buttons.

MICROSOFT COMPATIBLE DATA FORMAT

	Bit							
	6	5	4	3	2	1	0	Number
1	L	R	Y7	Y6	X7	X6	X5	Byte 1
0	X5	X4	X3	X2	X1	X0	Y0	Byte 2
0	Y5	Y4	Y3	Y2	Y1	Y0		Byte 3

L, R = Key data (Left, Right key) 1 = key depressed

X0-X7 = X distance 8-bit two's complement value -128 to +127

Y0-Y7 = Y distance 8-bit two's complement value -128 to +127

Positive = South

In the Microsoft Compatible Format, data is transferred in the form of seven-bit bytes. Y movement is positive to the south and negative to the north.

FIVE BYTE PACKED BINARY FORMAT (MOUSE SYSTEMS CORP)

	Bit								
	7	6	5	4	3	2	1	0	Number
1	0	0	0	0	L*	M*	R*		Byte 1
X7	X6	X5	X4	X3	X2	X1	X0		Byte 2
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0		Byte 3
X7	X6	X5	X4	X3	X2	X1	X0		Byte 4
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0		Byte 5

L*, M*, R* = Key data (Left, Middle, Right key), 0 = key depressed

X0-X7 = X distance 8-bit two's complement value -127 to +127

Y0-Y7 = Y distance 8-bit two's complement value -127 to +127

In the Five Byte Packed Binary Format data is transferred in the form of eight-bit bytes (eight data bits without parity). Bytes 4 and 5 are the movement of the mouse during the transmission of the first report.

THE COP822C MICROCONTROLLER

The COP822C is an 8-bit microcontroller with 20 pins, of which 16 are I/O pins. The I/O pins are separated into two ports, port L and port G. Port G has built-in Schmitt-triggered inputs. There is 1k of ROM and 64 bytes of RAM. In the mouse application, the COP822C's features used can be summarized below. Port G is used for the photo-transistor's input. Pin G0 is used as the external interrupt input to monitor the RTS signal for the microsoft compatible protocol. The internal timer can be used for baud rate timing and interrupt generation. The COP822C draws only 4 mA at a crystal frequency of 5 MHz. The instruction cycle time when operating at this frequency is 2 μ s.

A MOUSE EXAMPLE

The I/O pins for the COP822C are assigned as follows:

Pin	Function
G0	Interrupt Input (Monitoring RTS Toggle)
G1	Reserved for Input Data (TXD of Host)
G2	Output Data (RXD of Host)
G3-G6	LED Sensor Input
L0-L2	Button Input
L3	Jumper Input (for Default Mouse Mode)

The timer is assigned for baud rate generation. It is configured in the PWM auto-reload mode (with no G3 toggle output) with a value of 1A0 hex in both the timer and the auto-reload register. When operating at 5 MHz, it is equivalent to 833 μ s or 1200 baud. When the timer counts down, an interrupt is generated and the service routine will indicate in a timer status byte that it is time for the next bit. The subroutine that handles the transmission will look at this status byte to send the data.

The other interrupt comes from the G0 pin. This is implemented to satisfy the microsoft mouse requirement. As the RTS line toggles, it causes the microcontroller to be interrupted. The response to the toggling is the transmission of the character "M" to indicate the presence of the mouse.

The main program starts by doing some initializations. Then it loops through four subroutines that send the report, sense the movement, sense the buttons, and set up the report format.

Subroutine "SDATA" uses a state table to determine what is to be transmitted. There are 11 or 12 states because microsoft has only 7 data bits and mouse systems has 8. The state table is shown below:

SENDST	State
0	IDLE
1	START BIT
2-8	DATA (FOR MICROSOFT)
2-9	DATA (FOR MOUSE SYSTEMS)
9-10	STOP BIT (FOR MICROSOFT)
10-11	STOP BIT (FOR MOUSE SYSTEMS)
11	NEXT WORD (FOR MICROSOFT)
12	NEXT WORD (FOR MOUSE SYSTEMS)

The G2 pin is set to the level according to the state and the data bit that is transmitted.

Subroutine "SENSOR" checks the input pins connected to the LEDs. The horizontal direction is checked first followed by the vertical direction. Two jump tables are needed to decode the binary value formed by combining the present and previous status of the wheels. The movements are recorded in two counters.

Subroutines "BUTUS" and "BUTMS" are used for polling the button input. They compare the button input with the value polled last time and set up a flag if the value changes. Two subroutines are used for the ease of setting up reports for different mice. The same applies for subroutines "SRPTMS" and "SRPTUS" which set up the report format for transmission. The status change flag is checked and the report is formatted according to the mouse protocol. The

movement counters are then cleared. Since the sign of the vertical movement of mouse systems and microsoft is reversed, the counter value in subroutine "SRPTMS" is complemented to form the right value.

There is an extra subroutine "SY2RPT" which sets up the last two bytes in the mouse systems' report. It is called after the first three bytes of the report are sent.

The efficiency of the mouse depends solely on the effectiveness of the software to loop through sensing and transmission subroutines. For the COP822C, one of the most effective addressing modes is the B register indirect mode.

It uses only one byte and one instruction cycle. With autoincrement or autodecrement, it uses one byte and two instruction cycles. In order to utilize this addressing mode more often, the organization of the RAM data has to be carefully thought out. In the mouse example, it can be seen that by placing related variables next to each other, the saving of code and execution time is significant. Also, if the RAM data can fit in the first 16 bytes, the load B immediate instruction is also more efficient. The subroutine "SRPTMS" is shown below and it can be seen that more than half the instructions are B register indirect which are efficient and compact.

```

;
;   VARIABLES
;
WORDPT = 000 ;WORD POINTER
WORD1  = 001 ;BUFFER TO STORE REPORTS
WORD2  = 002
WORD3  = 003
CHANGE = 004 ;MOVEMENT CHANGE OR BUTTON PRESSED
XINC   = 005 ;X DIRECTION COUNTER
YINC   = 006 ;Y DIRECTION COUNTER
NUMWORD = 007 ;NUMBER OF BYTES TO SEND
SENDST = 008 ;SERIAL PROTOCOL STATE
;
;*****
;   SUBROUTINE SET UP REPORT 'SRPT' FOR MOUSE SYSTEMS
;   CHANGE OF STATUS DETECTED
;   SET UP THE FIRST 3 WORDS FOR REPORTING
;   IF IN IDLE STATE
;*****
;
SRPTMS:
    LD     A, CHANGE
    IFEQ  A, #0 ; EXIT IF NO CHANGE
    RET

;
    RBIT  GIE, PSW ; DISABLE INTERRUPT
    LD    B, #WORDPT
    LD    [B+], #01 ; (WORDPT) SET WORD POINTER
    LD    A, BUTSTAT
    X     A, [B+] ; (WORD1)
;
    LD    A, XINC
    X     A, [B+] ; (WORD2)
;
    SC
    CLR  A
    SUBC A, YINC ; FOR MOUSE SYSTEM NEG Y
    X     A, [B+] ; (WORD3)
;
    RBIT  RPT, [B] ; (CHANGE) RESET CHANGE OF STATUS
    SBIT  SYRPT, [B] ; (CHANGE)
    LD    A, [B+] ; INC B
    LD    [B+], #0 ; (XINC)
    LD    [B+], #0 ; (YINC)
;
    LD    [B+], #03 ; (NUMWORD) SEND 3 BYTES
    LD    [B], #01 ; (SENDST) SET TO START BIT STATE
;
    SBIT  GIE, PSW ; ENABLE INTERRUPT
    RET
;

```

CONCLUSION

The COP822C has been used as a mouse controller. The code presented is a minimum requirement for implementing a mouse systems and microsoft compatible mouse. About 550 bytes of ROM code has been used. The remaining ROM area can be used for internal diagnostics and for communicating with the host's mouse driver program. The unused I/O pins can be used to turn the LED's on only when necessary to save extra power. This report demonstrated the use of the efficient instruction set of the COP800 family. It can be seen that the architecture of the COP822C is most suitable for implementing a mouse controller. The table below summarizes the advantages of the COP822C.

Feature	Advantage
Port G	Schmitt Triggered Input for Photo-Transistors
G0	External Interrupt for RTS Toggling
Timer	For Baud Rate Generation
Low Power	4 mA at 5 MHz
Small Size	20-Pin DIP

REFERENCE

The mouse still reigns over data entry—Electronic Engineering Times, October 1988.

MICE for mainstream applications—PC Magazine, August 1987.

Logimouse C7 Technical Reference Manual—Logitech, January 1986.

APPENDIX A—MEMORY UTILIZATION**RAM Variables**

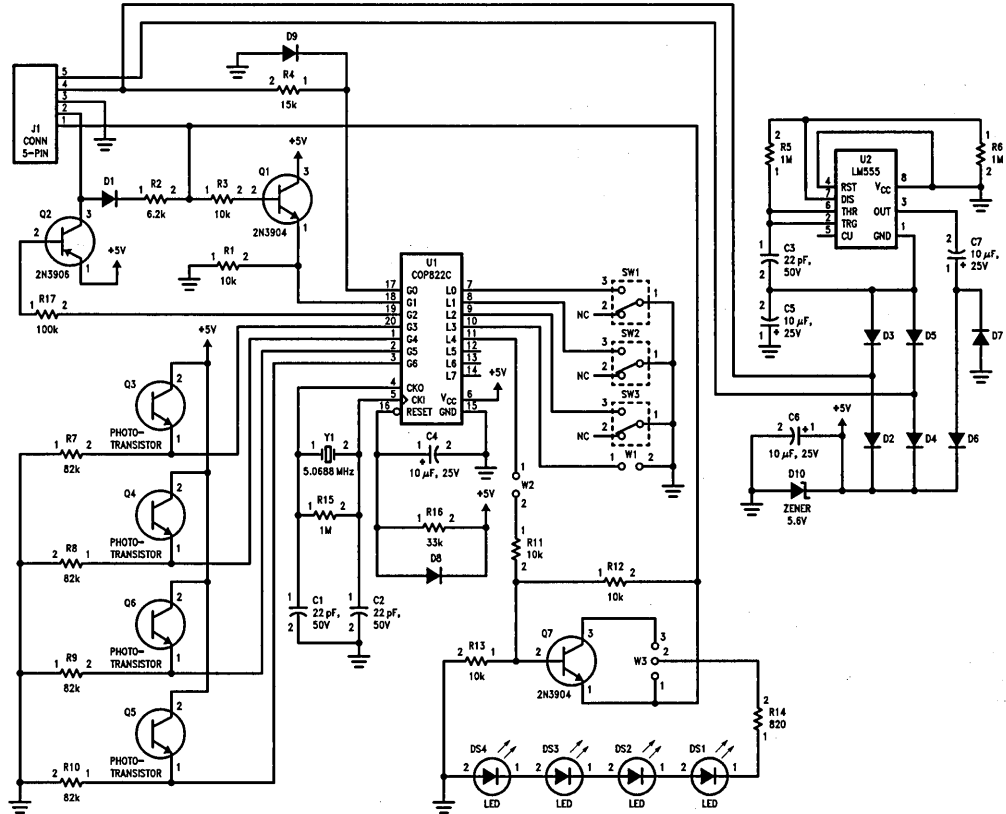
TEMP	= 0F1	Work Space
ASAVE	= 0F4	Save A Register
PSSAVE	= 0F6	Save PSW Register
WORDPT	= 000	Word Pointer
WORD1	= 001	Buffer to Store Report
WORD2	= 002	Buffer
WORD3	= 003	Buffer
CHANGE	= 004	Movement or Button Change
XINC	= 005	X Direction Counter
YINC	= 006	Y Direction Counter
NUMWORD	= 007	Number of Bytes to Send
SENDST	= 008	Serial Protocol State
TSTATUS	= 00A	Counter Status
MTYPE	= 00B	Mouse Type
GTEMP	= 00C	Track Input from G Port
TRACKS	= 00D	Previous Track Status
BTEMP	= 00E	Button Input from L Port
BUTSTAT	= 00F	Previous Button Status

APPENDIX B—SUBROUTINE SUMMARY

Subroutine	Location	Function
MLOOP	03D	Main Program Loop
SENSOR	077	Sample Photo-Transistor Input
INTRP	0FF	Interrupt Service Routines
SRPTUS	136	Set Up Report for Microsoft
SRPTMS	16C	Set Up 1st 3 Bytes Report for Mouse Systems
SDATA	191	Drive Data Transmission Pin According to Bit Value of Report
SY2RPT	1D1	Set Up Last 2 Bytes Report for Mouse Systems
BUTUS	200	Sample Button Input for Microsoft
BUTMS	210	Sample Button Input for Mouse Systems

APPENDIX C—SYSTEM SCHEMATIC, SYSTEM

Flowchart, complete program listing.

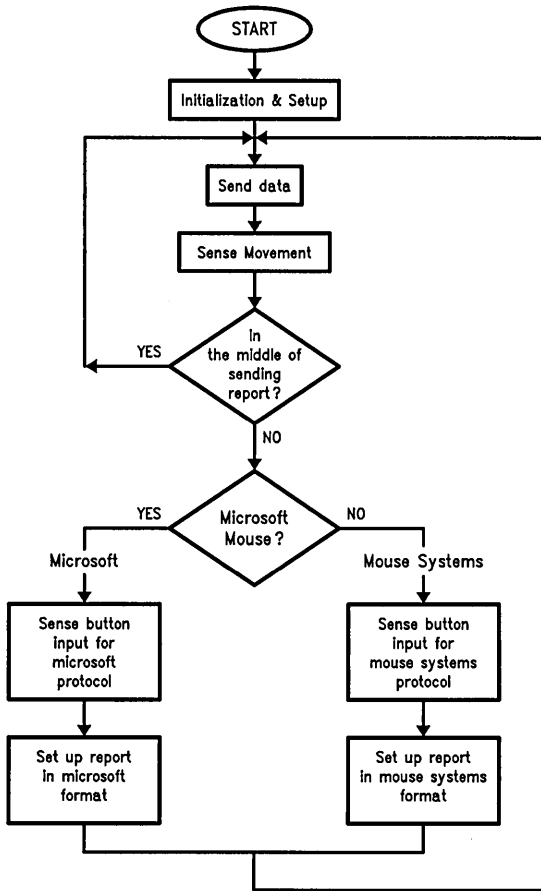


- Note 1:** All diodes are 1N4148.
- Note 2:** All resistor values are in ohms, 5%, 1/8W.
- Note:** Unless otherwise specified

FIGURE 3. System Schematic

TL/DD/10799-5

Flowchart for Mouse Systems and Microsoft Mouse



TL/DD/10789-6

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88
 AMOUSE

```

1          ;
2          ;      MICROSOFT AND MOUSE SYSTEM COMPATIBLE MOUSE
3          ;      02/14/89
4          ;      NAME : AMOUSE.MAC
5          ;
6          ;      .TITLE AMOUSE
7          ;      .CHIP 820
8          ;
9          ;
10         00D0      PORTLD =      0D0      ; PORT L DATA
11         00D1      PORTLC =      0D1      ; PORT L CONFIG
12         00D2      PORTLP =      0D2      ; PORT L PIN
13         ;
14         00D4      PORTGD =      0D4      ; PORT G DATA
15         00D5      PORTGC =      0D5      ; PORT G CONFIG
16         00D6      PORTGP =      0D6      ; PORT G PIN
17         ;
18         00EA      TMRLO =      0EA      ; TIMER LOW BYTE
19         00EB      TMRHI =      0EB      ; TIMER HIGH BYTE
20         00EC      TAULO =      0EC      ; TIMER REGISTER LOW BYTE
21         00ED      TAUHI =      0ED      ; TIMER REGISTER HIGH BYTE
22         ;
23         00EE      CNTRL =      0EE      ; CONTROL REGISTER
24         00EF      PSW   =      0EF      ; PSW REGISTER
25         ;
26         ;      CONSTANT DECLARE
27         ;
28         0000      INTR  =      0
29         0003      TIO   =      3
30         0004      SO    =      4
31         0005      SK    =      5
32         0006      SI    =      6
33         0007      CKO   =      7
34         ;
35         0007      TSEL  =      7
36         0006      CSEL  =      6
37         0005      TEDG  =      5
38         0004      TRUN  =      4
39         0003      MSEL  =      3
40         0002      IEDG  =      2
41         0001      S1    =      1
42         0000      S0    =      0
43         ;
44         0007      HCARRY =      7
45         0006      CARRY =      6
46         0005      TPND  =      5
47         0004      ENTI  =      4
48         0003      IPND  =      3
49         0002      BUSY  =      2
50         0001      ENI   =      1
51         0000      GIE   =      0

```

TL/DD/10799-7

```

52      ;
53      ;
54      ;      TSTATUS BITS
55      0002      TBAUB      =      2      ;BAUD RATE TIMER BIT
56      ;
57      0000      RPT      =      0      ;REPORT BIT OF CHANGE (CHANGE)
58      0001      SYRPT     =      1      ;SET UP MOUSE SYSTEM LAST 2 WORDS (CHANGE)
59      0007      USOFT     =      7      ;MICROSOFT (MTYPE)
60      0002      XMT      =      2      ;G2 AS XMT BIT (PORTGD)
61      ;
62      0003      SW       =      3      ;SLIDE SWITCH (PORTLP,MTYPE)
63      ;
64      ;      REGISTER ASSIGNMENTS
65      ;
66      00F0      RSVF     =      0F0
67      00F1      TEMP     =      0F1
68      00F3      TBAU     =      0F3      ;BAUD RATE TIMER
69      00F4      ASAVE    =      0F4      ;SAVE A
70      00F5      BSAVE    =      0F5      ;SAVE B
71      00F6      PSSAVE   =      0F6      ;SAVE PSW
72      ;
73      ;      VARIABLES
74      ;
75      0000      WORDPT   =      000      ;WORD POINTER
76      0001      WORD1    =      001      ;BUFFER TO STORE REPORTS
77      0002      WORD2    =      002
78      0003      WORD3    =      003
79      ;
80      0004      CHANGE   =      004      ;MOVEMENT CHANGE OR BUTTON PRESSED
81      0005      XINC     =      005      ;X DIRECTION COUNTER
82      0006      YINC     =      006      ;Y DIRECTION COUNTER
83      0007      NUMWORD  =      007      ;NUMBER OF BYTES TO SEND
84      0008      SENDST   =      008      ;SERIAL PROTOCOL STATE
85      ;
86      0009      TBAUR    =      009      ;BAUD RATE TIMER RELOAD
87      000A      TSTATUS  =      00A      ;COUNTER STATUS
88      000B      MTYPE    =      00B      ;MOUSE TYPE
89      ;
90      000C      GTEMP    =      00C      ;TRACK INPUT FROM G
91      000D      TRACKS   =      00D      ;PREVIOUS TRACK STATUS
92      ;
93      000E      BTEMP    =      00E      ;BUTTON INPUT
94      000F      BUTSTAT  =      00F      ;PREVIOUS BUTTON STATUS
95      ;
96      ;
97      ; MOST POSITIVE = SPACE = HI = ON = 0 = START BIT = RBIT
98      ; MOST NEGATIVE = MARK = LO = OFF = 1 = STOP BIT = SBIT
99      ;
100     ;      MICROSOFT FORMAT
101     ;
102     ;      1 L R Y7 Y6 X7 X6

```

TL/DD/10799-8

```

103      ;      0 X5 ..... X0
104      ;      0 Y5 ..... Y0
105      ;
106      ;      1200 BAUD 7 BIT NO PARITY 2 STOP BITS
107      ;
108      ;      MOUSE SYSTEMS FORMAT (FIVE BYTE PACKED BINARY)
109      ;
110      ;      1 0 0 0 0 L* M* R*
111      ;      X7 ..... X0
112      ;      Y7 ..... Y0
113      ;      X7 ..... X0
114      ;      Y7 ..... Y0
115      ;
116      ;      1200 BAUD 7 BIT NO PARITY 2 STOP BITS
117      ;
118      ;      G6,G5,G4,G3 ARE SENSOR INPUTS
119      ;
120      ;      L0, L1 AND L2 ARE BUTTON INPUTS
121      ;
122      ;      G0 IS INTERRUPT INPUT FOR DETECTING RTS TOGGLE
123      ;
124      ;      USE G2 AS TRANSMIT
125      ;
126      ;      G1 USED FOR RECEIVING COMMANDS FROM HOST (RESERVED)
127      ;
128      ;      START:
129 0000 DD2F      LD      SP,#02F
130 0002 BCEF00    LD      PSW,#0      ;DISABLE INTR
131 0005 BCEE80    LD      CNTRL,#080    ;10000000 - AUTORELOAD
132              ;              ;RISING EDGE EXT INT
133 0008 BCD504    LD      PORTGC,#004    ;G2 AS OUTPUT, OTHERS AS HI-Z
134 000B BCD404    LD      PORTGD,#004    ;G2 DATA 1 "MARK"
135              ;
136 000E BCD130    LD      PORTLC,#030    ;HI-Z INPUTS FOR L6-7,OUTPUT L4,5
137 0011 BCD00F    LD      PORTLD,#0F      ;WEAK PULL UP FOR L0-3
138              ;
139              ;      INIT RAM
140              ;
141 0014 5B        LD      B,#CHANGE
142 0015 9A00      LD      [B+],#0      ;(CHANGE)
143 0017 9A00      LD      [B+],#0      ;(XINC)
144 0019 9A00      LD      [B+],#0      ;(YINC)
145 001B BC0A00    LD      TSTATUS,#0
146              ;
147 001E 9DD6      LD      A,PORTGE
148 0020 B0        RRC      A
149 0021 953C      AND     A,#03C      ;NOW IN 6,5,4,3
150 0023 9C0D      X        A,TRACKS    ;GET INITIAL VALUE OF SENSORS
151              ;
152 0025 3067      JSR     SELECT      ;SELECT MOUSE TYPE
153              ;

```

TL/DD/10799-9

```

154 ;*****
155 ;
156 ; CRYSTAL FREQ = 4.96 MHZ 2.016 US INST CYCLE
157 ; FOR 1200 BAUD - TIMER = 413 COUNT
158 ;
159 ;*****
160 ;
161 ;
162 ; LTIMER:
162 0027 DEEA LD B,#TMRL0
163 0029 9A9D LD [B+],#09D ;FOR 2.016 US CYCLE
164 002B 9A01 LD [B+],#01
165 002D 9A9D LD [B+],#09D
166 002F 9E01 LD [B],#01
167 ;
168 0031 BC0800 LD SENDST,#0 ;SET TO IDLE STATE
169 0034 9DEF LD A,PSW
170 0036 9713 OR A,#013 ;ENABLE INTRIS SET GIE
171 0038 9CEF X A,PSW
172 003A BDEE7C SBIT TRUN,CNTRL ;START TIMER
173 ;
174 ; MLOOP:
175 003D BCD03F LD PORTLD,#03F ;TURN ON LED (NOT USED)
176 0040 3191 JSR SDATA
177 0042 3077 JSR SENSOR
178 0044 9D08 LD A,SENDST ;IF SENDING REPORT
179 0046 9300 IFGT A,#0 ;JUST DO SENSOR
180 0048 F4 JP MLOOP
181 ;
182 0049 9DD2 LD A,PORTLP ;GET INPUT FROM BUTTONS (L0,L1,L2)
183 004B B0 RRC A ;PUT IN CARRY FOR CHECKING
184 004C 51 LD B,#BTEMP ;PREPARATION TO SEE WHAT BUTTON IS PRESSED
185 ;
186 004D B0B77 IFBIT USOFT,MTYPE
187 0050 0B JP LPUS
188 ;
189 0051 3210 JSR BUTMS ;MOUSE SYSTEMS
190 0053 316C JSR SRPTMS
191 ;
192 0055 BDD273 IFBIT SW,PORTLP
193 0058 E4 JP MLOOP ;CONTINUE IF NO CHANGE IN SWITCH
194 0059 306B JSR USM ;ELSE NEW SET UP
195 005B E1 JP MLOOP
196 ;
197 005C 3200 LPUS: JSR BUTUS ;MICROSOFT
198 005E 3136 JSR SRPTUS
199 ;
200 0060 BDD273 IFBIT SW,PORTLP
201 0063 3071 JSR SYM ;IF CHANGED IN SWITCH, NEW SET UP
202 0065 203D + JP MLOOP
203 ;
204 ;*****

```

TL/DD/10799-10


```

205 ; SELECT MOUSE TYPE
206 ;*****
207 ;
208 SELECT:
209 0067 B0D213 IFBIT SW,PORTLP ;CHECK JUMPER
210 006A 06 JP SYM
211 ;
212 USM:
213 006B 54 LD B,#MYPE
214 006C 7F SBIT USOFT,[B] ;(MYPE) IS MICROSOFT MOUSE
215 006D BC0F87 LD BUTSTAT,#087 ;NO KEY PRESSED
216 0070 8E RET
217 ;
218 SYM:
219 0071 54 LD B,#MYPE
220 0072 6F RBIT USOFT,[B] ;(MYPE) IS MOUSE SYSTEMS
221 0073 BC0F00 LD BUTSTAT,#0 ;NO KEY PRESSED
222 0076 8E RET
223 ;
224 ;*****
225 ; SAMPLE SENSOR INPUT
226 ; INC OR DEC THE POSITION
227 ; -127 IS USED INSTEAD OF -128 IN CHECKING
228 ; NEGATIVE GOING POSITION SO THAT BOTH
229 ; MICROSOFT AND MOUSE SYSTEMS FIT IN
230 ;*****
231 ;
232 SENSOR:
233 0077 53 LD B,#GTEMP
234 0078 9DD6 LD A,PORTCP
235 007A BCD00F LD PORTLD,#0F ;(NOT USED) TURN OFF LED
236 007D B0 RRC A
237 007E 953C AND A,#03C ;G5,G4,G3,G2
238 0080 A6 X A,[B] ;(GTEMP)
239 ;
240 ;
241 ; (TRK1,TRK0)t-1 (TRK1,TRK0)t
242 ; CCW 0 1 0 0 4
243 ; 1 1 0 1 D
244 ; 1 0 1 1 B
245 ; 0 0 1 0 2
246 ;
247 ; CW 1 0 0 0 8
248 ; 0 0 0 1 1
249 ; 0 1 1 1 7
250 ; 1 1 1 0 E
251 ;
252 0081 AA LD A,[B+] ;(GTEMP) X IN 3,2
253 0082 B0 RRC A
254 0083 B0 RRC A
255 0084 9503 AND A,#03 ;GET X TRACKS

```

TL/DD/10799-11

```

256 0086 87          OR   A, [B]          ;OVERLAY WITH PREVIOUS (TRACKS)
257 0087 97B0        OR   A, #0B0        ;X MOVEMENT TABLE
258 0089 A5          JID
259
;
260 008A 0F          NOISEX: JP   YDIR
261
;
262                INCX:
263 008B 9D05        LD   A, XINC
264 008D 8A          INC   A
265 008E 03          JP   COMX          ;CHECK IF LIMIT IS REACHED
266
DECX:
267 008F 9D05        LD   A, XINC
268 0091 8B          DEC   A
269
COMX:
270 0092 9250        IFEQ  A, #80          ;CHECK FOR LIMIT
271 0094 05          JP   YDIR          ;YES DO NOTHING
272 0095 9C05        X     A, XINC      ;ELSE NEW POSITION
273 0097 58          LD   B, #CHANGE
274 0098 78          SBIT  RPT, [B]     ;(CHANGE)
275 0099 52          LD   B, #TRACKS
276
;
277                YDIR:
278 009A 52          LD   B, #TRACKS
279 009B AB          LD   A, [B-]       ;(TRACKS) Y IN 5,4
280 009C 65          SWAP  A
281 009D B0          RRC   A
282 009E B0          RRC   A
283 009F B0          RRC   A
284 00A0 95C0        AND  A, #0C0
285 00A2 87          OR   A, [B]       ;(GTEMP)
286 00A3 65          SWAP  A
287 00A4 97C0        OR   A, #0C0      ;Y MOVEMENT TABLE
288 00A6 A5          JID
289
;
290                .:=0B0
291
MOVEMX:
292 00B0 8A          .ADDR NOISEX      ;0
293 00B1 8F          .ADDR DECX        ;1
294 00B2 8B          .ADDR INCX        ;2
295 00B3 8A          .ADDR NOISEX      ;3
296 00B4 8B          .ADDR INCX        ;4
297 00B5 8A          .ADDR NOISEX      ;5
298 00B6 8A          .ADDR NOISEX      ;6
299 00B7 8F          .ADDR DECX        ;7
300 00B8 8F          .ADDR DECX        ;8
301 00B9 8A          .ADDR NOISEX      ;9
302 00BA 8A          .ADDR NOISEX     ;A
303 00BB 8B          .ADDR INCX        ;B
304 00BC 8A          .ADDR NOISEX     ;C
305 00BD 8B          .ADDR INCX        ;D
306 00BE 8F          .ADDR DECX        ;E

```

TL/DD/10799-12

```

307 00BF 8A      .ADDR NOISEX      ;F
308              ;
309      00C0      .=-OC0
310      MOVEMY:
311 00C0 D0      .ADDR NOISEY      ;0
312 00C1 D1      .ADDR INCY      ;1
313 00C2 D5      .ADDR DECY      ;2
314 00C3 D0      .ADDR NOISEY      ;3
315 00C4 D5      .ADDR DECY      ;4
316 00C5 D0      .ADDR NOISEY      ;5
317 00C6 D0      .ADDR NOISEY      ;6
318 00C7 D1      .ADDR INCY      ;7
319 00C8 D1      .ADDR INCY      ;8
320 00C9 D0      .ADDR NOISEY      ;9
321 00CA D0      .ADDR NOISEY      ;A
322 00CB D5      .ADDR DECY      ;B
323 00CC D0      .ADDR NOISEY      ;C
324 00CD D5      .ADDR DECY      ;D
325 00CE D1      .ADDR INCY      ;E
326 00CF D0      .ADDR NOISEY      ;F
327              ;
328 00D0 0F      NOISEY: JP      ESENS
329              ;
330 00D1 9D06     INCY: LD      A, YINC
331 00D3 8A      INC      A
332 00D4 03      JP      COMY
333              ;
334 00D5 9D06     DECY: LD      A, YINC
335 00D7 8B      DEC      A
336              ;
337 00D8 9280     COMY:  IFEQ   A, #080
338 00DA 05      JP      ESENS
339 00DB 9C06     X      A, YINC
340 00DD 5B      LD      B, #CHANGE
341 00DE 78      SBIT   RPT, [B]      ; (CHANGE)
342 00DF 53      LD      B, #GTEMP
343              ;
344              ;
345 00E0 53      ESENS: LD      B, #GTEMP
346 00E1 AA      LD      A, [B+]      ; (GTEMP) IN 5,4,1,0
347 00E2 A6      X      A, [B]      ; (TRACKS)NEW TRACK STATUS
348 00E3 8E      RET
349              ;
350              ;
351      00FF      .=-OFF
352              ;
353              ;*****
354              ; INTERRUP ROUTINES
355              ;*****
356              ;
357 00FF 9CF4     INTRP: X      A, ASAVE

```

TL/DD/10789-13

```

358 ;
359 0101 BDEF75 IFBIT TPND,PSW
360 0104 07 JP TINTR
361 0105 BDEF73 IFBIT IPND,PSW
362 0108 0A JP XINTR
363 ;
364 INTRET: ;INTERRUPT RETURN
365 0109 9DF4 LD A,ASAVE
366 010B 8F RETI
367 ;
368 ;*****
369 ; TIMER INTERRUPT
370 ; UPDATE ALL THE COUNTERS
371 ;*****
372 ;
373 TINTR:
374 010C BDEF6D RBIT TPND,PSW
375 010F BDDA7A SBIT TBAUS,TSTATUS ;SET BIT IN TSTATUS
376 0112 F6 JP INTRET
377 ;
378 ;*****
379 ; EXTERNAL INTERRUPT
380 ; RESPONSE TO RTS TOGGLING
381 ; BY SENDING AN 'M' 40H
382 ;*****
383 ;
384 0113 BDEF6B XINTR: RBIT IPND,PSW
385 0116 BDD877 IFBIT USOFT,MTYPE ;ONLY IF MICROSOFT PROTOCOL
386 0119 01 JP XINTR1 ;CONTINUE
387 011A EE JP INTRET ;ELSE DO NOTHING
388 ;
389 011B BC01FF XINTR1: LD WORD1,#0FF ;ALL MARK
390 011E BC024D LD WORD2,#'M'
391 0121 BC0702 LD NUMWORD,#02
392 ;
393 0124 9D08 LD A,SENDST
394 0126 9200 IFEQ A,#0 ;IF IDLE, SEND 'M'
395 0128 05 JP RTSR2
396 ;
397 0129 BC0001 LD WORDPT,#WORD1 ;FAKE CONTINUE LAST CHAR
398 012C 2109 + JP INTRET
399 ;
400 RTSR2:
401 012E BC0002 LD WORDPT,#WORD2 ;'M' ONLY
402 0131 BC0801 LD SENDST,#01
403 0134 2109 + JP INTRET
404 ;
405 ;*****
406 ; SUBROUTINE SET UP REPORT 'SRPT' FOR MICROSOFT
407 ; -----
408 ; CHANGE OF STATUS DETECTED

```

TL/DD/10799-14

```

409      ;      SET UP THE 3 WORDS FOR REPORTING IF IN IDLE STATE
410      ;*****
411      ;
412      SRPTUS:
413 0136 5B      LD      B, #CHANGE
414 0137 70      IFBIT  RPT, [B]
415 0138 01      JP      SROS1
416 0139 8E      RET      ;EXIT IF NOT CHANGE
417      ;
418      SROS1:
419 013A BDEF68  RBIT  GIE, PSW      ;DISABLE INTERRUPT
420 013D 5F      LD      B, #WORDPT
421 013E 9A01    LD      [B+], #WORD1 ; (WORDPT) SET WORD POINTER
422 0140 9D05    LD      A, XINC
423 0142 65      SWAP  A
424 0143 B0      RRC   A
425 0144 B0      RRC   A
426 0145 9503    AND   A, #03      ;X7, X6
427 0147 A6      X     A, [B]    ; (WORD1)
428      ;
429 0148 9D06    LD      A, YINC
430 014A 65      SWAP  A
431 014B 950C    AND   A, #0C      ;Y7, Y6
432 014D 87      OR    A, [B]    ; (WORD1)
433 014E 9740    OR    A, #040     ;SET BIT 6
434 0150 B0F87  OR    A, BUTSTAT  ;GET BUTTON STATUS
435 0153 A2      X     A, [B+]   ; (WORD1)
436      ;
437 0154 9D05    LD      A, XINC
438 0156 953F    AND   A, #03F    ;X0-X5
439 0158 A2      X     A, [B+]   ; (WORD2)
440      ;
441 0159 9D06    LD      A, YINC
442 015B 953F    AND   A, #03F    ;Y0-Y5
443 015D A2      X     A, [B+]   ; (WORD3)
444 015E 68      RBIT  RPT, [B]   ; (CHANGE) RESET CHANGE OF STATUS
445 015F AA      LD      A, [B+]  ; INC B
446 0160 9A00    LD      [B+], #0 ; (XINC)
447 0162 9A00    LD      [B+], #0 ; (YINC)
448      ;
449 0164 9A03    LD      [B+], #03 ; (NUMWORD) SEND 3 BYTES
450 0166 9E01    LD      [B], #01 ; (SENDST) SET TO START BIT STATE
451      ;
452 0168 BDEF78  SBIT  GIE, PSW   ;ENABLE INTERRUPT
453 016B 8E      RET
454      ;
455      ;*****
456      ;      SUBROUTINE SET UP REPORT 'SRPT' FOR MOUSE SYSTEMS
457      ;      -----
458      ;      CHANGE OF STATUS DETECTED
459      ;      SET UP THE FIRST 3 WORDS FOR REPORTING

```

TL/DD/10799-15

```

460      ;      IF IN IOLE STATE
461      ;*****
462      ;
463      SRPTMS:
464      016C 5B      LD      B, #CHANGE
465      016D 70      IFBIT  RPT, [B]
466      016E 01      JP      SRMS1
467      016F 8E      RET      ;EXIT IF NO CHANGE
468      ;
469      SRMS1:
470      0170 B0EF68  RBIT  GIE, PSW      ;DISABLE INTERRUPT
471      0173 5F      LD      B, #WORDPT
472      0174 9A01    LD      [B+], #WORD1 ; (WORDPT)SET WORD POINTER
473      0176 900F    LD      A, BUTSTAT
474      0178 A2      X      A, [B+]      ; (WORD1)
475      ;
476      0179 9005    LD      A, XINC
477      017B A2      X      A, [B+]      ; (WORD2)
478      ;
479      017C A1      SC
480      017D 64      CLR   A
481      017E B00681  SUBC  A, YINC      ;FOR MOUSE SYSTEM NEG Y
482      0181 A2      X      A, [B+]      ; (WORD3)
483      ;
484      0182 68      RBIT  RPT, [B]      ; (CHANGE)RESET CHANGE OF STATUS
485      0183 79      SBIT  SYRPT, [B]   ; (CHANGE)
486      0184 AA      LD      A, [B+]      ; INC B
487      0185 9A00    LD      [B+], #0     ; (XINC)
488      0187 9A00    LD      [B+], #0     ; (YINC)
489      ;
490      0189 9A03    LD      [B+], #03    ; (NUMWORD)SEND 3 BYTES
491      018B 9E01    LD      [B], #01     ; (SENDST)SET TO START BIT STATE
492      ;
493      018D B0EF78  RBIT  GIE, PSW      ;ENABLE INTERRUPT
494      0190 8E      RET
495      ;
496      ;
497      ;*****
498      ;      SUBROUTINE TO SEND DATA 'SDATA'
499      ;      CHECK THE BIT TO SEND AND DRIVE THE OUTPUT TO THE
500      ;      DESIRED VALUE
501      ;
502      ;      SENDST      STATE
503      ;      0          IDLE
504      ;      1          START BIT
505      ;      2-8        DATA
506      ;      2-9        DATA (FOR MOUSE SYSTEMS)
507      ;      9-10       STOP BIT
508      ;      10-11     STOP BIT (FOR MOUSE SYSTEMS)
509      ;      11        NEXT WORD
510      ;      12        NEXT WORD (FOR MOUSE SYSTEMS)

```

TL/DD/10799-16

```

511 ;
512 ;*****
513 ;
514 0191 55 SDATA: LD B, #TSTATUS
515 0192 72 IFBIT TBAUB, [B] ;(TSTATUS)CHECK IF BAUD RATE TIMER ENDS
516 0193 01 JP SDATA1
517 0194 8E RET
518 ;
519 SDATA1:
520 0195 6A RBIT TBAOB, [B] ;(TSTATUS)
521 0196 AA LD A, [B+] ;INC B TO (MTYPE)
522 0197 9008 LD A, SENDST
523 0199 97F0 OR A, #0F0
524 019B A5 JID
525 ;
526 019C 8E IDLE: RET ;EXIT IF IDLE
527 ;
528 019D 77 STAT9: IFBIT USOFT, [B] ;(MTYPE)
529 019E 16 JP STOPB
530 DATAB:
531 019F 9000 LD A, WORDPT
532 01A1 9CFE X A, B ;B POINTS TO THE WORD
533 ;
534 01A3 A0 RC
535 01A4 AE LD A, [B]
536 01A5 B0 RRC A ;XMIT LEAST SIG BIT
537 01A6 A6 X A, [B]
538 01A7 DED4 LD B, #PORTCD
539 01A9 88 IFC
540 01AA 7A SBIT XMT, [B]
541 01AB 89 IFNC
542 01AC 6A RBIT XMT, [B]
543 ;
544 01AD 9008 NEXT: LD A, SENDST
545 01AF 8A INC A
546 01B0 9C08 X A, SENDST
547 01B2 8E RET ;EXIT
548 ;
549 01B3 77 STAT11: IFBIT USOFT, [B] ;(MTYPE)
550 01B4 04 JP NXWORD
551 ;
552 01B5 BDD47A STOPB: SBIT XMT, PORTCD
553 01B8 F4 JP NEXT
554 ;
555 01B9 9D00 NXWORD: LD A, WORDPT
556 01BB 8A INC A
557 01BC BD0783 IFGT A, NUMWORD ;NUMBER OF WORDS TO SEND
558 01BF 09 JP ENDRPT ;END OF REPORT
559 01C0 9C00 X A, WORDPT
560 01C2 BC0801 LD SENDST, #01 ;SEND START BIT
561 ;

```

TL/DD/10799-17

```

562 01C5 BDD46A   STARTB: RBIT   XMT,PORTGD   ;SEND START BIT
563 01C8 E4       JP       NEXT
564               ;
565 01C9 BD0471   ENDRPT: IFBIT  SYRPT,CHANGE
566 01CC 04       JP       SY2RPT
567               ;
568 01CD BC0800   LD       SENDST,10
569 01D0 8E       RET
570               ;
571               ;*****
572               ;   SET UP LAST 2 WORDS IN MOUSE SYSTEM FORMAT
573               ;*****
574               ;
575               ;SY2RPT:
576 01D1 BDEF68   RBIT   GIE,PSW   ;DISABLE INTERRUPT
577               ;
578 01D4 5F       LD       B,#WORDPT
579 01D5 9A01   LD       [B+],#WORD1   ;(WORDPT)SET WORD POINTER
580 01D7 9005   LD       A,XINC
581 01D9 A2       X       A,[B+]   ;(WORD1)
582               ;
583 01DA A1       SC
584 01DB 64       CLR      A
585 01DC BD0681   SUBC   A,YINC   ;FOR MOUSE SYSTEM NEG Y
586 01DF A2       X       A,[B+]   ;(WORD2)
587               ;
588 01E0 AA       LD       A,[B+]   ;INC B
589 01E1 69       RBIT   SYRPT,[B]   ;(CHANGE)RESET CHANGE OF STATUS
590 01E2 AA       LD       A,[B+]   ;INC B
591 01E3 9A00   LD       [B+],#0   ;XINC
592 01E5 9A00   LD       [B+],#0   ;YINC
593               ;
594 01E7 9A02   LD       [B+],#02   ;(NUMWORD)SEND 2 BYTES
595 01E9 9E01   LD       [B],#01   ;(SENDST)SET TO START BIT STATE
596               ;
597 01EB BDEF78   SBIT   GIE,PSW   ;ENABLE INTERRUPT
598 01EE 21C5   JP       STARTB
599               ;
600 01F0         .:=01F0
601               ;
602 01F0 9C       .ADDR  IDLE       ;0
603 01F1 C5       .ADDR  STARTB      ;1
604 01F2 9F       .ADDR  DATAB       ;2
605 01F3 9F       .ADDR  DATAB       ;3
606 01F4 9F       .ADDR  DATAB       ;4
607 01F5 9F       .ADDR  DATAB       ;5
608 01F6 9F       .ADDR  DATAB       ;6
609 01F7 9F       .ADDR  DATAB       ;7
610 01F8 9F       .ADDR  DATAB       ;8
611 01F9 9D       .ADDR  STAT9      ;9
612 01FA B5       .ADDR  STOPB      ;10

```

TL/DD/10799-18


```

613 01FB B3          .ADDR  STAT11      ;11
614 01FC B9          .ADDR  NXWORD      ;12
615 01FD 9C          .ADDR  IDLE       ;13
616 01FE 9C          .ADDR  IDLE       ;14
617 01FF 9C          .ADDR  IDLE       ;15
618                  ;
619                  ;
620                  ;*****
621                  ;   SAMPLE BUTTON INPUT   FOR MICROSOFT
622                  ;   -----
623                  ;   INDICATE BUTTON STATUS
624                  ;*****
625                  ;
626                  ; BUTUS:
627 0200 9E00        LD      [B],#0      ;(BTEMP), (A=PORTLP, CARRY ROTATED)
628 0202 89          IFNC          ;MICROSOFT: 1=KEY DEPRESSED
629 0203 7D          SBIT      5,[B]      ;(BTEMP)
630                  ;
631 0204 B0          RRC      A
632 0205 B0          RRC      A
633 0206 89          IFNC          ;(BTEMP)
634 0207 7C          SBIT      4,[B]      ;(BTEMP)
635                  ;
636 0208 AA          LD      A,[B+]      ;(BTEMP)
637 0209 82          IFEQ      A,[B]      ;(BUTSTAT)
638 020A 8E          RET              ;NO CHANGE
639                  ;
640 020B A6          X        A,[B]      ;(BUTSTAT)
641 020C B00478      SBIT      RPT,CHANGE ;INDICATE TO SEND DATA
642 020F 8E          RET
643                  ;
644                  ;
645                  ;*****
646                  ;   SAMPLE BUTTON INPUT   FOR MOUSE SYSTEMS
647                  ;   -----
648                  ;   INDICATE BUTTON STATUS
649                  ;*****
650                  ;
651                  ; BUTMS:
652 0210 9E87        LD      [B],#087      ;(BTEMP)
653                  ;
654 0212 89          IFNC          ;MOUSE SYSTEM: 0=KEY DEPRESSED
655 0213 6A          RBIT      2,[B]      ;(BTEMP)
656                  ;
657 0214 B0          RRC      A
658 0215 89          IFNC          ;(BTEMP)
659 0216 69          RBIT      1,[B]      ;(BTEMP)
660                  ;
661 0217 B0          RRC      A
662 0218 89          IFNC          ;(BTEMP)
663 0219 68          RBIT      0,[B]      ;(BTEMP)

```

TL/DD/10799-19

```

664 ;
665 021A AA LD A,[B+] ;(BTEMP)
666 021B 82 IFEQ A,[B] ;(BUTSTAT)
667 021C 8E RET ;NO CHANGE
668 ;
669 021D A6 X A,[B] ;(BUTSTAT)
670 021E BD0478 SBIT RPT,CHANGE ;INDICATE TO SEND DATA
671 0221 8E RET
672 ;
673 ;*****
674 ;
675 03D0 .=03D0
676 03D0 28 .BYTE '(C) 1990 NATIONAL SEMICONDUCTOR AMOUSE VER 1.0'
03D1 43
03D2 29
03D3 20
03D4 31
03D5 39
03D6 39
03D7 30
03D8 20
03D9 4E
03DA 41
03DB 54
03DC 49
03DD 4F
03DE 4E
03DF 41
03E0 4C
03E1 20
03E2 53
03E3 45
03E4 4D
03E5 49
03E6 43
03E7 4F
03E8 4E
03E9 44
03EA 55
03EB 43
03EC 54
03ED 4F
03EE 52
03EF 20
03F0 41
03F1 4D
03F2 4F
03F3 55
03F4 53
03F5 45
03F6 20
03F7 56
03F8 45
03F9 52
03FA 20
03FB 31
03FC 2E
03FD 30
677 ;
678 .END

```

TL/DD/10799-20

TL/DD/10799-21

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88
 AMOUSE
 SYMBOL TABLE

ASAVE 00F4	B 00FE	BSAVE 00F5 *	BTEMP 000E
BUSY 0002 *	BUTMS 0210	BUTSTA 000F	BUTUS 0200
CARRY 0006 *	CHANGE 0004	CKO 0007 *	CNTRL 00EE
COMX 0092	COMY 0008	CSEL 0006 *	DATAB 019F
DECX 008F	DECY 0005	ENDRPT 01C9	ENI 0001 *
ENTI 0004 *	ESENS 00E0	GIE 0000	GTEMP 000C
HCARRY 0007 *	IDLE 019C	IEDG 0002 *	INCX 008B
INCY 00D1	INTR 0000 *	INTRET 0109	INTRP 00FF *
IPND 0003	LPUS 005C	LTIMER 0027 *	MLOOP 003D
MOVEMX 00B0 *	MOVEMY 00C0 *	MSEL 0003 *	MTYPE 000B
NEXT 01AD	NOISEX 008A	NOISEY 00D0	NUMOR 0007
NXWORD 01B9	PORTGC 00D5	PORTGD 00D4	PORTGP 00D6
PORTLC 00D1	PORTLD 00D0	PORTLP 00D2	PSSAVE 00F6 *
PSW 00EF	RPT 0000	RSVD 00F0 *	RISR2 012E
SO 0000 *	S1 0001 *	SDATA 0191	SDATA1 0195
SELECT 0067	SENDST 0008	SENSOR 0077	SI 0006 *
SK 0005 *	SO 0004 *	SP 00FD	SRMS1 0170
SRPTMS 016C	SRTUS 0136	SRUS1 013A	START 0000 *
STARTB 01C5	STAT11 01B3	STAT9 019D	STOPB 01B5
SW 0003	SYZPT 01D1	SYM 0071	SYRPT 0001
TAUHI 00ED *	TAULO 00EC *	TBAU 00F3 *	TBAUB 0002
TBAUR 0009 *	TEGC 0005 *	TEMP 00F1 *	TINTR 010C
TIO 0003 *	TMRHI 00EB *	TMRLO 00EA	TPND 0005
TRACKS 00DD	TRUN 0004	TSEL 0007 *	TSTATU 000A
USM 006B	USOPT 0007	WORD1 0001	WORD2 0002
WORD3 0003 *	WORDPT 0000	X 00FC	XINC 0005
XINTR 0113	XINTR1 011B	XMT 0002	YDIR 009A
YINC 0006			

TL/DD/10799-22

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88
 AMOUSE
 MACRO TABLE

NO WARNING LINES

NO ERROR LINES

556 ROM BYTES USED

SOURCE CHECKSUM = 987A
 OBJECT CHECKSUM = 0A39

INPUT FILE D:BMOUSE.MAC
 LISTING FILE D:BMOUSE.PRN
 OBJECT FILE D:BMOUSE.LM

TL/DD/10799-23

Using COP800 Devices to Control DC Stepper Motors

National Semiconductor
Application Note 714
Michelle Giles



INTRODUCTION

COP800 devices can be used to control DC stepper motors with limited effort. This application note describes the use of a COP820 to control the speed, direction and rotation angle of a stepper motor. In addition to the COP820, this application requires a quad high current peripheral driver (DS3658) to meet the high current needs of the stepper motor.

DC STEPPER MOTOR

A DC stepper motor translates current pulses into rotor movement. A typical motor contains four winding coils labeled red, yellow/white, red/white, and yellow. Applying current to these windings forces the motor to step. For normal operation, two windings are activated (pulsed) concurrently. The motor moves clockwise one step per change in windings activated with the following activation sequence: red and yellow, yellow and red/white, red/white and yellow/white, yellow/white and red, repeat. Half-steps may be generated by altering the sequence to: red and yellow, yellow, yellow and red/white, red/white, red/white and yellow/white, yellow/white, yellow/white and red, red, repeat. The motor runs in a counterclockwise direction if either sequence is applied in reverse order. The speed of rotation (number of steps/second) is controlled by the frequency of the pulses.

COP820 CONTROL OF STEPPER MOTOR

The COP820 controls the stepper motor by sending pulse sequences to the motor windings in response to control commands. Commands executed by the code in this application include: single step the motor in a clockwise or counterclockwise direction (i.e. rotate the rotor through a certain number of degrees), run the motor continuously at one of four speeds in a clockwise or counterclockwise direction, and stop the motor.

Note: Half-stepping is not implemented in this example.

During continuous mode operation, the 16-bit timer of the COP820 is used to control the speed of the stepper motor. The timer is set up with a value that causes an underflow once every x seconds or at a frequency of $1/x$. Each underflow of the timer interrupts the microcontroller. In response to the timer interrupt, the microcontroller generates a new pulse and causes a single step of the motor. Thus the motor steps at the frequency of the timer underflows. This application sets up the timer to generate interrupts at four different frequencies. These frequencies produce the following motor speeds: 25 steps/second, 100 steps/second, 200 steps/second, and 400 steps/second.

The determination of which windings to activate and deactivate to step the motor is performed by a single subroutine in this example. A block of memory is allocated to store a step pointer and the four possible stepper drive values are shown in Table I (9,C,6,3). Consecutive memory locations are used to store the stepper drive values so that applying the value from location X and then location $X+1$ (or $X-1$) causes the motor to step once. The motor drive subroutine increments or decrements the pointer to the current drive value based on the selection of a clockwise or counterclockwise direction. Writing the value from the newly selected location to the motor causes a single step of the motor in the appropriate direction.

During single step operation, the microcontroller steps the motor the exact number of times requested in the control command. Each step corresponds to 1.8 degrees of rotor movement. Therefore, a request to perform 200 steps will rotate the rotor through one complete revolution (360 degrees) at a fixed speed.

A block diagram of the application is shown in *Figure 1*. A flowchart of the code used to control the motor is given in *Figure 2*. The complete code is given at the end.

TABLE I. Stepper Motor Drive Sequence

Step	Yellow	Red/White	Yellow/White	Red	Hex Value
0	ON	OFF	OFF	ON	9
1	ON	ON	OFF	OFF	C
2	OFF	ON	ON	OFF	6
3	OFF	OFF	ON	ON	3
4	ON	OFF	OFF	ON	9

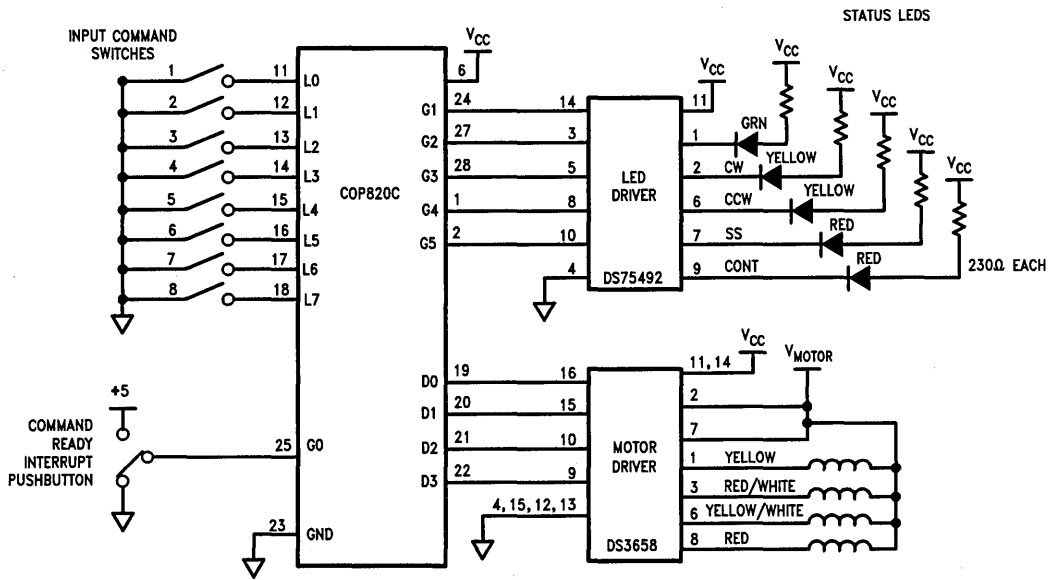


FIGURE 1. Schematic Diagram

TL/DD/11044-1

Program Code Flow Chart

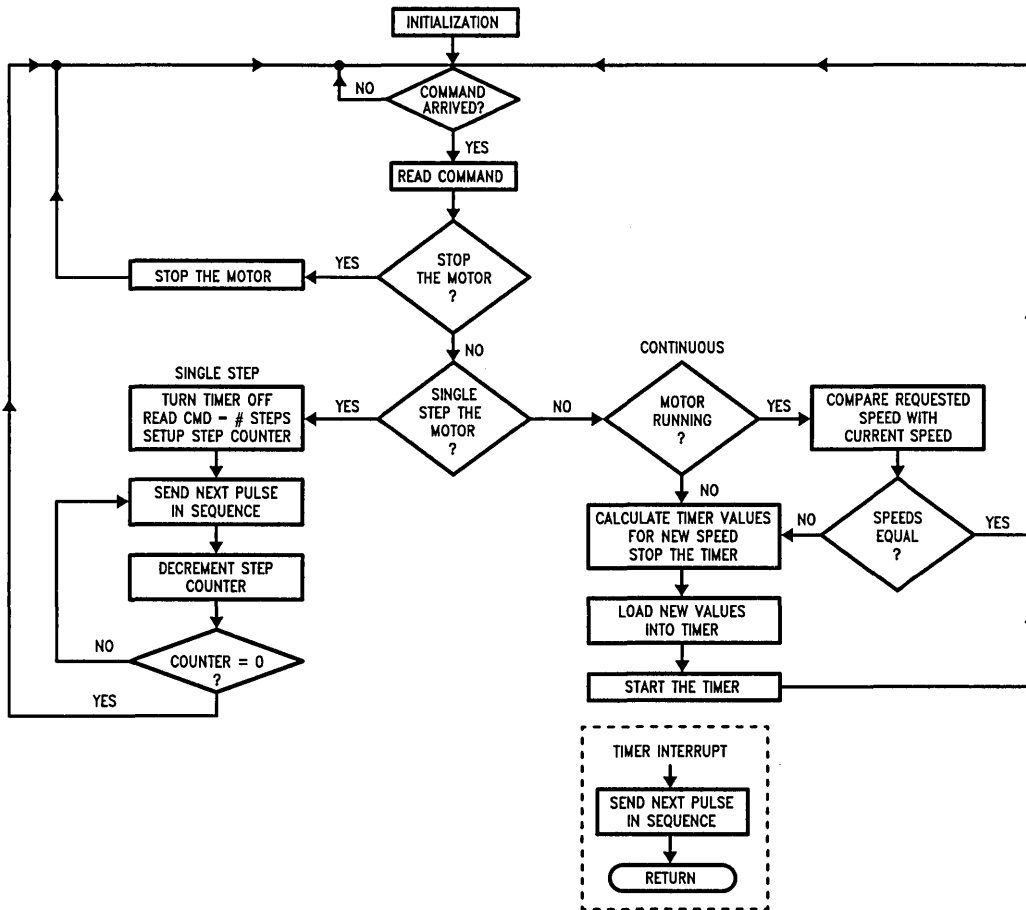


FIGURE 2. Program Flowchart

TL/DD/11044-2

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

```

1      ; STEPPER MOTOR CONTROL PROGRAM
2      ; MAY 1990
3
4      ; This program controls the speed, direction, and degree of rotation of
5      ; a DC stepper motor.
6
7
8      ;
9      ; Memory Map
10     ;
11     ; RAM CONTENTS
12     ; 00 (MS0) step motor drive value 09H      (two windings active per pulse)
13     ; 01 (MS1) step motor drive value 0CH
14     ; 02 (MS2) step motor drive value 06H
15     ; 03 (MS3) step motor drive value 03H
16     ; 04 (CMD) control command
17     ; bit7 - bit4 = motor speed or upper nibble of # single steps
18     ; bit 3      = unused
19     ; bit 2      = (MODE) single step or continuous mode select (1 = ss)
20     ; bit 1      = (DIR) cw or ccw direction select (1 = cw)
21     ; bit 0      = (GO) motor go or motor stop select (1 = stop)
22     ; 05 (STEPS) lower byte of number of single steps
23     ; 07 (FLGREG) flag register
24     ; bit 0      = (INT) ready to read in cmd (ext int occurred)
25     ; bit1 - bit7 = unused
26     ; 14 (TVAL0) value to load in lower byte of timer for speed X
27     ; 15 (TVAL1) value to load in upper byte of timer for speed X
28     ; D2 (PORTLP) port L input pins used for incoming commands
29     ; D4 (PORTGD) port G data pins used to drive status LEDs
30     ; DC (PORTD) port D data pins used to output pulses to the stepper motor
31     ; F0 (CREG0) step counter register zero
32     ; F1 (CREG1) step counter register one
33     ; F2 (STPPTR) pointer to current step motor drive value (RAM 00 - 03)
34
35     ; REGISTER AND CONSTANT DEFINITIONS
36
37     ; COMMAND BITS
38     ; 0000 GO = 0 ; GO COMMAND BIT
39     ; 0001 DIR = 1 ; DIRECTION COMMAND BIT
40     ; 0002 MODE = 2 ; MODE COMMAND BIT
41     ; 1 = STOP 0 = GO
42     ; 1 = CW 0 = CCW
43     ; 1 = SINGLE STEP 0 = CONTINUOUS
44
45     ; PORTG BITS
46     ; 0000 INT = 0 ; FLAG BIT (SET IF EXTINT OCCURS)
47     ; 0001 READY = 1 ; READY LED
48     ; 0002 CW = 2 ; CLOCKWISE LED
49     ; 0003 CCW = 3 ; COUNTER CLOCKWISE LED
50     ; 0004 SS = 4 ; SINGLE STEP LED
51     ; 0005 NS = 5 ; CONTINUOUS (NON-STOP) LED
52
53     ; REGISTERS
54     ; 0004 CMD = 04 ; INPUT COMMAND STORAGE REGISTER

```

TL/DD/11044-3

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

```

52      0005      STEPS = 05      ;INPUT #STEPS/SPEED REGISTER
53      00F0      CREG0  = 0F0    ;COUNTER REGISTER
54      00F1      CREG1  = 0F1    ;COUNTER REGISTER
55      0007      FLGREG = 07     ;FLAG REGISTER (FLAG BITS)
56      00F2      STPTR  = 0F2    ;CURRENT MOTOR STEP POINTER
57      0014      TVAL0  = 014    ;MOTOR SPEED LOAD VALUES
58      0015      TVAL1  = 015
59      0000      MS0    = 00
60      0001      MS1    = 01      ;STEPPER MOTOR DRIVE VALUES
61      0002      MS2    = 02
62      0003      MS3    = 03

```

;ASSIGNMENTS FOR COP820

```

64
65
66      00D5      PORTGC = 0D5
67      00D4      PORTGD = 0D4
68      00D6      PORTGP = 0D6
69      00D1      PORTLC = 0D1
70      00D0      PORTLD = 0D0
71      00D2      PORTLP = 0D2
72      00DC      PORTD  = 0DC
73      00D7      PORTI  = 0D7
74      00E9      SIOR   = 0E9
75      00EA      TMRLO  = 0EA
76      00EB      TMRHI  = 0EB
77      00EC      TAULO  = 0EC
78      00ED      TAUHI  = 0ED
79      00EE      CNTRL  = 0EE
80      00EF      PSW    = 0EF

```

```

81
82
83      0000      GIE     = 0
84      0001      ENI     = 1
85      0002      BUSY    = 2
86      0003      IPND    = 3
87      0004      ENTI    = 4
88      0005      TPND    = 5
89
90      0002      IEDG    = 2
91      0003      MSEL    = 3
92      0004      TRUN    = 4
93      0005      TC3     = 5
94      0006      TC2     = 6
95      0007      TC1     = 7
96
97

```

.CHIP 820

```

98
99
100     ;INITIALIZATION OF REGISTERS
101 0000 DD2F      LD      SP,#02F
102 0002 BCEE80   LD      CNTRL,#080

```

;*****

TL/DD/11044-4

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

```

103 0005 BCEF03          LD      PSW,#003          ;GLOBAL INT ENABLE/EXTINT ENABLE
104 0008 BCD401          LD      PORTGD,#01
105 000B BCD53E          LD      PORTGC,#03E        ;CONFIG PORTG FOR OUTPUTS
106 000E BCDC09          LD      PORTD,#09         ;START MOTOR DRIVE VALUE
107 0011 BCD100          LD      PORTLC,#00        ;CONFIG PORTL FOR INPUTS
108 0014 BCD0FF          LD      PORTLD,#0FF       ;CONFIG PORTL FOR WEAK PULL-UPS
109 0017 5F              LD      B,#MS0           ;SETUP MOTOR DRIVE VALUES
110 0018 9A09            LD      [B*],#09
111 001A 9A0C            LD      [B*],#0C
112 001C 9A08            LD      [B*],#08
113 001E 9E03            LD      [B],#03
114 0020 D200            LD      STPPTR,#00        ;INIT STEP POINTER
115 0022 BC0700          LD      FLGREG,#00       ;INIT FLAG REGISTER
116
117
118                      ;READ, DECODE, AND EXECUTE COMMAND
119
120 0025 BDD479          TOP:    SBIT      READY,PORTGD          ;*****
121 0028 3081            JSR      WAIT                          ;TURN ON READY FOR NEXT CMD LED
122 002A BDD469          RBIT      READY,PORTGD          ;WAIT FOR CMD AND READ CMD
123 002D 9C04            X                ;TURN OFF READY FOR NEXT CMD LED
124 002F BD0470          IFBIT     GO,CMD                ;STORE IN CMD REGISTER
125 0032 08              JP        STOP                    ;IF STOP BIT SET
126 0033 BD0472          IFBIT     MODE,CMD             ;THEN STOP MOTOR
127 0036 3041            JSR      SSTEP                   ;ELSE CHEK MODE
128 0038 305F            JSR      CONT                    ;IF MODE SET THEN GO SINGLE STEP
129 003A EA              JP        TOP                     ;ELSE GO CONTINUOUS
130                      STOP:
131 003B 308E            JSR      TMRSET                  ;GO WAIT FOR NEXT COMMAND
132 003D BCD401          LD      PORTGD,#01             ;STOP THE MOTOR
133 0040 E4              JP        TOP                     ;STOP THE TIMER
134                      ;TURN OFF ALL LEDS
135                      ;GO WAIT FOR NEXT CMD
136
137                      ;SINGLE STEP THE MOTOR (SS)
138
139 0041 308E            SSTEP:  JSR      TMRSET                  ;*****
140 0043 BCD410          LD      PORTGD,#010           ;STOP TIMER
141 0046 3081            JSR      WAIT                          ;TURN ON SS LED (RST ALL OTHER LEDS)
142 0048 8A              INC      A                          ;WAIT FOR CMD BYTE 2 (# STEPS)
143 0049 9CF0            X                ;ADD 1 TO CORRECT FOR LOOP
144 004B 9D04            LD      A,CREG0                ;STORE #STEPS IN LOBYTE COUNT REG
145 004D 65              SWAP     A                          ;LOAD HIBYTE # STEPS
146 004E 950F            AND      A,#0F                  ;MOVE TO LOWER NIBBLE
147 0050 8A              INC      A                          ;GET RID OF UPPER BITS
148 0051 9CF1            X                ;ADD 1 TO CORRECT FOR LOOP
149 0053 C0              TP2:    DRSZ     CREG0            ;MOVE TO HIBYTE OF COUNT REG
150 0054 05              JP        DO                       ;DECR LOBYTE AND IF NOT ZERO
151 0055 C1              MID:    DRSZ     CREG1            ;THEN GO DO A STEP
152 0056 01              JP        DO2                      ;ELSE DECR HIBYTE AND IF NOT ZERO
153 0057 8D              RETSK    DO2                       ;THEN GO DO A STEP AND RST LO COUNT
                                           ;ELSE END OF LOOP      RETURN

```

TL/DD/11044-5

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

```

154 0058 D0FF      DO2:  LD   CREGO,#0FF      ;RESET LOBYTE OF COUNTER
155 005A 3098      DO:   JSR   NXTVAL      ;STEP THE MOTOR
156 005C 3158      JSR   DELAY      ;SLOW THE STEPPING
157 006E F4        JP    TP2        ;GO TO TOP OF LOOP
158
159
160                ;RUN THE MOTOR CONTINUOUSLY (NS = NON-STOP = CONTINUOUSLY)
161
162                CONT:                ;*****
163 005F BDEE74      IFBIT  TRUN,CNTRL      ;IF MOTOR ALREADY RUNNING NS
164 0062 01         JP    CHKSPD      ;THEN CHECK THE CURRENT SPEED
165 0063 03         JP    SETGO      ;ELSE GO START THE MOTOR
166 0064 3148      CHKSPD: JSR   SPEED      ;COMPARE INPUT WITH ACTUAL SPD
167 0068 8E        RET                ;IF EQUAL RET ELSE RESTART MOTOR
168 0067 308E      SETGO: JSR   TMRSET      ;STOP THE TIMER
169 0069 BCD420     LD    PORTGD,#020      ;TURN ON CONTINUOUS LED
170 006C 3126      JSR   TIMVAL      ;CALCULATE TIMER (SPEED) VALUE
171 006E AE        LD    A,[B]        ;LOAD A WITH TVAL1
172 006F 9CEB      X     A,TMRHI      ;MOVE SPEED VAL INTO TIMER
173 0071 AB        LD    A,[B-]      ;LOAD A WITH TVAL1 POINT TO TVAL0
174 0072 9CED      X     A,TAUHI      ;MOVE SPEED VAL INTO AUTORELOAD REG
175 0074 AE        LD    A,[B]        ;LOAD A WITH TVAL0
176 0075 9CEA      X     A,TMRLO      ;MOVE SPEED VAL INTO TIMER
177 0077 AE        LD    A,[B]        ;LOAD A WITH TVAL0
178 0078 9CEC      X     A,TAULO      ;LOAD A WITH TVAL0
179 007A BDEF7C     SBIT  ENTI,PSW      ;ENABLE TIMER INTERRUPT
180 007D BDEE7C     SBIT  TRUN,CNTRL    ;START THE TIMER
181 0080 8E        RET                ;RET TO MAIN AND WAIT FOR TMRINT
182
183                ;SUPPORT ROUTINES *****
184
185                WAIT:                ;*****
186                ;WAIT FOR AN EXTERNAL INTERRUPT TO SIGNAL AN INCOMING COMMAND
187                ;READ THE INCOMING COMMAND FROM PORT L
188 0081 BD0770     IFBIT  INT,FLGREG      ;IF EXTERNAL INTERRUPT OCCURED
189 0084 01         JP    OUT        ;THEN JUMP OUT OF LOOP
190 0085 FB        JP    WAIT        ;ELSE CONTINUE TO WAIT
191 0086 BD0768     OUT:   RBIT  INT,FLGREG      ;RESET EXTERNAL INTERRUPT FLAG
192 0088 9DD2      LD    A,PORTLP      ;READ INCOMING COMMAND
193 008B 98FF      XOR   A,#0FF        ;COMPLEMENT INCOMING COMMAND
194 008D 8E        RET                ;RETURN COMMAND IN ACC
195
196                TMRSET:                ;*****
197                ;RESET THE TIMER
198                RBIT  TRUN,CNTRL      ;STOP THE TIMER
199 008E BDEE6C     RBIT  TPND,PSW      ;RESET THE TIMER PENDING BIT
200 0091 BDEF6D     RBIT  ENTI,PSW      ;DISABLE TIMER INTERRUPT
201 0094 BDEF6C     RET
202 0097 8E
203
204

```

TL/DD/11044-6

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

```

205          NXTVAL: ;*****
206          ;SEND THE NEXT DRIVE VALUE TO STEP THE MOTOR ONE STEP IN THE
207          ;APPROPRIATE DIRECTION (CW OR CCW)
208 0098 9DF2      LD      A,STPPTR      ;LOAD STEP VALUE POINTER
209 009A DED4      LD      B,#PORTGD    ;POINT TO PORT G
210 009C BD0471    IFBIT   DIR,CMD      ;IF CLOCKWISE
211 009F 11        JP      IPTR        ;THEN GO INCREMENT POINTER
212 00A0 6A        DPTR:  RBIT   CW,[B]   ;ELSE RST CW LED
213 00A1 7B        SBIT   CCW,[B]      ;TURN ON CCW LED
214 00A2 8B        DEC     A            ;AND DECREMENT POINTER
215 00A3 92FF      IFEQ   A,#OFF       ;IF OFF BOTTOM OF STEPS
216 00A5 9803      LD      A,#03        ;THEN LOOP TO TOP OF STEPS
217 00A7 9CF2      WRVAL:  X      A,STPPTR ;A -> STPPTR (SAVE NEW STPPTR)
218 00A9 9DF2      LD      A,STPPTR    ;[STPPTR] -> PORTD (LOOKUP VAL)
219 00AB 9CFE      X      A,B
220 00AD AE        LD      A,[B]
221 00AE 9CDC      X      A,PORTD      ;WRITE STEP VALUE TO MOTOR
222 00B0 8E        RET
223 00B1 6B        IPTR:  RBIT   CCW,[B]  ;TURN OFF CCW LED
224 00B2 7A        SBIT   CW,[B]      ;TURN ON CW LED
225 00B3 8A        INC     A            ;INCREMENT THE STEP POINTER
226 00B4 9204      IFEQ   A,#04        ;IF OFF TOP OF STEPS
227 00B6 64        CLR     A            ;THEN LOOP TO BOTTOM OF STEPS
228 00B7 EF        JP      WRVAL       ;GO WRITE VALUE TO MOTOR
229
230
231          ; INTERRUPT HANDLERS
232          . = OFF ;*****
233          ;BRANCH TO THE APPROPRIATE INTERRUPT HANDLER
234 00FF BDEF75    IFBIT   TPND,PSW      ;TIMER UNDERFLOW
235 0102 08        JP      TMRINT
236 0103 BDEF73    IFBIT   IPND,PSW      ;EXTERNAL INTERRUPT
237 0106 16        JP      EXTINT
238 0107 BDEF78    SBIT   GIE,PSW      ;SOFTWARE TRAP
239 010A 8D        RETSK
240
241          TMRINT: ;*****
242          ;RESET THE TIMER INTERRUPT PENDING BIT AND STEP THE MOTOR
243 010B 9CF9      X      A,0F9        ;CONTEXT SAVE ROUTINE
244 010D 9DFE      LD      A,B
245 010F 8CFA      X      A,0FA
246 0111 BDEF6D    RBIT   TPND,PSW      ;RESET PENDING BIT
247 0114 3098      JSR   NXTVAL        ;STEP THE MOTOR
248 0116 9DFA      LD      A,0FA        ;CONTEXT RESTORE ROUTINE
249 0118 9CFE      X      A,B
250 011A 9DF9      LD      A,0F9
251 011C 8F        RETI
252
253          EXTINT: ;*****
254 011D BD0778    SBIT   INT,FLGREG    ;SET INTERRUPT OCCURED FLAG
255 0120 3158      JSR   DELAY         ;WAIT
  
```

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

AN-714

```

256 0122 BDEF6B          RBIT  IPND,PSM          ;RESET PENDING BIT
257 0125 8F             RETI
258
259                     ;SUPPORT ROUTINES CONTINUED
260
261 TIMVAL:               ;*****
262                     ;During continuous operation, the motor is stepped once every
263                     ;timer underflow. Therefore, a timer value is calculated that will
264                     ;produce timer underflows every X microseconds causing the motor
265                     ;to step Xsteps/second.
266                     ;For example: To step 100 times per second.
267                     ;   microseconds/step = 1000000uS/sec x 1sec/100steps = 10000
268                     ;   10000uS/step = 02718Hex uS/step
269                     ;   1uS = one count down of the timer
270                     ; Therefore, load the timer with 02718H for 100 steps/sec.
271
272 0126 DE14             LD      B,#TVAL0          ;POINT TO STORAGE FOR TIMVAL
273 0128 BD0474          IFBIT  4,CMD          ;IF LOWEST SPEED BIT SET
274 012B 17              JP      SLOWER          ;THEN USE SLOWEST SPEED
275 012C BD0475          IFBIT  5,CMD          ;IF SECOND LOWEST SPD BIT SET
276 012F 0E              JP      SLOW          ;THEN USE SLOW SPEED
277 0130 BD0476          IFBIT  8,CMD          ;IF SECOND HIGHEST SPD BIT SET
278 0133 05              JP      FAST           ;THEN USE FAST SPEED
279 0134 9A02           FASTER: LD      [B+],#02      ;ELSE USE FASTEST SPEED
280 0136 9E08            LD      [B],#08          ;400steps/sec = 2rev/sec
281 0138 8E              RET
282 0139 9A88           FAST:  LD      [B+],#088      ;200steps/sec = 1rev/sec
283 013B 9E13            LD      [B],#013
284 013D 8E              RET
285 013E 9A18           SLOW:  LD      [B+],#018      ;100steps/sec = .5rev/sec
286 0140 9E27            LD      [B],#027
287 0142 8E              RET
288 0143 9A54           SLOWER: LD     [B+],#054      ;25steps/sec = .125rev/sec
289 0145 9E9C            LD      [B],#09C
290 0147 8E              RET
291
292
293 SPEED:                ;*****
294                     ;COMPARE CURRENT MOTOR SPEED WITH DESIRED MOTOR SPEED
295 0148 3126            JSR      TIMVAL          ;CALCULATED DESIRED SPEED VAL
296 014A 9D14            LD      A,TVAL0
297 014C BDEC82          IFEQ   A,TAULO          ;IF DESIRED LBYTE EQUALS CURRENT LBYTE
298 014F 01              JP      TSTHI          ;THEN GO TEST HI-BYTE
299 0150 8D              RETSK          ;ELSE NOT EQUAL RETURN AND SKIP
300 0151 9D15           TSTHI: LD      A,TVAL1
301 0153 BDED82          IFEQ   A,TAUHI          ;IF HI-BYTE EQUALS CURRENT HI-BYTE
302 0156 8E              RET           ;THEN DESIRED = CURRENT RETURN
303 0157 8D              RETSK          ;ELSE DESIRED != CURRENT RET & SKIP
304
305 DELAY:                ;*****
306                     ;INSERT A DELAY

```

TL/DD/11044-8

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV: D1, 12 OCT 88

```

307 0158 D301          LD      0F3,#01          ;FOR SINGLE STEP & EXTINT DEBOUNCE
308 015A D4FF          DLY1:  LD      0F4,#0FF        ;APPROX .256mS X 8
309 015C C4           DLY2:  DRSZ   0F4
310 015D FE           JP      DLY2
311 015E C3           DRSZ   0F3
312 015F FA           JP      DLY1
313 0160 8E           RET
314
315                   .END
  
```

TL/DD/11044-9

```

B      00FE          BUSY  0002 *      CCW   0003          CHKSPD 0064
CMD    0004          CNTRL 00EE          CONT  005F          CREGO  00F0
CREG1  00F1          CW     0002          DELAY 0158          DIR    0001
DLY1   015A          DLY2  016C          DO     005A          DO2    0058
DPTR   00A0 *       ENI    0001 *       ENTI  0004          EXTINT 011D
FAST   0139          FASTER 0134 *      FLGREG 0007          GIE    0000
GO     0000          IEDG  0002 *       INT   0000          IPND   0003
IPTR   00B1          MID   0055 *       MODE  0002          MS0    0000
MS1    0001 *       MS2   0002 *       MS3   0003 *       MSEL  0003 *
NS     0005 *       NXTVAL 0098          OUT   0086          PORTD  00DC
PORTGC 00D5          PORTGD 00D4          PORTGP 00D6 *      PORTI  00D7 *
PORTLC 00D1          PORTLD 00D0          PORTLP 00D2          PSW    00EF
READY  0001          SETGO 0067          SIOR  00E9 *       SLOW   013E
SLOWER 0143          SP     00FD          SPEED 0148          SS     0004 *
SSTEP  0041          STEPS 0005 *       STOP  003B          STPTR  00F2
TAUHI  00ED          TAULO  00EC          TC1   0007 *       TC2    0006 *
TC3    0005 *       TIMVAL 0126          TMRHI 00EB          TMRINT 010B
TMRLO  00EA          TMRSET 008E          TOP   0025          TP2    0053
TPND   0006          TRUN  0004          TSTHI 0151          TVALO  0014
TVAL1  0015          WAIT  0081          WRVAL 00A7          X      00FC
  
```

TL/DD/11044-10

MACRO TABLE

NO WARNING LINES

NO ERROR LINES

282 ROM BYTES USED

SOURCE CHECKSUM = 80C0
 OBJECT CHECKSUM = 0520

INPUT FILE C:MOTOR.MAC
 LISTING FILE C:MOTOR.PRN
 OBJECT FILE C:MOTOR.LM

TL/DD/11044-11

MF2 Compatible Keyboard with COP8 Microcontrollers

National Semiconductor
Application Note 734
Volker Soffel



ABSTRACT

This application note describes the implementation of an IBM MF2 compatible keyboard with National Semiconductor's COP888CL or COP943C/COP880CL microcontrollers. Two different solutions have been developed. One solution, suitable for laptop/notebook keyboards is based on the COP888CL with special power saving techniques. The other for most price competitive standard desktop keyboards is based on the COP943C/COP880C microcontrollers. The same principles can be applied to all types of keyboards or data input devices.

FEATURES

- Single chip solution
- Low cost R/C or ceramic oscillator optional
- LED direct drive capability
- I/Os with software programmable on chip pull-ups
- Current saving M2CMOS technology
- Multi-input wakeup and HALT mode for further power consumption reduction (COP888CL only)
- Software key rollover
- Schmitt triggers on keyboard data and clock lines

INTRODUCTION

The expression MF2 keyboard stands for multi-functional keyboard version 2. This type of keyboard was first developed and defined by IBM for use with all types of PC (XT,

AT, PS/2). In the meantime it has become an industry standard and today nearly all PCs have an MF2 compatible keyboard. As the name suggests, this keyboard features all operation modes which are necessary to stay compatible with the older XT and AT type keyboards. In the following chapters the features and functions of an MF2 keyboard as well as their implementation with a COP8 microcontroller are described.

MF2 KEYBOARD KEY-LAYOUT

Figure 1 shows the key layout of the U.S. version of an MF2 keyboard. Its outer appearance is characterized by 101 keys (102 for some countries), a separate cursor and numeric key pad, and 12 function keys in the upper row. The keyboard sends a "make" code if a key is depressed and a "break" code if the key is released. These make and break codes are independent of any country-specific keyboard layouts, which means they are independent of the symbols printed on the keys. These codes are solely determined by the physical position of a key on the keyboard. The physical position of a key on an MF2 keyboard is defined by its assigned key number, which is shown in Figure 1.

HARDWARE

Laptop/Notebook Keyboard With COP888CL

Figure 2 shows the schematics of an MF2 keyboard with a COP888CL microcontroller. The G, C and L ports of the COP888CL are software programmable I/Os and can be programmed either as TRI-STATE® inputs, inputs with weak pull-up, push-pull output low, or push-pull output high.

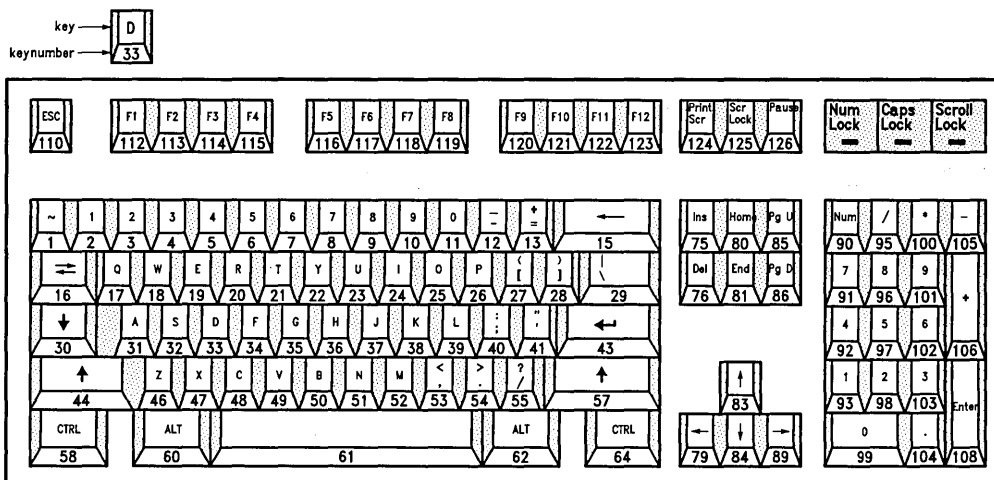
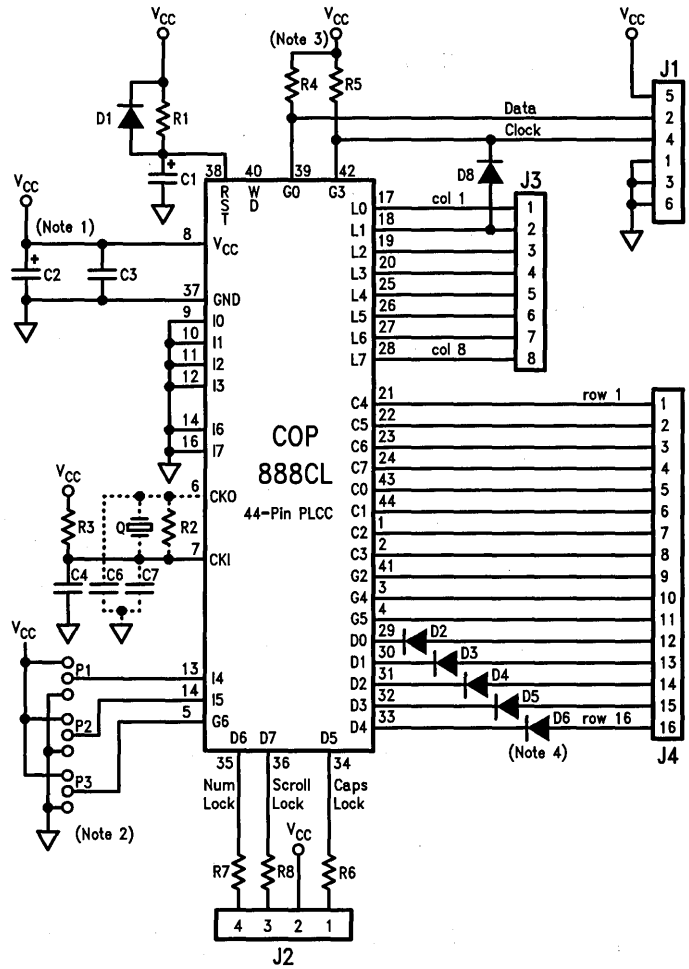


FIGURE 1. MF2 Keyboard U.S. Layout

TL/DD/11091-10



Note 1: C2 (47 μ F level off capacitor) can be removed when the power supply ripple < $\pm 10\%$, 0.5 V/ms.

Note 2: Jumper P1: Mode select: 0 = XT-mode, 1 = AT-mode. Jumper P2, P3: not used.

Note 3: Care must be taken if there are pullups in the computer system that clock/data line current < 3 mA.

Note 4: Diodes D2-D6 should be removed if keyboard has hardware keyrollover (diodes in matrix).

FIGURE 2. MF-2 Keyboard Schematics with a 44-Pin COP888CL

TL/DD/11091-11

The keyboard is organized as an 8 input by 16 output matrix. The COP888CL's L port is configured as a weak pull-up input port, thus allowing the use of the multi-input wakeup feature. Most of the time the chip is in the current saving HALT mode ($I_{dd} \leq 10 \mu A$). Any keystroke or a data transmission from the computer will create a high to low transition on one of the L lines, which wakes up the μC from HALT mode. After returning from the HALT mode, the keyboard is scanned in order to detect which key is pressed and the appropriate key code is sent to the computer. This event-driven keyboard scanning results in lowest possible current consumption as HALT mode is even entered between successive single keystrokes. The diodes in the D-lines of the key matrix prevent a high current from being drawn. When two keys in the same column are pressed, two outputs could be potentially connected together: one of the D output lines, which is high and the polled line, which is pulled low. In this case, excessive current would be drawn without the protection diodes. These diodes can be omitted if the keyboard already has decoupling diodes in its matrix (hardware key rollover). All other matrix lines source current in the μA range and there is no need for current limiting diodes.

The G0 and G3 pins are used for the keyboard data and clock lines. The pull-ups on these lines ensure a defined logic "1" level. The keyboard interface on the computer side uses open collector drivers and the G0, G3 pins of the COP888CL are configured as TRI-STATE (Hi-Z) inputs when a "1" is written to the data or clock line. To output a logic "0" the μC pulls the data or clock line low (push-pull low output). A maximum current of 3 mA can be sunk into the data and clock pins. Schmitt triggers on the data and clock line inputs reduce the risk of errors in the data received by the keyboard.

The microcontroller provides the option of using a low cost R/C oscillator with frequency variation tight enough to fulfill the requirements for a keyboard, in addition to the option of using a crystal or a ceramic clock.

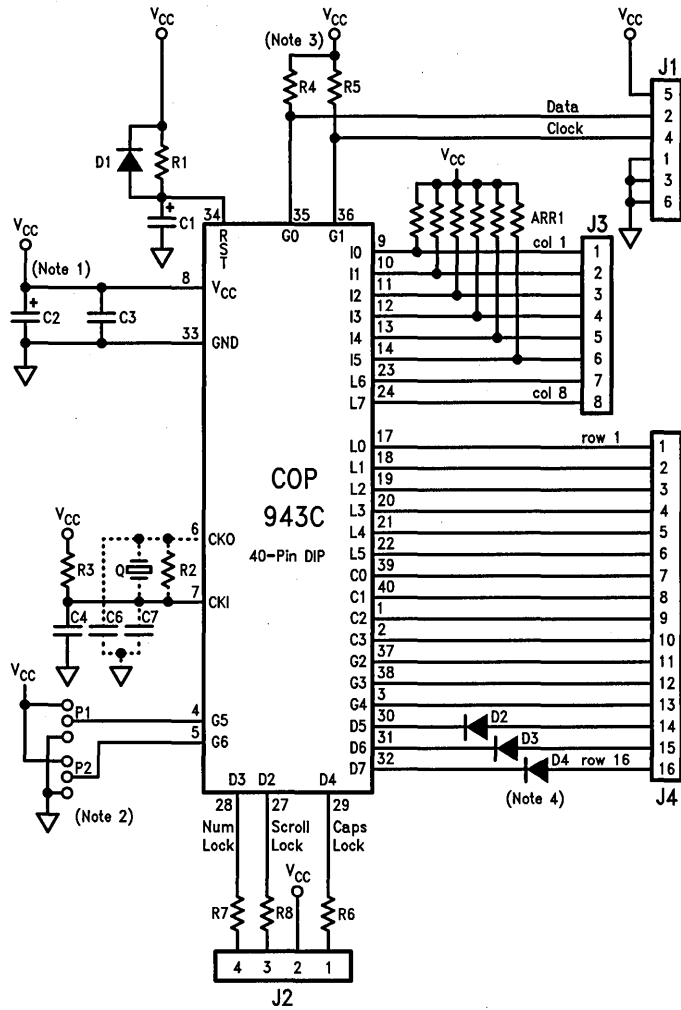
The XT or AT/PS-2 operation mode can be selected via a hardware switch. Additional inputs for customer specific settings are available.

The three LEDs of an MF2 keyboard are driven directly by three of the COP888CL's high sink D-lines (max. 15 mA for each pin), thus eliminating the need for additional LED drivers or transistors.

The keyboard logic generates a Power-On Reset (POR) signal when the power is first applied to the keyboard. After POR the keyboard performs the Basic Assurance Test (BAT). The BAT consists of a keyboard controller self-test. During the BAT, any activity on the data and clock lines is ignored. The 3 keyboard LEDs are turned on at the beginning and turned off at the end of the BAT. Upon satisfactory completion of the BAT, the keyboard sends the BAT completion code (hex AA) to the computer and keyboard scanning begins. Any code other than hex AA is interpreted by the computer as a BAT error.

Desktop Keyboard with COP943C or COP880C

Figure 3 shows the schematic for an MF2 keyboard with the COP943C/COP880C. The only difference compared to COP888CL solution is that the COP943C/COP880C microcontrollers do not have the multi-input wakeup feature, which allows an event driven keyboard scanning. The key matrix is therefore continuously scanned in a loop. With the COP943C/COP880C solution a part of the I port is used as the key matrix input. The I port is a TRI-STATE (Hi-Z) input port (requires external pull-ups).



Note 1: C2 (47 μ F level off capacitor) can be removed when the power supply ripple < $\pm 10\%$, 0.5 V/ms.

Note 2: Jumper P1: Mode select; 0 = XT-mode, 1 = AT-mode. Jumper P2: P3: not used.

Note 3: Care must be taken if there are pullups in the computer system that clock/data line current < 3 mA.

Note 4: Diodes D2-D4 should be removed if keyboard has hardware keyrollover (diodes in matrix).

FIGURE 3. MF-2 Keyboard Schematic with a 40-Pin COP943C/COP880C

TL/DD/11091-12

Key Matrix Organization

Figure 4 shows an example of what an MF2 keyswitch matrix could look like. Each key position in the matrix is marked with its key number.

For example: Key number "58" is located at the key matrix position number "2" and has the AT-set make code "14

Hex". Looking at Figure 1, one can see that key number "58" belongs to the left "CNTRL" key. Note that the "SHIFT", "CNTRL" and "ALT" keys are located in their own matrix lines, separate from all other keys. The reasons for that will be explained in the chapter "Software Key Roll-over".

		76:	61:	0C:	34:	03:	33:	0B:		52:		70:	71:	E0:	75:	11:																	
L0	1	0:	8:	10:	110:	18:	45:	20:	115:	28:	35:	116:	38:	36:	40:	117:	48:	50:	41:	58:	60:	99:	68:	104:	70:	83:	78:	60:					
L1	2	1:	9:	44:	11:	16:	19:	30:	21:	114:	29:	21:	31:	15:	39:	22:	41:	28:	49:	118:	51:	27:	59:	92:	61:	97:	69:	102:	71:	79:			
L2	3	14:	2:	58:	A:	0E:	05:	06:	2E:	01:	36:	55:	0A:	4E:	E0:	71:	E0:	70:	E0:	7D:	E0:	6C:											
L3	4	3:	B:		16:	1E:	26:	25:	09:	3D:	3E:	46:	45:	78:	07:	E0:	7A:	E0:	69:	*													
L4	5	4:	C:		15:	1D:	24:	2D:		3C:	43:	44:	4D:	6C:	75:	7D:	79:	7E:															
L5	6	5:	D:		14:	1C:	1B:	23:	2B:	5D:	5B:	42:	4B:	4C:	69:	72:	7A:	E0:	5A:														
L6	7	E0:	59:	1A:	22:	21:	2A:	5A:	3A:	41:	49:	5D:	77:	E0:	4A:	7C:	E1:																
L7	8	6:	64:	E:	57:	16:	46:	1E:	47:	26:	48:	2E:	49:	36:	43:	3E:	52:	46:	53:	4E:	54:	56:	42:	5E:	90:	88:	95:	6E:	100:	76:	126:	7E:	
L8	7:	F:		17:	1F:	27:	2F:	50:	37:	61:	3F:	51:	47:	4F:		4A:	E0:	72:	E0:	74:	7B:	E0:	6B:	E0:	11:								
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16																
		C4	C5	C6	C7	C0	C1	C2	C3	G2	G4	G5	D0	D1	D2	D3	D4																

FIGURE 4. Keyboard Matrix COP888CL AT Code Set

TL/DD/11091-13

Code Sets

The MF2 keyboard supports 3 different sets of make and break codes. Code set 1 is used for XT/PC and PS/2-30 compatible computers. Code set 2 is used for AT and all other PS/2 models compatible computers and code set 3 is used for workstations and terminal emulations on the PC. The country specific keyboard driver on the PC side converts the "key position" codes from the keyboard into the ASCII codes that correspond to the characters printed on the keycaps (as long as the right driver is installed on the PC). Appendix 1 gives a complete overview of the key numbers and their make and break codes for all 3 code sets. The symbols of the U.S. keyboard layout are only listed for reference and are different for other country layouts. The break code for code set 1 is equal to the make code with the most significant bit set. The make codes preceded with a "F0 Hex" code give the break codes of code sets 2 and 3.

KEYBOARD SOFTWARE

The software of the keyboard microcontroller can be subdivided into the following five main tasks:

- key detection
- software key rollover
- key decoding and encoding
- keycode transmission
- keyboard command set

Key Detection

Key detection is done by scanning the keyboard matrix in the following way. Sequentially each of the 16 matrix output lines are pulled low, while all the others are high. The 8

matrix input lines are read and the 8-bit input value is compared with the result of the previous scanning of the same matrix output line (a history of the previous scan is kept in the μ C's RAM). Thus the keyboard microcontroller's key detection routine detects any key change in that matrix output line (key pressed or released) since the previous scan. It is important to recognize released keys, as the MF2 keyboard not only sends a key's "make" code when the key is pressed, but also a key's "break" code when the key is released. Key debouncing is performed by software by making sure that the time between two scans is bigger than the key bounce time (typically 8 ms).

Software Key Rollover

Software key rollover means that no decoupling diodes are used in the key switch matrix. However, the keyboard action is still N key rollover in nature. That is, if N keys are depressed in some sequence and held down, the make code of these keys is transmitted in that sequence. However, if three keys from three corners of a rectangle in the key switching matrix are depressed, a "ghost" key (a key which is not really pressed) would be created (see *Figure 5*). To prevent this, a special algorithm, which checks for such special key combinations, has been implemented into the keyboard software. If a "ghost" key has been detected the keyboard outputs the "key detection error code" and the N key rollover reverts to a 2 key rollover. To ensure that all 3-key combinations used on a PC (e.g., CNTRL+ALT+DEL) are still possible, keyboard manufacturers using this method organize the key switch matrix accordingly (an example is given in *Figure 4*).

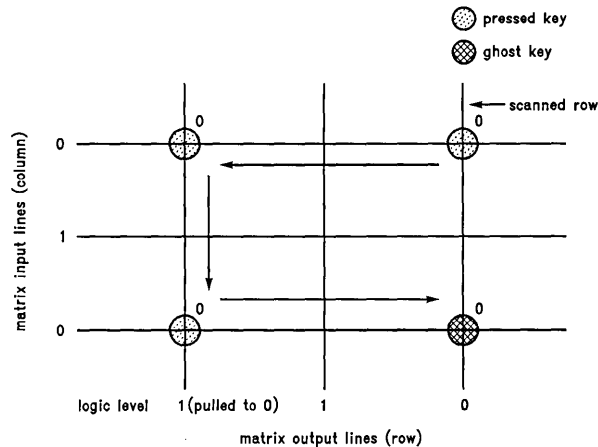


FIGURE 5. Software Key Rollover

TL/DD/11091-14

```

;      SOFTWARE KEY ROLLOVER
;
;LENGTHC: COUNTER FOR NO. OF BYTES (15 FOR A 16 BY 8 MATRIX)
;      WHICH HAVE TO BE COMPARED WITH THE ACTUAL SCANNED
;      BYTE.
;LASTSCN: RAM LOCATION WHICH CONTAINS THE RESULT OF THE ACTUAL
;      SCANNED LINE
;
;PNTSCAN: RAM LOCATION WHICH CONTAINS A POINTER TO THE RAM
;      CELL IN THE SCAN HISTORY TABLE THAT STORES THE RESULT
;      OF THE PREVIOUS SCAN FOR THE ACTUAL SCANNED MATRIX
;      LINE
;SCNLOT: START ADDRESS OF THE RAM SCAN HISTORY TABLE (16 BYTES)
;MATLEN: MATRIX LENGTH (IN THIS CASE MATLEN=16dec)
;BITC : SHIFT COUNTER FOR BYTE SHIFT
;TYPHAV: RAM ADDRESS OF TYPOMATIC RATE SAVE REGISTER
;TYPST : RAM ADDRESS FOR TYPOMATIC RATE VALUE
;STATUS: RAM ADDRESS OF GENERAL STATUS FLAG REGISTER
;STAT2 : RAM ADDRESS OF GENERAL STATUS FLAG REGISTER 2
;TYPCO1: RAM ADDRESS OF REGISTER THAT CONTAINS TYPOMATIC KEY
;      MAKE CODE
;SCNCNT: SCAN COUNTER FOR 16 MATRIX LINES
;
;
;
;      .LOCAL
KEYROL:
      LD      LENGTHC,#00F      ;LOAD TABLE LENGTH COUNTER
      LD      X,#LASTSCN      ;POINT TO RAM LOCATION WHERE
                                ;RESULT OF PREVIOUS SCAN IS
                                ;STORED
      LD      A,PNTSCAN      ;LOAD POINTER TO ACTUAL SCAN
                                ;LINE
      INC     A
      X      A,B      ;POINT TO THE NEXT SCAN LINE
$NEXTB:
      IFBNE   #((SCNLOT+MATLEN)&00F) ;IF END OF HISTORY SCANTABLE
                                ;IN RAM NOT REACHED
      JP      $1      ;THEN OK
      LD      B,#SCNLOT      ;ELSE POINT TO BEGINNING OF TABLE
$1:
      LD      A,[X]      ;COMPARE NEW SCANNED MATRIX LINE
      OR     A,[B]      ;WITH ALL OTHER PREVIOUS SCANNED
                                ;BYTES IN TABLE
      IFEQ   A,#0FF      ;IF NO KEYS PRESSED IN
                                ;SAME INPUT LINE
      JP      $INCB      ;THEN COMPARE WITH NEXT BYTE
                                ;IN SCAN TABLE
                                ;ELSE LOOK IF MORE THAN
                                ;TWO KEYS ARE PRESSED
                                ;IN ONE OF THE TWO
                                ;COMPARED BYTES
      LD      A,[X]      ;LOAD 1ST OF COMP.BYTES

```

TL/DD/11091-1

```

LD      BITC, #08      ;LOAD BIT COUNTER
$ZERO1: RRC      A
IFNC                    ;IF 1 KEY PRESSED
JP      $ZERO3        ;THEN TEST IF 2ND
                        ;KEY IS PRESSED
DRSZ    BITC          ;IF NOT ALL BITS CHECKED
JP      $ZERO1        ;THEN CONTINUE CHECK
JP      $INCB

$ZERO2: RRC      A
IFNC                    ;IF 2ND KEY PRESSED
JP      $ENDLP        ;THEN ERROR: "GHOST KEY"
$ZERO3: DRSZ    BITC   ;IF NOT ALL BITS CHECKED
JP      $ZERO2        ;THEN CONTINUE CHECK

$INCB:  LD      A, [B+] ;INC B
DRSZ    LENGTHC      ;IF NEW SCANNED MATRIX LINE
                        ;NOT COMPARED WITH ALL OTHER
                        ;BYTES IN TABLE
JP      $NEXTB       ;THEN COMP. WITH NEXT
                        ;BYTE IN TABLE
SC      ;IF ALL COMPARED, SET NO ERROR
                        ;FLAG

$ENDLP: LD      B, #STAT2 ;POINT TO STATUS FLAG REGISTER
IFNC                    ;ERROR DURING THIS SCAN?
JP      $ERROR        ;YES, DO ERROR PROCEDURE
IFBIT   ERR2, [B]     ;ERROR DURING PREVIOUS SCANS,
                        ;BUT NO ERROR DURING THIS
                        ;SCAN?
JP      $RESTORE      ;YES, RESTORE TYPEMATIC RATE
RET

$RESTORE: RBIT    ERR2, [B]
JSR     TSTOP         ;STOP TYPEMATIC TIMER
LD      A, TYPST      ;LOAD SAVED TYPEMATIC VALUE
X      A, TYPST      ;RESTORE OLD TYPEMATIC VALUE
RET     ;NO ERROR DURING THIS SCAN:
                        ;RETURN

$ERROR: IFBIT    ERR2, [B] ;IF ERROR OCURRED ALREADY
                        ;DURING PREVIOUS SCAN
JP      $ERREND      ;THEN DO NOTHING
SBIT   ERR2, [B]     ;ELSE SET PREVIOUS ERROR FLAG
LD      B, #TYPST    ;POINT TO TYPEMATIC VALUE
                        ;REGISTER
LD      A, [B]
X      A, TYPST      ;SAVE TYPEMATIC RATE/DELAY
LD      [B], #07F    ;SET TYPEMATIC TO 1s DELAY,
                        ;2 CHARACTERS/s FOR ERROR CODE

```

```

LD      A, #000                ;REPETITION
LD      B, #STATUS            ;IF SET2,3 ERROR CODE 00
IFBIT  SET1, [B]              ;POINT TO STATUS FLAG REGISTER
LD      A, #0FF                ;ELSE ERROR CODE FF
X       A, TYPCO1              ;PUT IN TYPEMATIC BUFFER
JSR     TSTART                 ;INIT & START TYPEMATIC TIMER

$SERREND:
LD      A, SCNCNT              ;INCREMENT SCAN COUNTER
INCA
X       A, SCNCNT              ;RET AND SKIP FOR ROLLOVER ERROR
RETSK
.LOCAL
.END

```

TL/DD/11091-3

Key Decoding and Encoding

After detection of a key change (pressing or releasing a key), the software first has to determine the physical location of the key in the key matrix. This decoding process is done by calculating an internal key number out of the key matrix column and row position of the changed key. At the same time, it is determined if the key has been pressed or released. A pressed or released key is then signaled by setting or resetting a "key down" flag in RAM. The internal key number and the "key down" status flag are the input parameters to the key encoding procedure. The internal key number is used to get the "make" code for the key out of a ROM look-up table, which has been matched to the physical matrix organization of the keyboard. If the "key down" flag is reset (key is released) the software calculates the key "break" code out of the previously fetched key "make" code. In this way, each pressed or released key is encoded with its appropriate "make" or "break" code, which is then written to the keyboard controllers 16 byte output buffer (FIFO) until the computer interface is ready to receive it. Before writing to the FIFO the software checks whether there is still enough capacity to store the key code.

Key Repetition

All keys are typematic (repetitive) by default. That means when a key is pressed and held down, the μ C continues to send the "make" code for that key until it is released. When two or more keys are held down, only the code for the last key pressed is repeated. Typematic operation will stop

when this key is released, even if other keys are still held down.

The default values for typematic operation are:

delay time = 500 ms
 repetition rate = 10 characters/second,

where the delay time is the time which is inserted before a character is repeated for the first time.

Operating Protocol

There are two different transmission protocols for an MF2 keyboard: the AT transmission protocol and the XT transmission protocol. Data transmission to and from the keyboard is synchronous serial, the data format for the XT mode is:

9 bits in length
 1 start bit (high)
 8 data bits (LSB first)

The data format for AT and PS/2 modes is:

11 bits in length
 1 start bit (low)
 8 data bits (LSB first)
 1 parity bit (odd)
 1 stop bit (high)

If no data is transmitted, both data and clock lines are in the high state. The clock signal is always provided by the keyboard. *Figure 6* shows the XT and the AT protocol timings.

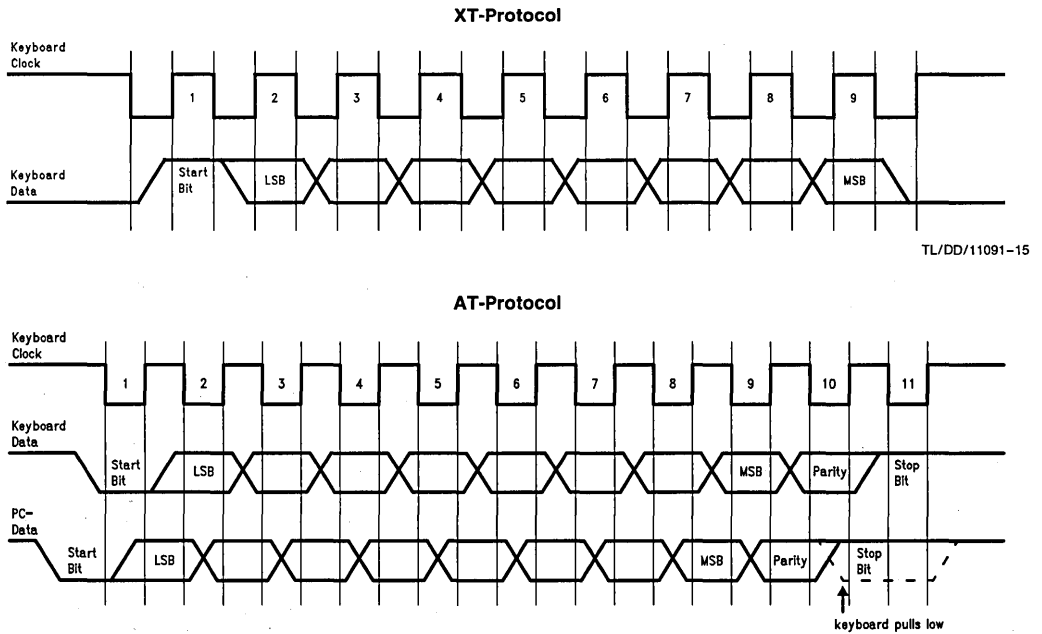


FIGURE 6. XT and AT Protocol Timings

Keyboard Data Transmission in XT Format

At the falling edge of the clock, the start bit (high) is shifted out, followed by the 8 data bits (least significant bit first). Data is valid on the rising edge of the clock and changes after the falling edge of the clock.

Keyboard Data Transmission in AT Format

Before sending data, the keyboard monitors the clock and data lines. If the clock line is low, then the keyboard is disabled by the computer and no data is transmitted. The microcontroller continues to scan the keyboard and stores key data in its output buffer. If the data line is low, while the clock line is high, the computer requests to send and the

keyboard goes into receive mode. The keyboard is only allowed to transmit data when both data and clock lines are high.

The keyboard pulls the data line low (start bit) and starts the clock. The 8 data bits (least significant bit first) are shifted out, followed by the parity (odd) and stop bit (high). Data is valid after the falling edge of the clock and changes after the rising edge of the clock. If no data is transmitted both data and clock lines are high. If the computer pulls the clock line low for at least 60 μ s before the 10th bit is transmitted, the keyboard stops transmission and stores the aborted data in its output buffer.

```

; SENDBY: SEND BYTE TO COMPUTER
;INPUT PARAMETER:
;BYTSEN: RAM LOCATION CONTAINING THE
; BYTE TO BE TRANSMITTED
;OUTPUT:
; DATSEN FLAG IN STATUS REGISTER
; 1=BYTE SENT,0=BYTE NOT SENT
;PARCNT: PARITY COUNTER REGISTER
;BITC : DATA LENGTH COUNTER FOR TRANSMISSION LOOP
;
;CLOCK HIGH TIME (=CLOCK LOW TIME) = 40us
;AT 3.58MHz CLOCK (INSTR. CYCLE = 2.79us)
;
;DATA REGISTER OF PORT G DATA AND CLOCK LINES IS
;PRESET WITH "0"

```

.LOCAL

```

SendBy:
LD      B,#STATUS      ;POINT TO STATUS FLAG REGISTER
RBIT   DatSen,[B]     ;RESET "BYTE SEND" FLAG
LD      A,BytSen       ;LOAD BYTE TO SEND
LD      BITC,#009      ;DATA LENGTH
IFBIT  PCXT,[B]       ;IF XT MODE
JMP     PCMode         ;THEN JUMP TO XT
                          ;SEND ROUTINE
                          ;ELSE SEND AT PROTOCOL

```

```

$ATSEND:
LD      PARCNT,#10     ;LOAD PARITY COUNTER
LD      B,#PORTGP     ;POINT TO GPORT INPUT
                          ;REGISTER

```

WAITS:

TL/DD/11091-4


```

IFBIT  ClockL, [B]      ;IF CLOCKLINE HIGH
JP      $ClocOK         ;THEN OK
JP      WAITS           ;ELSE KEYBOARD DISABLED:
                          ;WAIT

$ClocOK:
IFBIT  DataLn, [B]     ;IF DATALINE IS HIGH
JP      $DataOK        ;THEN OK
RET     ;ELSE PC SENDS DATA:
        ;RETURN (GOTO RECEIVE)

$DataOK:
LD      B, #PORTGC     ;POINT TO PORT G CONFIGURATION
                          ;REGISTER
RC      ;STARTBIT = 0

$SendBt:
RBIT   ClockL, [B]     ;SET CLOCKLINE HIGH (TRI-STATE)
IFBIT  ClockL, PORTGP  ;IF PC DOES NOT PULL CLOCKL LOW
JP      $ClockH        ;THEN OK
RBIT   DataLn, [B]     ;ELSE SET DATA LINE BACK TO HIGH
RET     ;STOP TO SEND

$ClockH:
IFC    ;IF BIT TO TRANSMIT = "1"
JP      $DATHI         ;THEN DATALINE HIGH
SBIT   DataLn, [B]     ;ELSE DATALINE LOW
JP      $CLKLOW        ;SET CLOCKLINE LOW

$DATHI:
RBIT   DataLn, [B]     ;SET DATALINE HIGH (TRI-STATE)

$CLKLOW:
SBIT   ClockL, [B]     ;SET CLOCKLINE LOW
IFC    ;IF BIT=1
DRSZ   PARCNT          ;THEN DECR. PARITY COUNTER
RRC    A               ;SHIFT NEXT BIT INTO CARRY
NOP
DRSZ   BITC            ;IF NOT ALL BITS SENT
JP      $SendBt        ;THEN TRANSMIT NEXT BIT

$PARITY:
NOP    ;SEND PARITY BIT
NOP    ;DELAY
NOP
RBIT   ClockL, [B]     ;SET CLOCKLINE HIGH
IFBIT  00, PARCNT      ;IF NUMBER OF "1" = ODD
JP      $DLOW          ;THEN PARITY = 0
RBIT   DataLn, [B]     ;ELSE PARITY = 1
JP      $CLKL2

$DLOW:
SBIT   DataLn, [B]     ;SET DATALINE LOW
NOP

$CLKL2:
NOP    ;DELAY
NOP

```

```

NOP
SBIT      ClockL, [B]      ;SET CLOCKLINE LOW
JSR      DEL12             ;INSERT DELAY 12 INSTR. CYCLES

RBIT      ClockL, [B]      ;SET CLOCKLINE HIGH

RBIT      DataLn, [B]      ;TRANSMIT STOP BIT
JSR      DEL11             ;SET DATA LINE HIGH (STOP BIT)
SBIT      Clock1, [B]      ;INSERT DELAY 11 INSTR. CYCLES
JSR      DEL12             ;SET CLOCKLINE LOW
;INSERT DELAY 12 INSTR. CYCLES

$ENDSB:
RBIT      ClockL, [B]      ;SET CLOCKLINE HIGH
RBIT      DATALN, [B]      ;DATA HIGH (XT MODE)
LD        B, #STATUS      ;POINT TO STATUS FLAG REG.
SBIT      DatSen, [B]      ;SET DATA SENT FLAG

LD        A, BYTSEN
IFEQ     A, #0FE           ;IF SENT BYTE = RESEND
;COMMAND
RET      ;THEN DON'T SAVE
X        A, SENBYT        ;ELSE SAVE LAST SENT BYTE
;IN SENBYT IN CASE PC ASKS
;KEYBOARD TO RESEND

RET

;XT TRANSMISSION PROTOCOL
PCMode:
IFBIT    CLOCKL, PORTGP   ;CLOCKLINE HIGH?
JP        $PCSND          ;YES, START TO SEND
JMP      POWRUP           ;ELSE RESET

$PCSND:
LD        B, #PORTGC
SBIT      DATALN, [B]     ;DATA LINE LOW BEFORE
;START TO SEND
;START BIT = 1

$PCSEND:
SBIT      ClockL, [B]     ;CLOCKLINE LOW
IFC       ;IF BIT TO SEND=1
JP        $DATH2          ;THEN SET DATALINE HIGH
SBIT      DataLn, [B]     ;ELSE SET DATALINE LOW
NOP
NOP
NOP
NOP
NOP
NOP
JP        $CLKHI

$DATH2:
RBIT      DataLn, [B]     ;SET DATALINE HIGH
IFBIT    DATALN, PORTGP   ;IF DATALINE HIGH
JP        $CLKHI          ;THEN OK
;ELSE KEYBOARD DISABLED

RBIT      CLOCKL, [B]     ;CLOCKLINE HIGH

```

TL/DD/11091-6

```

RET                                ;STOP TO SEND
$CLKHI:
RBIT    ClockL, [B]                ;SET CLOCKLINE HIGH
RRC     A                          ;SHIFT NEXT BIT TO TRANSMIT
                                              ;INTO CARRY
NOP                                           ;DELAY
NOP
NOP
NOP
$PCOK:
DRSZ    BITC                        ;IF NOT ALL BITS SENDEED
JP      $PCSEND                     ;THEN CONTINUE
SBIT    CLOCKL, [B]                 ;ELSE CLOCKLINE LOW
SBIT    DATALN, [B]                 ;DATA LOW
JSR     DELAYD                      ;10 INSTR. CYCLES DELAY
JP      $ENDSB

DEL12:  NOP
DEL11:  NOP
DELAYD: RET
        .LOCAL
        .END

```

TL/DD/11091-7

Keyboard Receives Data

The keyboard can only receive data from the computer in AT-PS/2 mode. The computer pulls the data line low (start bit) after which the keyboard starts to shift out 11 clock pulses within 15 ms. Transmission has to be completed within 2 ms. Data from the computer changes after the falling edge of the clock and is valid before the rising edge of

the clock. After the start bit, 8 data bits (least significant bit first), followed by the parity bit (odd) and the stop bit (high) are shifted out by the computer with the clock signal provided by the keyboard. The keyboard pulls the stop bit low in order to acknowledge the receipt of the data. If a transmission error occurred (parity error or similar) the keyboard issues the "RESEND" command to the PC.

```

;
; RECDAT: RECEIVE DATA COMMING FROM PC
;
;RETURN, IF PARITY ERROR
;
;RETURN SKIP , IF BYTE WAS RECEIVED
;WITHOUT ERROR
;
;BTRECV: RAM LOCATION CONTAINING THE
; RECEIVED BYTE
;
;
;BITC : RECEIVE LOOP COUNTER REGISTER
;PARCNT: PARITY COUNTER REGISTER
;

RecDat:

    CLRA
    LD     B, #PORTGC      ;B POINT TO PORT G
                                ;CONFIGURATION
    LD     X, #BTRECV      ;X POINT TO RECEIVED BYTE
                                ;RAM CELL
    LD     PARCNT, #10     ;LOAD PARITY COUNTER
    LD     BITC, #009     ;LOAD RECEIVE LOOP COUNTER
                                ;(8 DATABITS + 1 PARITY BIT)
    RC
                                ;START BIT= "0"

RdByte:

    SBIT   ClockL, [B]     ;SET CLOCKLINE LOW
                                ;(CLOCK IN START BIT)
    IFC
    DRSZ   PARCNT         ;IF "1"-BIT RECEIVED
                                ;THEN DECR. PARITY COUNTER
    RRC    A              ;SHIFT CARRY TO BIT 7 OF ACCU
    X      A, [X]         ;STORE RECEIVED BYTE
    LD     A, [X]         ;RESTORE AS LONG AS NOT
                                ;FULL BYTE RECEIVED

    RBIT   ClockL, [B]     ;SET CLOCKLINE HIGH
;READ IN RECEIVED BIT
    RC      ;RECEIVED BIT= "0"
    IFBIT  DataLn, PORTGP  ;IF DATALINE = "1"
    SC     ;THEN RECEIVED BIT= "1"
    DRSZ   BITC           ;9 BITS RECEIVED?
    JP     RdByte         ;NO, LOOP

;CLOCK LOW PULSE AFTER PARITY HAS BEEN RECEIVED
    SBIT   ClockL, [B]     ;SET CLOCKLINE LOW
    JSR   DELAYD         ;INSERT 10 INSTR. CYCLES DELAY
    RBIT   ClockL, [B]     ;SET CLOCKLINE HIGH
;PC SENDS STOP BIT
    SBIT   DataLn, [B]     ;PULL STOP BIT LOW

```

TL/DD/11091-8

```

;TO ACKNOWLEDGE RECEIPT OF BYTE
JSR    DELAYD                ;INSERT DELAY
;CLOCK LOW PULSE (CLOCK ACKNOWLEDGE FOR PC)
SBIT   ClockL, [B]          ;SET CLOCKLINE LOW
JSR    DELAYD                ;INSERT DELAY
RBIT   ClockL, [B]          ;SET CLOCKLINE HIGH

RBIT   DataLn, [B]          ;RETURN DATA TO HIGH

;PARITY CHECK
IFBIT  00, PARCNT            ;IF NO. OF RECEIVED DATA "1"=ODD
JP     PAR0                  ;THEN PARITY BIT MUST BE "0"
ParOne:                ;ELSE PARITY BIT MUST BE "1"
IFC    ;IF RECEIVED PARITY BIT=1
RETSK  ;THEN OK, RETURN SKIP
JP     PARERR                ;ELSE PARITY ERROR

PAR0:
IFNC   ;IF RECEIVED PARITY BIT =0
RETSK  ;THEN OK, RETURN SKIP
;ELSE PARITY ERROR

ParErr: LD    BytSen, #0FE    ;LOAD "RESEND" CODE
JSR    SByWpO                ;SEND RESEND CODE TO PC
RET    ;ERROR, RETURN
.END

```

TL/DD/11091-9

Commands from the Computer

The following table shows the commands and their hexadecimal values the computer may send to the keyboard. Only AT-PS/2 compatible computers can send commands to the keyboard and the keyboard can only receive the commands when operated in the AT-mode.

The commands can be sent to the keyboard at any time. The keyboard responds within 20 ms to any valid transmission with ACK (FA Hex), except for the ECHO command where the keyboard responds with EE Hex, the RESEND command and the reserved commands.

Command	Hex Value
Set/Reset Mode Indicators	ED
Echo	EE
Reserved	EF
Select Alternate Code Set	F0
Reserved	F1
Read Keyboard ID	F2
Set Typematic Rate/Delay	F3
Enable	F4
Default Disable	F5
Set Default	F6
Set All Keys	
Typematic/No Break	F7
Make/Break/No Typematic	F8
Make/No Typematic	F9
Typem./Make/Br.	FA
Set Key Type	
Typematic/No Break	FB
Make/Break/No Typematic	FC
Make/No Typematic	FD
Resend	FE
Reset	FF

In the XT mode the keyboard only accepts the RESET command, which is assumed when the computer pulls the clock line low for at least 10 ms.

Commands to the Computer

The following table shows the commands and their hexadecimal values the keyboard may send to the system.

Command	Hex Value
Key Detection Error/ Buffer Overrun	00 (Code Sets 2 and 3)
Keyboard ID	83AB
BAT Completion Code	AA
BAT Failure Code	FC
Echo	EE
Acknowledge	FA
Resend	FE
Key Detection Error/ Buffer Overrun	FF (Code Set 1)

SUMMARY

When using National Semiconductor's microcontroller to implement the functions of an MF2 keyboard, very few external components are necessary. *Figure 2* shows the complete schematic of an MF2 keyboard based on the COP888CL. The implementation of software key rollover eliminates the need for decoupling diodes in the 16 by 8 key matrix. LED direct drive capability of the COP8 and a RC oscillator with tolerances tight enough to meet the requirements for a keyboard further reduce component count and price. Schmitt triggers on the ports used for the keyboards data and clock lines add additional security against transmission errors. Where low power consumption is the most important design factor (e.g., laptop or notebook computers) the COP8's M2CMOS technology and the multi-input wakeup feature offer a remarkable improvement over the NMOS controllers used in most of today's existing solutions.

National Semiconductor offers three chips tailored for the needs of a keyboard designer. Starting with the most price competitive 2.5k ROM device COP943C, an upgrade path is provided with the COP880C to 4k ROM. Both devices are intended for the use in standard MF2 desktop keyboards. The COP888CL is ideally suited for notebook or lap-

top keyboards, as it has special power saving features. The complete software for an MF2 keyboard as well as complete demo keyboards and keyboard evaluation boards for the COP888CL and COP943C/COP880C microcontrollers are available. Contact National Semiconductor's μ C marketing or applications for further information.

APPENDIX I. KEY NUMBERS AND THEIR CORRESPONDING MAKE/BREAK CODES FOR ALL THREE CODE SETS

Key Position and Symbol	Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)		Table III (Terminal MODE)		
	Make	Break	Make	Break	Code	Type	
01	~	29	A9	0E	F0-0E	0E	Typematic
02	1	02	82	16	F0-16	16	Typematic
03	@ 2	03	83	1E	F0-1E	1E	Typematic
04	# 3	04	84	26	F0-26	26	Typematic
05	\$ 4	05	85	25	F0-25	25	Typematic
06	% 5	06	86	2E	F0-2E	2E	Typematic
07	^ 6	07	87	36	F0-36	36	Typematic
08	& 7	08	88	3D	F0-3D	3D	Typematic
09	* 8	09	89	3E	F0-3E	3E	Typematic
10	(9	0A	8A	46	F0-46	46	Typematic
11) 0	0B	8B	45	F0-45	45	Typematic
12	_ -	0C	8C	4E	F0-4E	4E	Typematic
13	+ =	0D	8D	55	F0-55	55	Typematic
15	B.S. ←	0E	8E	66	F0-66	66	Typematic
16	TAB	0F	8F	0D	F0-0D	0D	Typematic
17	Q	10	90	15	F0-15	15	Typematic
18	W	11	91	1D	F0-1D	1D	Typematic
19	E	12	92	24	F0-24	24	Typematic
20	R	13	93	2D	F0-2D	2D	Typematic
21	T	14	94	2C	F0-2C	2C	Typematic
22	Y	15	95	35	F0-35	35	Typematic
23	U	16	96	3C	F0-3C	3C	Typematic
24	I	17	97	43	F0-43	43	Typematic
25	O	18	98	44	F0-44	44	Typematic
26	P	19	99	4D	F0-4D	4D	Typematic
27	{ [1A	9A	54	F0-54	54	Typematic
28	}]	1B	9B	5B	F0-5B	5B	Typematic
29*	\	2B	AB	5D	F0-5D	5C	Typematic
30	Caps Lk	3A	BA	58	F0-58	14	Make/Break
31	A	1E	9E	1C	F0-1C	1C	Typematic
32	S	1F	9F	1B	F0-1B	1B	Typematic
33	D	20	A0	23	F0-23	23	Typematic
34	F	21	A1	2B	F0-2B	2B	Typematic
35	G	22	A2	34	F0-34	34	Typematic

Key Position and Symbol		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)		Table III (Terminal MODE)	
		Make	Break	Make	Break	Code	Type
36	H	23	A3	33	F0-33	33	Typematic
37	J	24	A4	3B	F0-3B	3B	Typematic
38	K	25	A5	42	F0-42	42	Typematic
39	L	26	A6	4B	F0-4B	4B	Typematic
40	: ;	27	A7	4C	F0-4C	4C	Typematic
41	" ' ,	28	A8	52	F0-52	52	Typematic
42**	\	2B	AB	5D	F0-5D	53	Typematic
43	Enter (L)	1C	9C	5A	F0-5A	5A	Typematic
44	Shift (L)	2A	AA	12	F0-12	12	Typematic
45**	Macro	56	D6	61	F0-61	13	Typematic
46	Z	2C	AC	1A	F0-1A	1A	Typematic
47	X	2D	AD	22	F0-22	22	Typematic
48	C	2E	AE	21	F0-21	21	Typematic
49	V	2F	AF	2A	F0-2A	2A	Typematic
50	B	30	B0	32	F0-32	32	Typematic
51	N	31	B1	31	F0-31	31	Typematic
52	M	32	B2	3A	F0-3A	3A	Typematic
53	< ,	33	B3	41	F0-41	41	Typematic
54	> .	34	B4	49	F0-49	49	Typematic
55	? /	35	B5	4A	F0-4A	4A	Typematic
57	Shift (R)	36	B6	59	F0-59	59	Make/Break
58	Ctrl (L)	1D	9D	14	F0-14	11	Make/Break
60	Alt (L)	38	B8	11	F0-11	19	Make/Break
61	Space	39	B9	29	F0-29	29	Typematic
62	Alt (R)	E0-38	E0-B8	E0-11	E0-F0-11	39	Make
64	Ctrl (R)	E0-1D	E0-9D	E0-14	E0-F0-14	58	Make
90	Num Lk	45	C5	77	F0-77	76	Make
91	7 Home	47	C7	6C	F0-6C	6C	Make
92	4 ←	4B	CB	6B	F0-6B	6B	Make
93	1 End	4F	CF	69	F0-69	69	Make
96	8 ↑	48	C8	75	F0-75	75	Make
97	5	4C	CC	73	F0-73	73	Make
98	2 ↓	50	D0	72	F0-72	72	Make
99	0 Ins	52	D2	70	F0-70	70	Make
100	*	37	B7	7C	F0-7C	7E	Make

*101-Keyboard only

**102-Keyboard only

Key Position and Symbol		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)		Table III (Terminal MODE)	
		Make	Break	Make	Break	Code	Type
101	9 Pg UP	49	C9	7D	F0-7D	7D	Make
102	6 →	4D	CD	74	F0-74	74	Make
103	3 Pg DN	51	D1	7A	F0-7A	7A	Make
104	Del	53	D3	71	F0-71	71	Make
105	-	4A	CA	7B	F0-7B	84	Make
106	+	4E	CE	79	F0-79	7C	Make
108	Enter	E0-1C	E0-9C	E0-5A	E0-F0-5A	79	Typematic
110	Esc	01	81	76	F0-76	08	Make
112	F1	3B	BB	05	F0-05	07	Make
113	F2	3C	BC	06	F0-06	0F	Make
114	F3	3D	BD	04	F0-04	17	Make
115	F4	3E	BE	0C	F0-0C	1F	Make
116	F5	3F	BF	03	F0-03	27	Make
117	F6	40	C0	0B	F0-0B	2F	Make
118	F7	41	C1	83	F0-83	37	Make
119	F8	42	C2	0A	F0-0A	3F	Make
120	F9	43	C3	01	F0-01	47	Make
121	F10	44	C4	09	F0-09	4F	Make
122	F11	57	D7	78	F0-78	56	Make
123	F12	58	D8	07	F0-07	5E	Make
125	Scr Lk	46	C6	7E	F0-7E	5F	Make

Key Position and Symbol		Cursor Pad <NUM Lock Off/Shift Off> or <NUM Lock On/Shift On>				Table III (Terminal Mode)	
		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)			
		Make	Break	Make	Break	Code	Type
75	Insert	E0-52	E0-D2	E0-70	E0-F0-70	67	Make
76	Delete	E0-53	E0-D3	E0-71	E0-F0-71	64	Typematic
79	←	E0-4B	E0-CB	E0-6B	E0-F0-6B	61	Typematic
80	Home	E0-47	E0-C7	E0-6C	E0-F0-6C	6E	Make
81	End	E0-4F	E0-CF	E0-69	E0-F0-69	65	Make
83	↑	E0-48	E0-C8	E0-75	E0-F0-75	63	Typematic
84	↓	E0-50	E0-D0	E0-72	E0-F0-72	60	Typematic
85	PG UP	E0-49	E0-C9	E0-7D	E0-F0-7D	6F	Make
86	PG DN	E0-51	E0-D1	E0-7A	E0-F0-7A	6D	Make
89	→	E0-4D	E0-CD	E0-74	E0-F0-74	6A	Typematic

*. Cursor Pad Key—<NUM Lock On/Shift Off>

Table I: Make Code == E0-2A—Make Code

Break Code == Break Code—E0-AA

Table II: Make Code == E0-12—Make Code

Break Code == Break Code E0-F0-12

*. Cursor Pad Key—<NUM Lock Off/Shift On>

Table I: Make Code = E0-AA—Make Code

Break Code = Break Code—E0-2A

Table II: Make Code = E0-F0-12—Make Code

Break Code = Break Code E0-12

Key Code of "Pause", "PRTSC" and "/" Keys

TABLE I. XT and PS/2 30

Key Position and Symbols		Make	Break
126	Pause	E1-1D-45-E1-9D-C5	No Break Code (Make Only)
	Ctrl-"Pause"	E0-46-E0-C6	No Break Code (Make Only)
124	Print Screen	E0-2A-E0-37	E0-B7-E0-AA
	Shift-"PRTSC"	E0-37	E0-B7
	Ctrl-"PRTSC"	E0-37	E0-B7
	Alt-"PRTSC"	54	D4
95	/	E0-35	E0-B5
	Shift-"/"	E0-AA-E0-35	E0-B5-E0-2A

TABLE II. AT and PS/2 50, 60, 80

Key Position and Symbols		Make	Break
126	Pause	E1-14-77-E1-F0-14-F0-77	No Break Code (Make Only)
	Ctrl-"Pause"	E0-7E-E0-F0-7E	No Break Code (Make Only)
124	Print Screen	E0-12-E0-7C	E0-F0-7C-E0-F0-12
	Shift-"PRTSC"	E0-7C	E0-F0-7C
	Ctrl-"PRTSC"	E0-7C	E0-F0-7C
	Alt-"PRTSC"	84	F0-84
95	/	E0-4A	E0-F0-4A
	Shift-"/"	E0-F0-12-E0-4A	E0-F0-4A-E0-12

TABLE III. Terminal Mode

Key Position and Symbols		Code	Type
126	Pause	62	Make
124	Print Screen	57	Make
95	/	77	Make

APPENDIX II. REFERENCES

1. IBM Technical Reference Manuals XT, AT and PS/2
2. Chicony, Chicony Keyboards General Specification, 1988
3. C' T Magazin fuer Computertechnik, No. 6, 1988, pages 148ff. No. 7, 1988, pages 178ff. Martin Gerdes, "Knoepfchen, Knoepfchen"

RS-232C Interface with COP800

National Semiconductor
Application Note 739
Michelle Giles



INTRODUCTION

This application note describes an implementation of the RS-232C interface with a COP888CG. The COP888CG 8-bit microcontroller features three 16-bit timer/counters, MICROWIRE/PLUSTM Serial I/O, multi-source vectored interrupt capability, two comparators, a full duplex UART, and two power saving modes (HALT and IDLE). The COP888CG feature set allows for efficient handling of RS-232C hardware handshaking and serial data transmission/reception.

SYSTEM OVERVIEW

In this application, a COP888CG is connected to a terminal using the standard RS-232C interface. The serial port of the terminal is attached to the COP888CG interface hardware using a standard ribbon cable with DB-25 connectors on either end. The terminal keyboard transmits ASCII characters via the cable to the COP888CG interface. All characters received by the COP888CG are echoed back to the terminal screen. If the COP888CG detects a parity or framing error, it transmits an error message back to the terminal screen.

HARDWARE DESCRIPTION

The COP888CG features used in this application include the user programmable UART, the 8-bit configurable L PORT, and vectored interrupts. In addition to the COP888CG, the RS-232C interface requires a DS14C88 driver and a DS14C89A receiver. The DS14C88 converts TTL/CMOS level signals to RS-232C defined levels and the DS14C89A does the opposite. *Figure 1* contains a diagram of the COP888CG interface hardware.

The COP888CG is configured as data communications equipment (DCE) and the terminal is assumed to be data terminal equipment (DTE). The following RS-232C signals are used to communicate between the COP888CG (DCE) and the terminal (DTE):

RS-232C Signal Name	Signal Origin
TxD (Transmit Data)	DTE
RxD (Receive Data)	DCE
CTS (Clear To Send)	DCE
RTS (Request To Send)	DTE
DSR (Data Set Ready)	DCE
DTR (Data Terminal Ready)	DTE
DCD (Data Carrier Detect)	DCE

Five general purpose I/O pins on the COP888CG L PORT are used for the control signals CTS, DSR, DCD, RTS and DTR. Two additional L PORT pins are used for TxD and RxD. These two general purpose pins are configured for their alternate functions, UART transmit (TDX) and UART receive (RDX). According to the RS-232C interface standard, DCE transmits data to DTE on RxD and receives data from DTE on TxD. Therefore, the UART transmit data pin (TDX) is used for the RS-232C receive data signal (RxD) and the UART receive data pin (RDX) is used for the RS-232C transmit data signal (TxD). In this example, all handshaking between DCE and DTE is performed in hardware.

The terminal is setup to interface with the COP888CG by selecting the 9600 baud, 7 bits/character, odd parity and one stop bit options. The local echo back of characters is disabled to allow the COP888CG to perform the echo back function. The terminal is also configured to use the hardware control signals (CTS, DSR, RTS, DTR) for handshaking.

SOFTWARE DESCRIPTION

The software for this application consists of an initialization routine, several interrupt routines, and a disable routine. These routines handle RS-232C handshaking, transmitting and receiving of characters, error checking, and echoing back of received characters. *Figures 2* thru *5* contain flowcharts of the routines. The complete code is given at the end of this application note.

The initialization routine configures the UART, initializes the transmit/receive data buffer, and enables the 8-bit L PORT handling of RS-232C control signals. In this particular example, the UART is configured to operate at 9600 BAUD in full duplex, asynchronous mode. The framing format is chosen to be: 7 bits/character, odd parity, and one stop bit. Different baud rates, modes of operation, and framing formats may be selected by setting the ENUCMD, ENUICMD, BAUDVAL and PSRVAL constants located at the beginning of the code to alternative values. (Refer to the COP888CG data sheet or COP888 Family User's Manual for details on configuring the UART.) Each RS-232C control signal is assigned to an L PORT pin. Pins L0, L2, L5 and L6 are configured as outputs for the DCD, TxD, CTS and DSR signals, respectively. Pins L3, L4 and L7 are configured as inputs for TxD, RTS and DTR, respectively. The transmit/receive data buffer is a circular buffer whose location and size is selected by setting the START and END constants located at the beginning of the program. The initialization routine sets up the buffer based on these constants.

The interrupt routines respond to transmit buffer empty, receive buffer full, and L PORT interrupts. A generic context switching routine is used for entering and exiting all interrupts. This routine saves the contents of the accumulator, the PSW register and the B pointer before vectoring to the appropriate interrupt routine. It also restores the contents of saved registers before a return from interrupt is executed.

The UART transmitter interrupt is called when the transmit buffer empty flag (TBMT) is set. This routine checks for active RTS and DTR control signals. If both signals are active and there is data to be transmitted, a byte of data is loaded into the UART transmit buffer. Otherwise, the UART transmitter is disabled.

The L PORT interrupts are used to indicate an active-low transition of RTS and/or DTR. When both signals are active (the remote receiver is ready to accept data), this routine enables the UART transmitter.

The UART receiver interrupt routine is called when the receive buffer full flag (RBFL) is set. This routine reads the

UART receive buffer and checks for errors. If no errors are detected, the incoming data is placed in the data buffer for echoing. If errors are detected, an error message is queued for transmission.

The receiver interrupt disables the remote transmitter by deactivating CTS whenever the transmit/receive data buffer is almost full. This action prevents the data buffer from overflowing. Note that CTS is turned off before the buffer is completely full to insure buffer space will exist for storing characters which are in the process of being sent when CTS is deactivated.

The disable routine clears the UART control registers, disables the L PORT interrupts, and de-activates the RS-232C control signals.

CONCLUSION

The user configurable UART, multiple external interrupt capabilities, and vectored interrupt scheme of the COP888CG microcontroller allow for an efficient implementation of the RS-232C interface standard. This application note shows how the COP888CG may be configured for connection to a terminal using these features. However, the code for this application can be easily adapted to other applications requiring different baud rates or framing formats, connection to a modem (DCE), separate transmit and receive buffers, incoming command decoding and/or handling of character strings. The versatility of the RS-232C standard and the COP888CG provides a means to develop practical solutions for many applications.

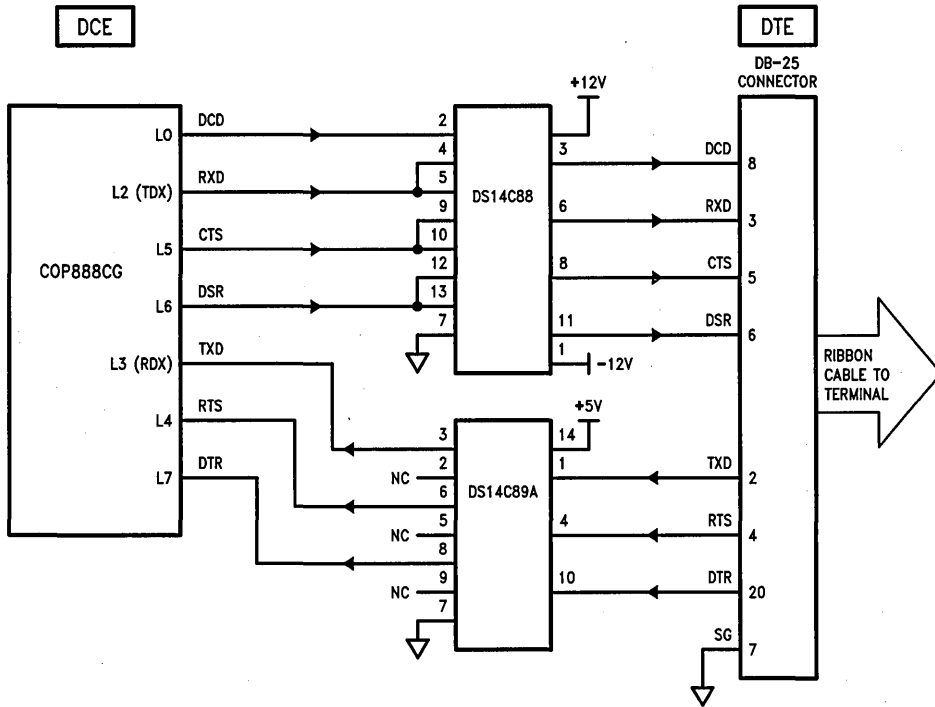


FIGURE 1. Interface Diagram

TL/DD/11110-1

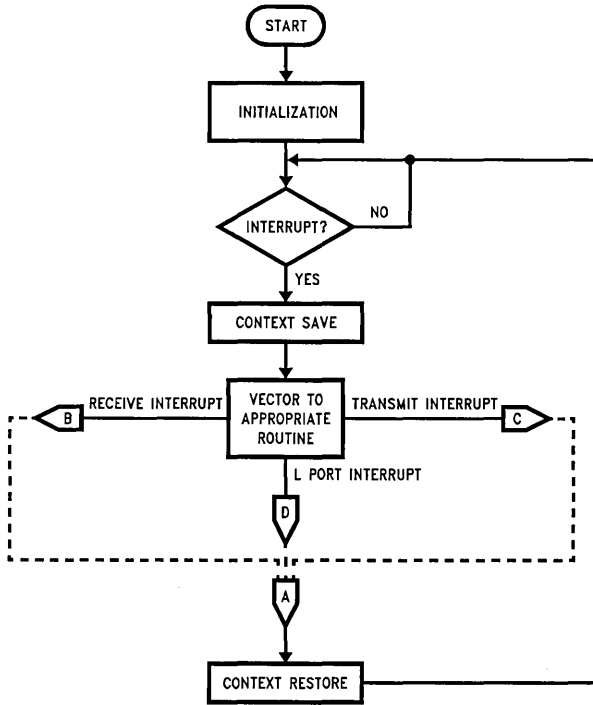


FIGURE 2. Main Program Flow

TL/DD/11110-2

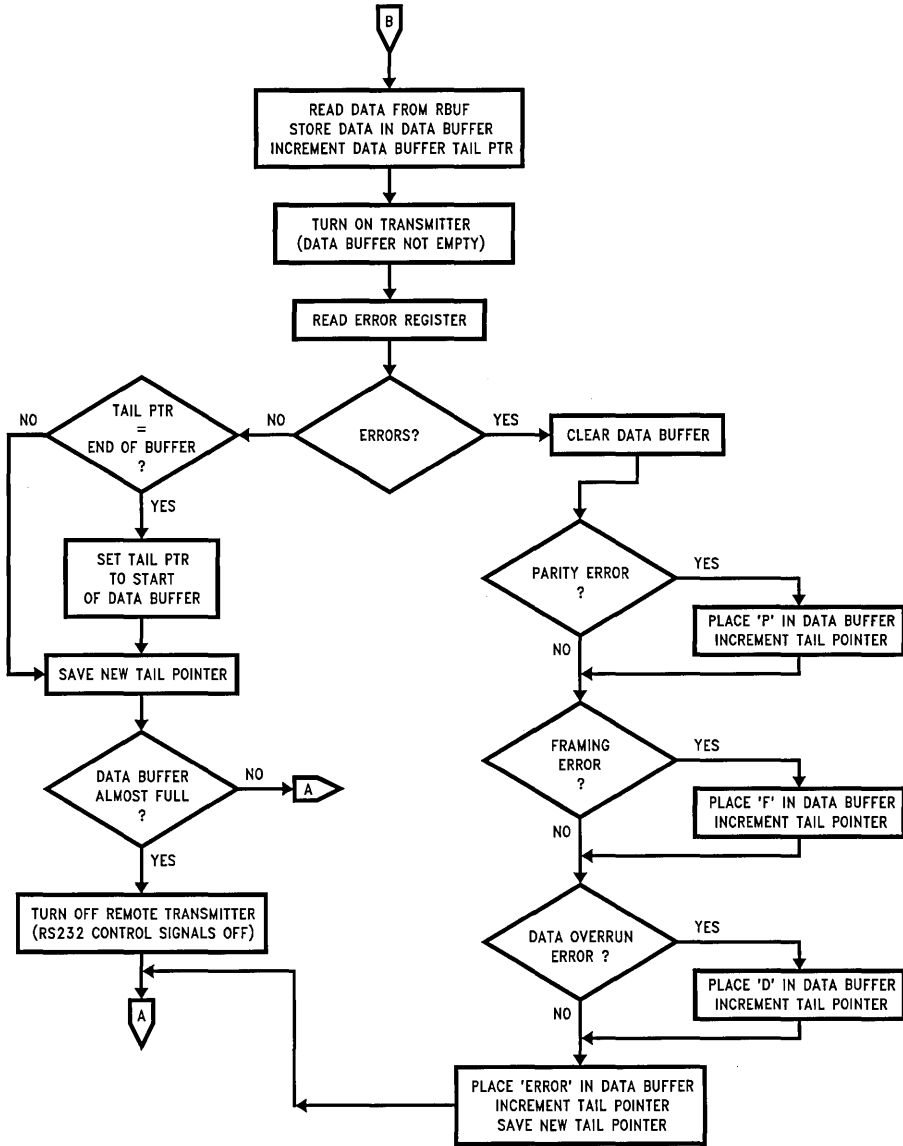


FIGURE 3. Receiver Interrupt Routine

TL/DD/11110-3

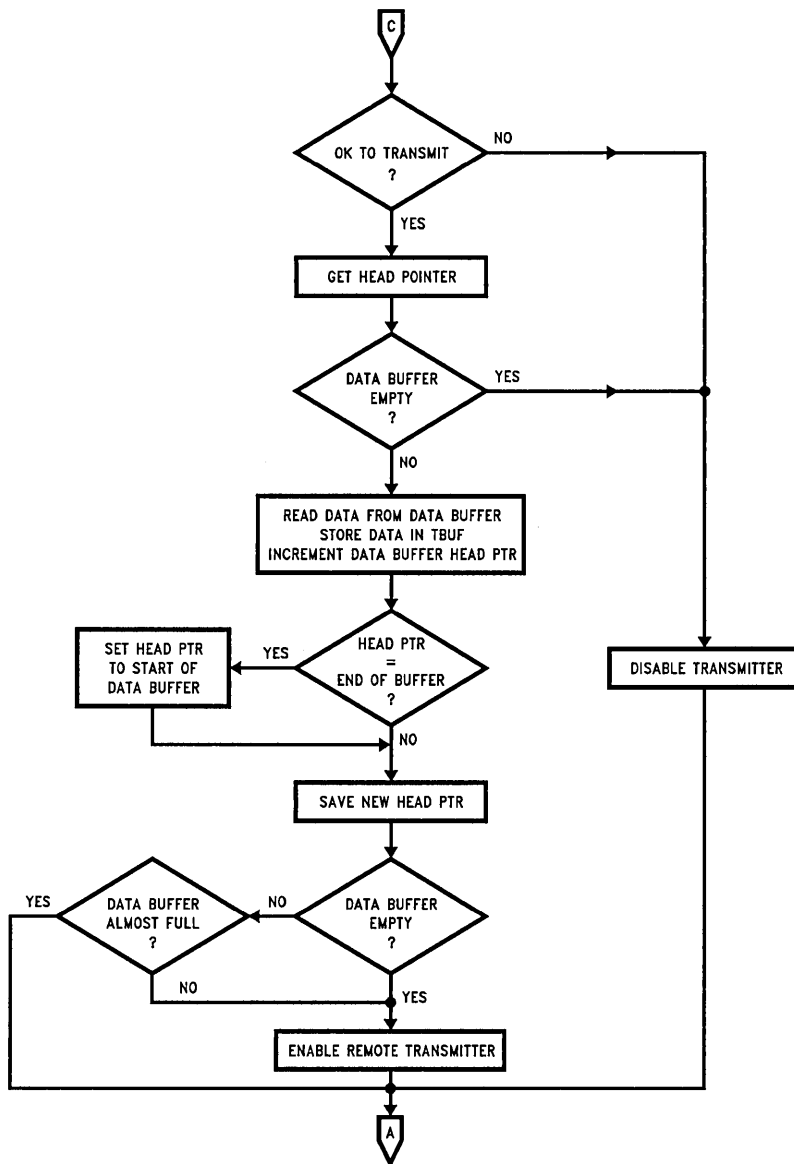
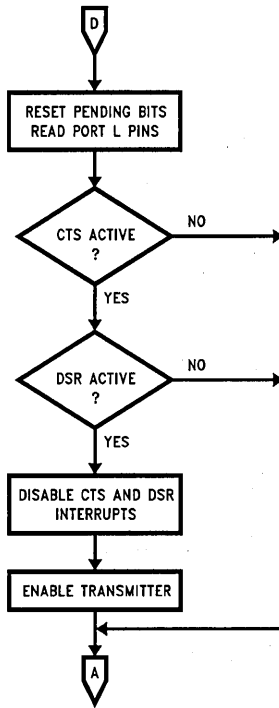


FIGURE 4. Transmitter Interrupt Routine

TL/DD/11110-4



TL/DD/11110-5

FIGURE 5. L Port Interrupt Routine

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88

```

1      ;The following set of routines uses the COP888CG UART and several I/O pins
2      ;to simulate an RS232 port interface. The code handles hardware control
3      ;signals, echo back of received characters, and error checking. A single
4      ;routine called INIT initializes the UART and hardware control signals.
5      ;The transmitting and receiving of characters is handled in several
6      ;interrupt routines. The UART is disabled by calling the DISABLE routine.
7      ;The user must select values for several constants before compiling
8      ;this code.
9      ;
10     ;NOTES:
11     ; * The COP transmitter is enabled only when the transmit/receive
12     ;   buffer is not empty and the appropriate RS232 control signals
13     ;   from the remote receiver are present.
14     ; * The COP receiver is always enabled. the remote transmitter
15     ; * The remote transmitter is disabled whenever the transmit/
16     ;   receive data buffer is full.
17
18     ;Definition of Constants
19     0089      ENUCMD = 089      ;Value to put in the ENU register
20     ;         ;Selects bits per char and parity option
21     ;         ;DEFAULT = 081 (7 bits/char and odd parity)
22
23     0020      ENUICMD = 020     ;Value to put in the ENUI register
24     ;         ;Selects number of stop bits, uart clock option,
25     ;         ;sync/async option, xmit/rcv interrupt enable,
26     ;         ;and TDX pin enable
27     ;         ;DEFAULT = 023 ( 1 stop bit, internal BRG,
28     ;         ;async operation, no interrupt, and TDX enabled)
29
30     0004      BAUDVAL = 04      ;Baud rate divisor equals N - 1
31     00C8      PSRVAL = 0C8     ;Baud rate prescalar
32     ;         ; BR = FC/(16 * N*P) where
33     ;         ;   FC = CKI frequency
34     ;         ;   N = Baud Divisor
35     ;         ;   P = Prescalar
36     ;GIVEN:          CALCULATE:  BAUDVAL:  PSRVAL:
37     ;CKI = 10MHz    N = 5
38     ;BR = 9600      P = 13        04      0C8
39     ;
40     ;CKI = 10MHz    N = 10
41     ;BR = 4800      P = 13        09      0C8
42     ;
43     ;See tables in users manual for translation
44     ;of N and P to BAUDVAL and PSRVAL
45
46     0010      START = 010     ;Beginning address of the xmit/rcv buffer
47     001D      END = 01D       ;End address of the xmit/rcv buffer
48     001E      HEAD = 01E      ;RAM address where current head of buffer stored
49     001F      TAIL = 01F      ;RAM address where current tail of buffer stored
50     000E      SIZE = 0E       ;Size of transmit/receive data buffer
51

```

TL/DD/11110-6

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88

```

52 0000 DCD = 00 ;Bit position of DCD signal on L port pins
53 0005 CTS = 05 ;Bit position of CTS signal on L port pins
54 0007 DTR = 07 ;Bit position of DTR signal on L port pins
55 0004 RTS = 04 ;Bit position of RTS signal on L port pins
56 0006 DSR = 06 ;Bit position of DSR signal on L port pins
57 0005 ETDX = 05 ;Bit position of TDX enable pin in ENUI
58 0000 TIE = 00 ;Bit position of TX interrupt enable bit
59 0001 RIE = 01 ;Bit position of RX interrupt enable bit
60 0005 PE = 05 ;Bit position of parity error in ENUR
61 0006 FE = 06 ;Bit position of framing error in ENUR
62 0007 DOE = 07 ;Bit position of data overrun error in ENUR
63
64
65

```

```
.INCLD COP888. INC
```

```

66
67
68 0002 3008 MAIN: JSR INIT ;INITIALIZE UART
69 0004 FF JP . ;DO OTHER TASKS
70 0005 3044 JSR DISABLE ;DISABLE UART
71 0007 FF JP . ;DO OTHER TASKS
72
73
74 0008 9FEF INIT: LD B, #PSW
75 000A 68 RBIT GIE, [B] ;DISABLE ALL INTERRUPTS
76 000B BCBE00 LD PSR, #00 ;UART OFF (POWERDOWN)
77 000E BCD165 LD PORTLC, #065 ;SET I/O
78 0011 9FD0 LD B, #PORTLD ;NOT READY TO RECEIVE
79 0013 7E SBIT DSR, [B] ; TURN OFF DATA SET READY
80 0014 7D SBIT CTS, [B] ; TURN OFF CLEAR TO SEND
81 0015 68 RBIT DCD, [B] ; TURN ON DATA CARRIER DETECT
82 0016 BC1E10 LD HEAD, #START ;INIT HEAD POINTER
83 0019 BC1F10 LD TAIL, #START ;INIT TAIL POINTER
84 001C 9FE8 LD B, #ICNTRL ;CONFIGURE PORTL INTERRUPTS
85 001E 6E RBIT LPEN, [B] ; DISABLE PORTL INTERRUPTS
86 001F BCC890 LD WKEDG, #090 ; SELECT FALLING EDGE FOR RTS AND DTR
87 0022 BCC990 LD WKEN, #090 ; ENABLE RTS AND DTR INTERRUPT
88 0025 BCCA00 LD WKPND, #00 ; CLEAR PORTL INTERRUPT PENDING FLAGS
89 0028 7E SBIT LPEN, [B] ; ENABLE PORT L INTERRUPTS
90 0029 BCBA89 LD ENU, #ENUCMD ;SELECT BITS/CHAR AND PARITY OPTION
91 002C BCBB00 LD ENUR, #00 ;CLEAR ERROR BITS
92 002F BCBC20 LD ENUI, #ENUICMD ;SELECT CLOCK, INTERRUPTS, STOPBITS
93 0032 BCBD04 LD BAUD, #BAUDVAL ;SETUP BRG
94 0035 9FBC LD B, #ENUI
95 0037 78 SBIT TIE, [B] ;ENABLE TRANSMITTER INTERRUPT
96 0038 79 SBIT RIE, [B] ;ENABLE RECEIVER INTERRUPT
97 0039 BCBE08 LD PSR, #PSRVAL ;UART ON
98 003C 9FD0 LD B, #PORTLD ;READY TO RECEIVE
99 003E 6E RBIT DSR, [B] ; TURN ON DATA SET READY
100 003F 6D RBIT CTS, [B] ; TURN ON CLEAR TO SEND
101 0040 9FEF LD B, #PSW
102 0042 78 SBIT GIE, [B] ;ENABLE ALL INTERRUPTS

```

TL/DD/11110-7

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88

```

103 0043 8E          RET
104
105          DISABLE:
106 0044 BDEF68      RBIT   GIE, PSW          ;DISABLE INTERRUPTS
107 0047 BCD061      LD     PORTLD, #061      ;TURN OFF HANDSHAKING SIGNALS
108 004A BCBE00      LD     PSR, #00          ;UART POWERDOWN
109 004D BCBA00      LD     ENU, #00         ;CLEAR UART CONTROL REGISTERS
110 0050 BCBC00      LD     ENUI, #00
111 0053 BCBB00      LD     ENUR, #00
112 0056 9FC9        LD     B, #WKEN         ;DISABLE RTS AND DTR INTERRUPTS
113 0058 6C          RBIT   RTS, [B]
114 0059 6F          RBIT   DTR, [B]
115 005A BDEF78      SBIT   GIE, PSW          ;ENABLE INTERRUPTS
116 005D 8E          RET
117
118
119          ;INTERRUPT ROUTINES
120
121          00FF          . = 0FF          ;INTERRUPT START ADDRESS
122 00FF 67          PUSH  A          ;CONTEXT SAVE
123 0100 9DFE        LD     A, B
124 0102 67          PUSH  A
125 0103 9DEF        LD     A, PSW
126 0105 67          PUSH  A
127 0106 B4          VIS
128 0107 8C          REST: POP  A          ;CONTEXT RESTORE
129 0108 9CEF        X     A, PSW
130 010A 8C          POP  A
131 010B 9CFE        X     A, B
132 010D 8C          POP  A
133 010E 8F          RETI
134
135
136          ;PORT L INTERRUPTS
137          ; The port L interrupts are used to indicate a return to active
138          ; state of the DTR and RTS signals from the remote receiver.
139          ; If both DTR and RTS are active, the remote receiver is ready
140          ; to accept data and the COP transmitter is enabled.
141
142          LINT:
143 010F BCCA00      LD     WKPND, #00      ;PORT L INTERRUPT
144 0112 9DD2        LD     A, PORTLP      ;RESET PENDING BITS
145 0114 6010        IFBIT #RTS, A       ;READ PORT L PINS
146 0116 06          JP     NOTRDY        ;IF RTS (ACTIVE LOW) NOT PRESENT
147 0117 60B0        IFBIT #DTR, A       ;THEN REMOTE NOT READY TO RECEIVE
148 0119 03          JP     NOTRDY        ;IF DTR (ACTIVE LOW) NOT PRESENT
149 011A 9FBC        READY: LD     B, #ENUI    ;THEN REMOTE NOT READY TO RECEIVE
150 011C 78          SBIT   TIE, [B]     ;RE-ENABLE TRANSMITTER INTERRUPT
151 011D E9          NOTRDY: JP    REST   ;EXIT INTERRUPT
152
153

```

TL/DD/11110-8

NATIONAL SEMICONDUCTOR CORPORATION
 COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88

```

154          ;UART RECEIVE INTERRUPT
155          ; The UART receive interrupt does the following:
156          ;   1. Reads the received data
157          ;   2. Checks for receiver errors
158          ;   3. If no errors detected, places the received data in
159          ;       the transmit/receive buffer and enables the transmitter.
160          ;   4. If errors detected, the transmit/receive buffer is cleared
161          ;       of ALL data and an error message is placed in the data buffer.
162          RCVINT:          ;RECEIVER INTERRUPT
163 011E 9D1F          LD      A, TAIL
164 0120 9CFE          X      A, B          ;GET TAIL POINTER
165 0122 9DB9          LD      A, RBUF          ;READ RECEIVED DATA
166 0124 A2           X      A, [B+]          ;STORE RECEIVED DATA
167 0125 9DBB          LD      A, ENUR          ;READ ERROR REGISTER
168 0127 BDBC78       SBIT   TIE, ENUI          ;ENABLE TRANSMITTER INTERRUPT
169 012A 60E0         ANDSZ  A, #0E0          ;CHECK FOR PE, DOE, FE
170 012C 1A          JP      ERROR          ;THROW DATA AWAY IN BUFFER
171 012D 9DFE          LD      A, B          ;LOAD ACC WITH NEW TAIL PTR
172 012F 921E         IFEQ   A, #END+1          ;IF END OF DATA BUFFER
173 0131 9810         LD      A, #START          ; SET TAIL PTR TO START OF BUFFER
174 0133 9C1F          X      A, TAIL          ;SAVE TAIL PTR
175 0135 9D1E         LD      A, HEAD          ;IS DATA BUFFER FULL?
176 0137 A1          SC
177 0138 BD1F81       SUBC   A, TAIL          ;A = HEAD - TAIL
178 013B 89          IFNC          ;IF BORROWED (TAIL ) HEAD)
179 013C 940E         ADD     A, #SIZE          ;THEN ADD BUFFER SIZE TO RESULT
180 013E 9303         IFGT   A, #03          ;IF DATA BUFFER NOT FULL
181 0140 2107         JMP     REST          ; THEN EXIT INTERRUPT
182 0142 BDD07D       RXOFF: SBIT   CTS, PORTLD          ; ELSE TURN OFF REMOTE TRANSMITTER
183 0145 2107         JMP     REST          ;EXIT INTERRUPT
184
185 0147 BC1E10       ERROR:  LD      HEAD, #START          ;CLEAR BUFFER
186 014A 9F10         LD      B, #START          ;POINT TO START OF BUFFER
187 014C 6020         IFBIT  PE, A
188 014E 9A50         LD      [B+], #'P'          ;P = PARITY
189 0150 6040         IFBIT  FE, A
190 0152 9A46         LD      [B+], #'F'          ;F = FRAMING
191 0154 6080         IFBIT  DOE, A
192 0156 9A44         LD      [B+], #'D'          ;D = DATA OVERRUN
193 0158 9A20         LD      [B+], #020          ;BLANK SPACE
194 015A 9A45         LD      [B+], #'E'
195 015C 9A52         LD      [B+], #'R'
196 015E 9A52         LD      [B+], #'R'
197 0160 9A4F         LD      [B+], #'D'
198 0162 9A52         LD      [B+], #'R'
199 0164 9A0A         LD      [B+], #0A          ;LINE FEED
200 0166 9A0D         LD      [B+], #0D          ;CARRIAGE RETURN
201 0168 9DFE         OUTERR: LD      A, B          ;SAVE NEW TAIL PTR
202 016A 9C1F          X      A, TAIL
203 016C 2107         JMP     REST
204

```

TL/DD/11110-9

```

205
206          ;UART TRANSMIT INTERRUPT
207          ; The UART transmit interrupt does the following:
208          ;   1. Checks for RTS and DTR signals (OK to transmit?)
209          ;   3. If OK to transmit and buffer not empty, transmits data.
210          ;   4. If not OK to transmit or buffer empty, disables transmitter.
211
212          XMITINT:
213 016E 9DD2          LD      A, PORTLP
214 0170 6090          ANDSZ  A, #090
215 0172 219C          JMP     IDLE
216 0174 9D1E          LD      A, HEAD
217 0176 BD1F82        IFEQ   A, TAIL
218 0179 219C          JMP     IDLE
219 017B 9CFE          X       A, B
220 017D AA           LD      A, [B+]
221 017E 9C88          X       A, TBUF
222 0180 9DFE          LD      A, B
223 0182 921E          IFEQ   A, #END+1
224 0184 9810          LD      A, #START
225 0186 9C1E          X       A, HEAD
226 0188 9D1E          LD      A, HEAD
227 018A BD1F82        IFEQ   A, TAIL
228 018D 09           JP      NFULL
229 018E A1           SC
230 018F BD1F81        SUBC  A, TAIL
231 0192 89           IFNC
232 0193 940E          ADD     A, #SIZE
233 0195 9303          IFGT  A, #03
234 0197 BDD06D        RBIT   CTS, PORTLD
235 019A 2107          JMP     REST
236 019C 9FBC          LD      B, #ENUI
237 019E 68           RBIT   TIE, [B]
238 019F 2107          JMP     REST
239
240          ;Software Trap
241          ;
242 01A1 B5           SFTINT: RPND
243 01A2 2000          JMP     00
244
245          ;VECTOR INTERRUPT TABLE
246
247          . =01E2
248 01E2 010F          .ADDRW LINT
249          . =01EC
250 01EC 016E          .ADDRW XMITINT
251 01EE 011E          .ADDRW RCVINT
252          . =01FE
253 01FE 01A1          .ADDRW SFTINT
254          .END

```

TL/DD/11110-10

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88

SYMBOL TABLE

B	00FE	BAUD	00BD	BAUDVA	0004	CNTRL	00EE	*
CTS	0005	DCD	0000	DISABL	0044	DUE	0007	
DSR	0006	DTR	0007	END	001D	ENU	00BA	
ENUCMD	0089	ENUI	00BC	ENUICM	0020	ENUR	00BB	
ERROR	0147	ETDX	0005	FE	0006	GIE	0000	
HEAD	001E	ICNTRL	00E8	IDLE	019C	INIT	0008	
LINT	010F	LPEN	0006	MAIN	0002	NFULL	0197	
NOTRDY	011D	OUTERR	0168	PE	0005	PORTLC	00D1	
PORTLD	00D0	PORTLP	00D2	PSR	00BE	PSRVAL	00C8	
PSW	00EF	RBUF	00B9	RCVINT	011E	READY	011A	*
REST	0107	RIE	0001	RTS	0004	RXOFF	0142	*
SFTINT	01A1	SIZE	000E	SP	00FD	START	0010	
TAIL	001F	TBUF	00B8	TIE	0000	WKEDG	00C8	
WKEN	00C9	WKPND	00CA	X	00FC	XMITIN	016E	

TL/DD/11110-11

NATIONAL SEMICONDUCTOR CORPORATION
COP800 CROSS ASSEMBLER, REV:D1, 12 OCT 88

MACRO TABLE

NO WARNING LINES

NO ERROR LINES

267 ROM BYTES USED

SOURCE CHECKSUM = 6884
OBJECT CHECKSUM = 096B

INPUT FILE C:UART.MAC
LISTING FILE C:UART.PRN
OBJECT FILE C:UART.LM

TL/DD/11110-12

Quadrature Signal Interface to a COP400 Microcontroller

National Semiconductor
Application Note 749
Walter Bacharowski



AN-749

INTRODUCTION

Switches have always been a popular way of getting information into a microcontroller. Two-bit quadrature output devices, also known as two-bit gray code output, use two switches that are mechanically coupled together thru a shaft so that as the shaft is rotated the switches generate two square waves that are 90 degrees out of phase with each other. This is also known as being in quadrature, see *Figure 1*. The reason for doing this is that within the two signals there is the information to detect the direction of rotation, i.e., clockwise (CW) or counterclockwise (CCW). This type of device allows an input variable to be increased or decreased by CW or CCW rotation of the shaft. Additionally, these devices allow continuous rotation in either direction, which lets the span and resolution of the input variable to be a function of the software.

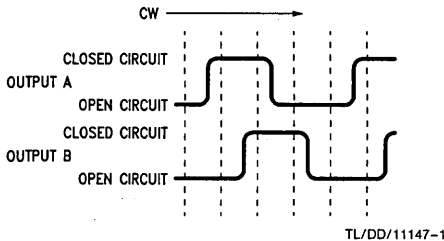


FIGURE 1

OPERATION

Figure 2 shows a hardware connection of a quadrature output device to the COP400 microcontroller. Although in this example the G0 and G1 I/O pins are used, any pin that can be used as an input could be used with the appropriate changes in the software.

In this example the output of device QD1 is processed to detect a state change in the quadrature signal and which direction the change was in. A 3-digit BCD variable, which is stored in RAM, is then incremented or decremented. The variable is defined to have a range of 200 to 350 units. The routine allows the variable to saturate at its upper and lower limits when reached.

Figure 3 displays the two waveforms that are generated by QD1 as its shaft is rotated from an arbitrary starting position. Each edge represents a change of state. By keeping track of the state that was moved from and the state that currently exists, it can be determined which direction the rotation was in.

Referring to *Figure 3*, there are 4 possible states for a starting position, (00, 01, 11, 10), and they will be referred to as the previous state. There are also 4 possible states to move to, (00, 01, 11, 10), and they will be referred to as the current state. *Figure 4* lists the 8 possible combinations of bits that can be formed by starting from each previous state and rotating CW or CCW to the current state. If the two bits of the previous state and current state are concatenated into one 4-bit value, each value will be unique. The routine

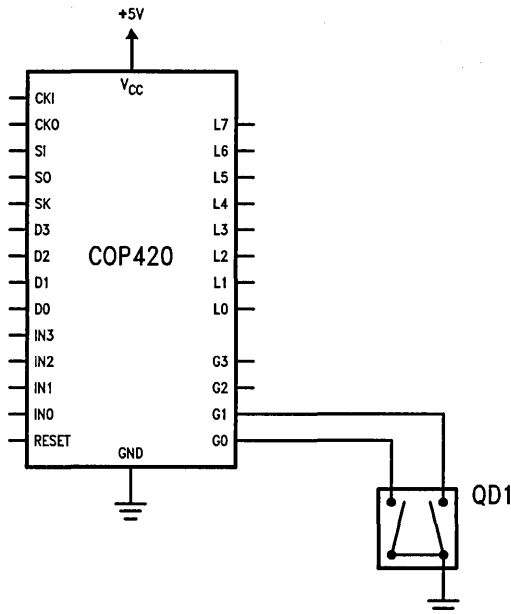
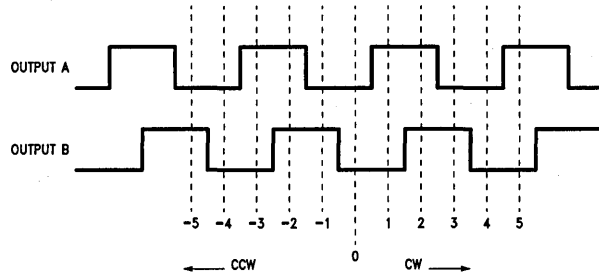


FIGURE 2

POSITION	OUTPUT STATE	
	A	B
-5	0	1
-4	0	0
-3	1	0
-2	1	1
-1	0	1
0	0	0
1	1	0
2	1	1
3	0	1
4	0	0
5	1	0

TL/DD/11147-4



TL/DD/11147-3

FIGURE 3. Quadrature Signal Output (Gray Code)

used in this example assigns the two bits of the current state to the low 2 bits of a 4-bit value, and the 2 bits of the previous state to the high 2 bits of a 4-bit value. This 4-bit value (see the column under the PS/CS heading in Figure 4) is then used as a pointer into a jump table which branches to the add or subtract part of the routine. This method takes advantage of the "jump indirect" instruction which implements a multiway branch based on the value of a pointer. The routine to input data from the quadrature device reads the value of G0 and G1 and compares it to the value stored from the previous read operation. If the two values are equal there is no input to process. If the two values are not equal there is an input and the data is processed to determine if one is to be added to or subtracted from the variable.

The flow chart details the operation of the subroutine "QUAD".

In Figure 4, only 8 of the possible 16 combinations are used. To account for potential spurious operation if one of the 8 undefined combinations occur, they are ignored by this routine by branching to a return instruction which bypasses any additional processing.

The source listing for this example subroutine, which is named "QUAD", is provided. An initialization routine that is required to set up the starting parameters is also included.

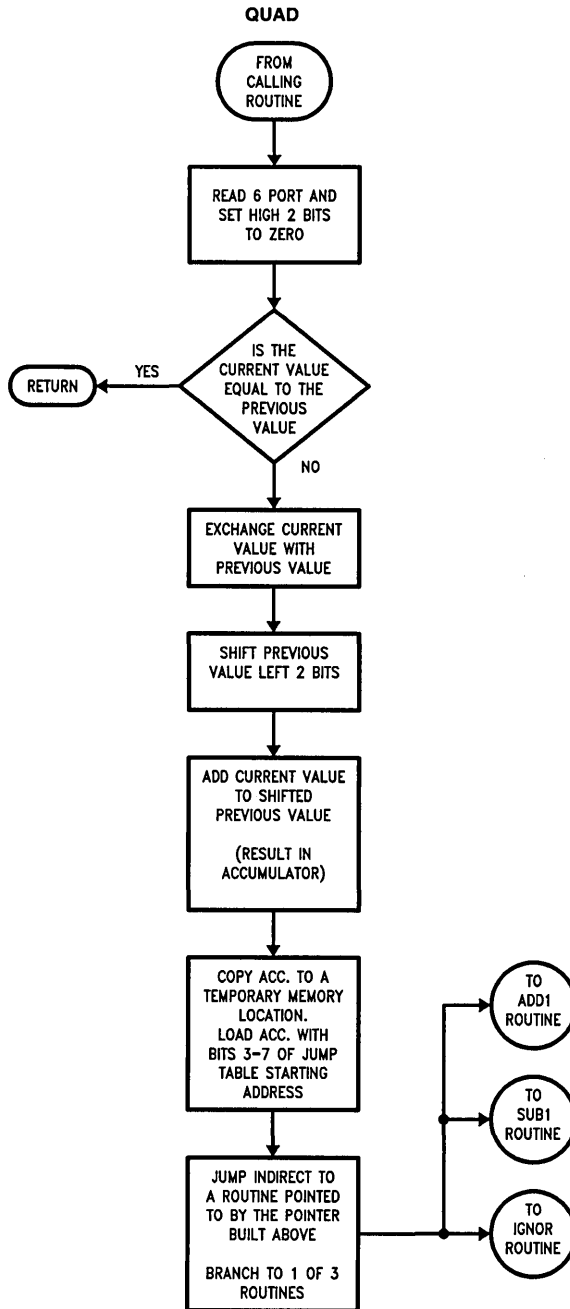
CONCLUSION

This application note demonstrates the relative ease of interfacing a quadrature device to a COP400 microcontroller. The combination of a low cost microcontroller and input device can provide the basis for a cost effective instrument or appliance design.

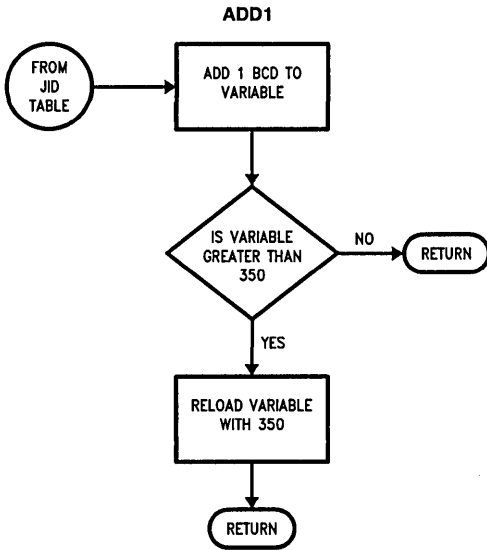
Direction	PS/CS	Hex Value	Operation
CW	00 10	2	Add 1
CCW	00 01	1	Subtract 1
CW	01 00	4	Add 1
CCW	01 11	7	Subtract 1
CW	11 01	D	Add 1
CCW	11 10	E	Subtract 1
CW	10 11	B	Add 1
CCW	10 00	8	Subtract 1

PS = Previous State
CS = Current State

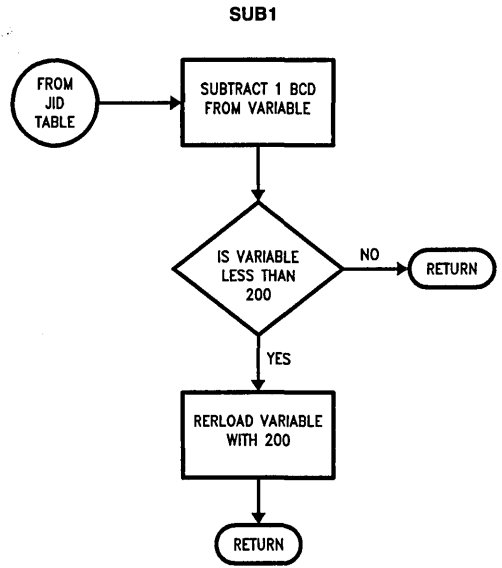
FIGURE 4



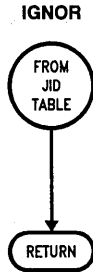
TL/DD/11147-5



TL/DD/11147-6



TL/DD/11147-7



TL/DD/11147-8

NATIONAL SEMICONDUCTOR CORPORATION
 COP400 CROSS ASSEMBLER, REV:D, 8 MAY 85
 QUAD

```

1      ;
2      ;
3      ;           QUAD.MAC
4      ;           QUADRATURE SIGNAL INTERFACE TO THE COP420
5      ;           AUGUST 1, 1990
6      ;           WALTER BACHAROWSKI
7      ;
8      ;           .TITLE QUAD
9      ;           .CHIP 420
10     ;
11     ;*****
12     ;
13     ;ASSIGNMENTS
14     ;
15     ;*****
16     000F      PV      =      0,15      ;PREVIOUS VALUE REGISTER
17     003D      SCRO    =      3,13      ;SCRATCH PAD LOCATION
18     003E      SCR1    =      3,14      ;SCRATCH PAD LOCATION
19     003F      SCR2    =      3,15      ;SCRATCH PAD LOCATION
20     001D      VLD     =      1,13      ;VARIABLE VALUE, LOW DIGIT
21     001E      VMD     =      1,14      ;VARIABLE VALUE, MIDDLE DIGIT
22     001F      VHD     =      1,15      ;VARIABLE VALUE, HIGH DIGIT
23     ;
24     ;*****
25     ;
26     ;PROGRAM START
27     ;
28     ;*****
29     ;
30     POR:
31     0000 00      CLRA
32     0001 335F    OGI      15      ;SET G PORT TO 1's SO THEY CAN BE USED AS INPUTS.
33     0003 0E      LBI      PV      ;GET INITIAL SETTING OF QUADRATURE DEVICE
34     0004 332A    ING
35     0006 06      X
36     0007 42      RMB      2      ;MASK HIGH 2 BITS
37     0008 43      RMB      3
38     ;
39     0009 1C      LBI      VLD     ;LOAD MINIMUM VALUE FOR THE VARIABLE
40     000A 70      STII     0
41     000B 70      STII     0
42     000C 72      STII     2
43     ;
44     IDLE:
45     000D 6A11    JSR      QUAD    ;CONTINUOUS LOOP TO CHECK THE INPUT
46     000F CD      JP        IDLE
47     ;
48     ;
49     0200      . =      X'200    ;FORCE THE QUAD ROUTINE TO START AT HEX 200
50     ;
51     ;

```

TL/DD/11147-9

NATIONAL SEMICONDUCTOR CORPORATION
 COP400 CROSS ASSEMBLER, REV:D, 8 MAY 85
 QUAD

```

52          ;*****
53          ;
54          ;START OF JUMP TABLE FOR PROCESSING INPUTS
55          ;
56          ;*****
57          ;
58          QUADJT:          ;POINTER VALUE IN HEX
59 0200 10          .ADDR  IGNOR  ;0
60 0201 28          .ADDR  ADD1   ;1
61 0202 3D          .ADDR  SUB1   ;2
62 0203 10          .ADDR  IGNOR  ;3
63 0204 3D          .ADDR  SUB1   ;4
64 0205 10          .ADDR  IGNOR  ;5
65 0206 10          .ADDR  IGNOR  ;6
66 0207 28          .ADDR  ADD1   ;7
67 0208 28          .ADDR  ADD1   ;8
68 0209 10          .ADDR  IGNOR  ;9
69 020A 10          .ADDR  IGNOR  ;A
70 020B 3D          .ADDR  SUB1   ;B
71 020C 10          .ADDR  IGNOR  ;C
72 020D 3D          .ADDR  SUB1   ;D
73 020E 28          .ADDR  ADD1   ;E
74 020F 10          .ADDR  IGNOR  ;F          END OF JUMP TABLE
75          ;
76          IGNOR:          ;BYPASS ANY ADDITIONAL PROCESSING
77 0210 48          RET
78          ;
79          ;*****
80          ;
81          ;PROCESS INPUT TO CHECK FOR A CHANGE OF STATE
82          ;
83          ;*****
84          ;
85          QUAD:
86 0211 3E          LBI    SCR2          ;GET CURRENT INPUT STATE
87 0212 332A        ING          ;AND MASK HIGH TWO BITS
88 0214 06          X           ;THEN COMPARE PREVIOUS AND CURRENT
89 0215 42          RMB    2          ;STATE
90 0216 43          RMB    3          ;
91 0217 35          LD     3          ;COPY MASKED VALUE TO ACCUM. AND POINT
92 0218 21          SKE          ;TO PREVIOUS STATE. CHECK IF EQUAL
93 0219 DB          JP    QUAD2
94 021A 48          RET          ;THEY ARE EQUAL SO RETURN
95          ;
96          QUAD2:
97 021B 36          X     3          ;EXCHANGE CURRENT AND PREVIOUS VALUES
98 021C 06          X          ;AND POINT TO SCRATCH LOCATION
99 021D 00          CLRA
100 021E 31         ADD          ;DO A LEFT SHIFT OF 2 BITS
101 021F 31         ADD          ;THIS FORMS THE 2 HIGH BITS OF THE
102 0220 31         ADD          ;JUMP POINTER

```

TL/DD/11147-10

NATIONAL SEMICONDUCTOR CORPORATION
 COP400 CROSS ASSEMBLER, REV:D, 8 MAY 85
 QUAD

```

103 0221 31      ADD
104              ;
105 0222 0E      LBI      PV              ;NOW ADD THE CURRENT STATE THAT WAS JUST
106 0223 31      ADD              ;PROCESSED. BECOMES THE 2 LOW BITS OF
107              ;THE JUMP POINTER
108 0224 3E      LBI      SCR2          ;SET UP THE POINTER FOR THE
109 0225 06      X              ;JUMP INDIRECT POINTER
110 0226 00      CLRA
111              ;
112 0227 FF      JID              ;BRANCH TO REQUIRED ROUTINE
113              ;
114              ;*****
115              ;
116              ;ADD 1 TO THE VALUE OF THE VARIABLE AND CHECK FOR ITS MAX
117              ;VALUE.
118              ;
119              ;*****
120              ;
121
122 0228 1C      ADD1:  LBI      VLD          ;POINT TO LEAST SIGNIFICANT DIGIT
123 0229 22      SC              ;USE CARRY TO ADD 1
124
125 022A 00      ADD1L: CLRA
126 022B 56      AISC      6              ;BCD CORRECTION
127 022C 30      ASC
128 022D 4A      ADT
129 022E 04      XIS              ;STORE DIGIT AND POINT TO NEXT DIGIT
130 022F EA      JP      ADD1L
131 0230 3C      LBI      SCRO          ;STORE VALUE TO CHECK FOR MAX VALUE
132 0231 79      STII     9
133 0232 74      STII     4
134 0233 76      STII     6
135 0234 6A53   JSR      ADDLIM
136 0236 20      SKC              ;IF CARRY IS SET ON RETURN FROM
137 0237 48      RET              ;ADDLIM THEN THE VARIABLE IS LARGER
138 0238 1C      LBI      VLD          ;THEN ITS MAXIMUM VALUE
139 0239 70      STII     0              ;SO RESET IT TO ITS MAX VALUE
140 023A 75      STII     5
141 023B 73      STII     3
142 023C 48      RET
143              ;
144              ;*****
145              ;
146              ;SUBTRACT ONE FROM THE VARIABLE AND CHECK FOR
147              ;IT BEING GREATER THEN THE MINIMIUM VALUE.
148              ;
149              ;*****
150              ;
151
152 023D 1C      SUB1:  LBI      VLD          ;SUBTRACT 1 BY FORCING A BORROW
153 023E 32      RC
  
```

TL/DD/11147-11

NATIONAL SEMICONDUCTOR CORPORATION
 COP400 CROSS ASSEMBLER, REV:D, 8 MAY 85
 QUAD

```

154                                SUB1L:
155 023F 00                        CLRA

156 0240 10                        CASC
157 0241 4A                        ADT                                ;BCD CORRECTION
158 0242 04                        XIS                                ;STORE DIGIT AND POINT TO NEXT DIGIT
159 0243 623F                      JMP                                SUB1L
160                                ;
161 0245 3C                        LBI    SCR0                    ;STORE VALUE TO CHECK LOWER LIMIT OF THE VARIABLE
162 0246 70                        STII   0
163 0247 70                        STII   0
164 0248 78                        STII   8
165 0249 6A53                      JSR    ADDLIM
166                                ;
167 024B 20                        SKC                                ;IF CARRY IS NOT SET ON RETURN THEN VARIABLE
168 024C CE                        JP     SUB2                    ;LESS THEN IT'S MINIMUM VALUE
169 024D 48                        RET
170                                SUB2:
171 024E 1C                        LBI    VLD                    ;FORCE VARIABLE TO ITS MIN VALUE
172 024F 70                        STII   0
173 0250 70                        STII   0
174 0251 72                        STII   2
175 0252 48                        RET
176                                ;
177                                ;*****
178                                ;
179                                ;ADD A VALUE STORED IN SCR0 TO SCR3 TO THE VALUE OF THE
180                                ;VARIABLE NONDESTRUCTIVELY. THE STATE OF THE CARRY BIT
181                                ;IS USED BY THE CALLING ROUTINE AS A RESULT.
182                                ;
183                                ;*****
184                                ;
185                                ADDLIM:
186 0253 1C                        LBI    VLD
187 0254 32                        RC
188                                ADLIM1:
189 0255 25                        LD     2                        ;BCD ADDITION OF 3 DIGITS
190 0256 56                        AISC   6
191 0257 30                        ASC
192 0258 4A                        ADT
193 0259 24                        XIS    2                        ;PROCESS NEXT DIGIT
194 025A D5                        JP     ADLIM1
195 025B 48                        RET
196                                ;
197                                .END

```

TL/DD/11147-12

QUAD
SYMBOL TABLE

ADD1	0228	ADD1L	022A	ADDLIM	0253	ADLIM1	0255
IDLE	000D	IGNOR	0210	POR	0000 *	PV	000F
QUAD	0211	QUAD2	021B	QUADJT	0200 *	SCR0	003D
SCR1	003E *	SCR2	003F	SUB1	023D	SUB1L	023F
SUB2	024E	VHD	001F *	VLD	001D	VMD	001E *

NO ERROR LINES

108 ROM BYTES USED

COP 420 ASSEMBLY

SOURCE CHECKSUM = 2177
OBJECT CHECKSUM = 01FF

INPUT FILE C:QUAD.MAC
LISTING FILE C:QUAD.PRN
OBJECT FILE C:QUAD.LM

TL/DD/11147-13





Section 4
HPC Family



Section 4 Contents

The 16-Bit HPC Family: Optimized for Performance	4-3
HPC16083/HPC26083/HPC36083/HPC46083/HPC16003/HPC26003/HPC36003/ HPC46003 High-Performance Microcontrollers	4-6
HPC36164/HPC46164/HPC36104/HPC46104 High-Performance Microcontrollers with A/D	4-39
HPC16064/HPC26064/HPC36064/HPC46064/HPC16004/HPC26004/HPC36004/ HPC46004 High-Performance Microcontrollers	4-75
HPC36400E/HPC46400E High-Performance Communications Microcontrollers	4-108
HPC167064/HPC467064 High-Performance Microcontrollers with a 16K UV Erasable CMOS EPROM	4-134
HPC46100 High-Performance Microcontroller with DSP Capability	4-165



The 16-Bit HPC™ Family: Optimized for Performance

Key Features

- World's first 16-bit CMOS microcontroller
- World's fastest CMOS microcontroller
- 100 ns for fastest instruction at 40 MHz
- Full 16-bit architecture and implementation
- 64 kbyte address space
- High code efficiency with single-byte, multiple-function instructions
- 16 x 16-bit multiply, 32 x 16-bit divide
- Eight vectored interrupt sources
- Watchdog logic monitors
- 16-bit timer/counters
- Up to 52 general-purpose high-speed I/O lines
- On-chip ROM to 16 kbytes
- On-chip RAM to 1 kbyte
- On-chip peripherals
 - DMA
 - HDLC
 - Timers
 - Input-capture registers
 - UART
 - User-programmable memory
 - High speed SRAM
 - High speed timers
 - A/D
 - DSP
- M²CMOS fabrication
- MICROWIRE/PLUSTM serial interface
- ROMless versions available
- Wide operating voltage range:
+ 4.5V to +5.5V
- Military temp range available
(-55°C to +125°C)
- MIL-STD-883C versions available
- 68-pin PGA, PLCC, LDCC packages and 80-pin PQFP

National's High Performance Controller (HPC) family is not only the world's first 16-bit CMOS microcontroller family, but also the world's fastest.

Currently operating at a clock rate of 40 MHz, the HPC fabricated in scalable M²CMOSTM, allowing die-shrinks ultimately, to submicron levels. Meaning the HPC will be operating at much higher frequencies in the future.

The HPC is designed for high-performance applications. With its 16 x 16-bit multiply and 32 x 16-bit divide, the HPC is appropriate for compute-intensive environments that used to be the sole domain of the microprocessor.

The HPC is ideal, for example, for signal conditioning applications. The HPC's high throughput helps eliminate external components from typical signal processing/control circuits, and allows key parts of the application to be implemented in software rather than hardware.

This not only reduces system cost and development time, but also increases the flexibility and market life of the product.

At the same time, because the HPC has a control-oriented architecture, important functions are still implemented in hardware, providing critical performance advantages unavailable in a pure-software solution, such as a general microprocessor-based design.

It is this powerful performance capability that, when combined with the wide range of peripheral functions that are available (such as UARTs and HDLC), make the HPC a true systems solution on a chip.

The Powerful HPC Core

The HPC is an "application-specific" microcontroller.

Based on a common, high-performance CPU "core", each HPC family member is "customized" to meet the exact needs of a particular application.

The core, based on a microprocessor-like von Neumann architecture, contains seven key functional elements:

1. Arithmetic Logic Unit (ALU)
2. 6 working registers
3. 8 interrupts
4. 3 timers
5. Control logic
6. Watchdog circuitry
7. MICROWIRE/PLUS interface

The internal data paths, registers, timers, and ALU are all 16 bits wide.

So the HPC can directly address up to 64 kbytes of "external" memory.

The external data bus, however, is configurable as 8 or 16 bits, allowing it to efficiently interface with a variety of peripheral devices.

Flexible Peripheral Support

The HPC core can support a full range of peripheral functions:

- High-level Data Link Control (HDLC) for ISO-standard data communications
- Universal Asynchronous Receiver/Transmitters (UARTs) for full-duplex, 300/1200/2400/9600-baud serial communications
- High-Speed Outputs and Pulse-Width Modulated (PWM) timers for efficient external interfaces
- User-programmable memory
- Analog-to-Digital (A/D) converters for interfacing "real-world" inputs
- Multiply/Accumulate Unit for fast signed multiply or multiply-accumulate
Plus:
- Up to 64 kbytes of direct-addressable memory
- Up to 52 I/O ports on a 68-pin package
- Chip select output logic with programmable control

Efficient Instruction Set

The HPC family achieves much of its performance through its unique, highly optimized instruction set. Unlike the instruction set of a typical microprocessor, the HPC instruction set is designed for maximum code efficiency. Because ROM-space is necessarily limited on a single-chip solution, programs must be compact and economical.

The HPC instruction set supports nine addressing modes, like a high-performance 16-bit microprocessor. And each instruction in the set is designed to execute a number of individual functions, so the same operations can be executed with tighter code.

As a result, the typical HPC instruction cycle is only 50 ns at 40 MHz. And the typical HPC 16-bit multiply or divide takes less than 4 μ s.

To achieve the same level of performance in other 16-bit and high-end 8-bit microcontrollers, as indicated by recent benchmark studies, would require up to *two times the memory space* as the HPC.

Low Power Operation

The HPC uses power as efficiently as it uses memory space.

The HPC draws only 47 mA of current at 20 MHz. And its even less at lower clock rates.

In addition, the HPC has two software-selectable power-down modes:

1. IDLE, which stops all operations except for the oscillator and one timer, thereby maintaining all RAM, registers, and I/O in a static state.
2. HALT, which stops all operations including the oscillator and timers, but holds RAM, registers, and I/O stable.

Key Applications

- Signal conditioning/processing/control
- Automotive systems
- Data processing
- Telecommunications
- Hard disk drives
- Military
- Embedded controllers
- Medical
- Factory automation
- Industrial control
- Compute-intensive environments
- High-end control
- Tape and disk drives
- Security systems
- Laser printers
- SCSI control

High Level Language Support

A C compiler and C-source level debugger is available for software development on standard platforms: the IBM PC running DOS Sun system running UNIX.

With powerful tools such as these, the HPC can be quickly and efficiently programmed for any high-performance application.

For further information, see Section 7 on Microcontroller Development Support.

HPC Family

COMMON FEATURES

- M²C MOS Process Technology
- Instruction Cycle Time:
 - 100 ns @ 20 MHz
 - 67 ns @ 30 MHz
 - 50 ns @ 40 MHz for HPC46100
- Idle and Halt Modes

- WATCHDOG
- 64k Address Space
- MICROWIRE/PLUS Serial Interface
- UART
- Minimum of four 16-Bit Timer/Counters with Synchronous Outputs

Part Prefix	COMM Temp.	IND Temp.	AUTO Temp.	MIL Temp.	Memory		Features				Packages			I/O		Development Support			
	0°C to +70°C	-40°C to +85°C	-40°C to +125°C	-55°C to +125°C	ROM (Bytes)	RAM (Bytes)	Interrupt	Stack	Timers	Additional Features	V	U	VF	# of Pins	I/O Pins	Serial I/O	Emulator	Emulator Programmer	Development* System
ROMLESS HPC																			
HPC	46003	36003	26003	16003	0	256	8 Sources	In RAM	8	4 ICRs	x	MIL	x	68/80	52	Yes	N/A	N/A	HPC-DEV-SYS1
HPC	46004	36004	26004	16004	0	512	8 Sources	In RAM	8	4 ICRs	x	MIL	x	68/80	52	Yes	N/A	N/A	HPC-DEV-SYS3
HPC	46100				0	1k	8 Sources	In RAM	7	A/D & MAC/CS		MIL	x	80	31	Yes	N/A	N/A	HPC-DEV-KIT1
ROM'D HPC																			
HPC	46083	36083	26083	16083	8k	256	8 Sources	In RAM	8	4 ICRs	x	MIL	x	68/80	52	Yes	Future (HPC467064)	HPC-DEV-SYS1	HPC-DEV-SYS1
HPC	46064	36064	26064	16064	16k	512	8 Sources	In RAM	8	4 ICRs	x	MIL	x	68/80	52	Yes	Future (HPC467064)	Future (Data I/O)	HPC-DEV-SYS3
HPC	46164	36164	26164	16164	16k	512	8 Sources	In RAM	8	4 ICRs, A/D			x	80	52	Yes	Future (HPC467164)	Future (Data I/O)	
SINGLE CHIP EMULATOR FOR HPC46083, HPC46064 AND HPC46164																			
HPC	467064			167064	16k	512	8 Sources	In RAM	8	4 ICRs	x	MIL	x	68/80	52	Yes	N/A	Future (Data I/O)	
DATA COMMUNICATIONS HPC																			
HPC	46400E	36400E			0	256	8 Sources	In RAM	4	2 Ch HDLC, 4 Ch DMA	x			68	36	Yes	N/A	N/A	HPC-DEV-SYS2

*Hewlett Packard Corporation supports HPC development with the HPC64775

V = Plastic Leaded Chip Carrier (PLCC)

U = Pin Grid Array (PGA)—MIL temperature range product only

VF = Plastic Quad Flat Pack (PQFP)—80 pins

EL = Leaded Chip Carrier-Prototyping package and Military temperature product only



PRELIMINARY

HPC16083/HPC26083/HPC36083/HPC46083/ HPC16003/HPC26003/HPC36003/HPC46003 High-Performance microControllers

General Description

The HPC16083 and HPC16003 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC16083 has 8k bytes of on-chip ROM. The HPC16003 has no on-chip ROM and is intended for use with external direct memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICROWIRE/PLUSTM provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC16083" is used throughout this data-sheet to refer to the HPC16083 and HPC16003 devices unless otherwise specified.

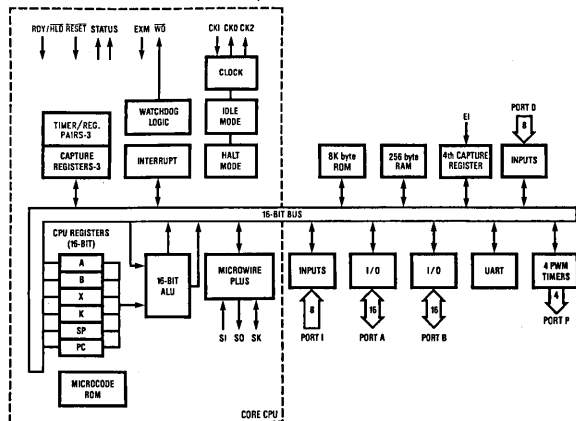
The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LDCC, PGA and 80-Pin PQFP packages.

Features

- HPC family—core features:
 - 16-bit architecture, both byte and word
 - 16-bit data bus, ALU, and registers
 - 64k bytes of external direct memory addressing
 - FAST—200 ns for fastest instruction when using 20.0 MHz clock, 134 ns at 30 MHz
 - High code efficiency—most instructions are single byte
 - 16 x 16 multiply and 32 x 16 divide
 - Eight vectored interrupt sources
 - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
 - MICROWIRE/PLUS serial I/O interface
 - CMOS—very low power with two power save modes: IDLE and HALT
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 8k bytes of ROM, 256 bytes of RAM on chip
- ROMless version available (HPC16003)
- Commercial (0°C to +70°C), industrial (-40°C to +85°C), automotive (-40°C to +105°C) and military (-55°C to +125°C) temperature ranges

For applications requiring more RAM and ROM see HPC16064 data sheet.

Block Diagram (HPC16083 with 8k ROM shown)



TL/DD/8801-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C

V_{CC} with Respect to GND -0.5V to 7.0V
All Other Pins ($V_{CC} + 0.5$)V to (GND - 0.5)V

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46083/HPC46003, -40°C to $+85^\circ\text{C}$ for HPC36083/HPC36003, -40°C to $+105^\circ\text{C}$ for HPC26083/HPC26003, -55°C to $+125^\circ\text{C}$ for HPC16083/HPC16003

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC1}	Supply Current	$V_{CC} = 5.5V, f_{in} = 30\text{ MHz}$ (Note 1)		65	mA
		$V_{CC} = 5.5V, f_{in} = 20\text{ MHz}$ (Note 1)		47	mA
		$V_{CC} = 5.5V, f_{in} = 2.0\text{ MHz}$ (Note 1)		10	mA
I_{CC2}	IDLE Mode Current	$V_{CC} = 5.5V, f_{in} = 30\text{ MHz}$ (Note 1)		5.0	mA
		$V_{CC} = 5.5V, f_{in} = 20\text{ MHz}$, (Note 1)		3.0	mA
		$V_{CC} = 5.5V, f_{in} = 2.0\text{ MHz}$, (Note 1)		1	mA
I_{CC3}	HALT Mode Current	$V_{CC} = 5.5V, f_{in} = 0\text{ kHz}$, (Note 1)		200	μA
		$V_{CC} = 2.5V, f_{in} = 0\text{ kHz}$, (Note 1)		50	μA

INPUT VOLTAGE LEVELS FOR SCHMITT TRIGGERED INPUTS $\overline{\text{RESET}}$, NMI AND $\overline{\text{WO}}$; AND ALSO CKI

V_{IH1}	Logic High		$0.9 V_{CC}$		V
V_{IL1}	Logic Low			$0.1 V_{CC}$	V

ALL OTHER INPUTS

V_{IH2}	Logic High		$0.7 V_{CC}$		V
V_{IL2}	Logic Low			$0.2 V_{CC}$	V
I_{LI1}	Input Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 2	μA
I_{LI2}	Input Leakage Current RDY/HLD, EXUI	$V_{IN} = 0$	-3	-50	μA
I_{LI3}	Input Leakage Current B12	$\overline{\text{RESET}} = 0, V_{IN} = V_{CC}$	0.5	7	mA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF

OUTPUT VOLTAGE LEVELS

V_{OH1}	Logic High (CMOS)	$I_{OH} = -10\ \mu\text{A}$ (Note 2)	$V_{CC} - 0.1$		V
V_{OL1}	Logic Low (CMOS)	$I_{OH} = 10\ \mu\text{A}$ (Note 2)		0.1	V
V_{OH2}	Port A/B Drive, CK2 (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅)	$I_{OH} = -7\text{ mA}$	2.4		V
V_{OL2}		$I_{OL} = 3\text{ mA}$		0.4	V
V_{OH3}	Other Port Pin Drive, $\overline{\text{WO}}$ (open drain) (B ₀ -B ₉ , B ₁₃ , B ₁₄ , P ₀ -P ₃)	$I_{OH} = -1.6\text{ mA}$ (except $\overline{\text{WO}}$)	2.4		V
V_{OL3}		$I_{OL} = 0.5\text{ mA}$		0.4	V
V_{OH4}	ST1 and ST2 Drive	$I_{OH} = -6\text{ mA}$	2.4		V
V_{OL4}		$I_{OL} = 1.6\text{ mA}$		0.4	V
V_{OH5}	Port A/B Drive (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅) when used as External Address/Data Bus	$I_{OH} = -1\text{ mA}$	2.4		V
V_{OL5}		$I_{OL} = 3\text{ mA}$		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE® Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 5	μA

Note 1: I_{CC1} , I_{CC2} , I_{CC3} measured with no external drive (I_{OH} and $I_{OL} = 0$, I_{IH} and $I_{IL} = 0$). I_{CC1} is measured with $\overline{\text{RESET}} = V_{SS}$, I_{CC3} is measured with NMI = V_{CC} , CKI driven to V_{IH1} and V_{IL1} , with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

20 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and *Figure 1* thru *Figure 5*) $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46083/HPC46003, $-40^\circ C$ to $+85^\circ C$ for HPC36083/HPC36003, $-40^\circ C$ to $+105^\circ C$ for HPC26083/HPC26003, $-55^\circ C$ to $+125^\circ C$ for HPC16083/HPC16003

	Symbol and Formula	Parameter	Min	Max	Units	Note
Clocks	f_C	CKI Operating Frequency	2	20	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	50	500	ns	
	t_{CKIH}	CKI High Time	22.5		ns	
	t_{CKIL}	CKI Low Time	22.5		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	100		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	100		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CKI Falling Edge	0	55	ns	(Note 1)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CK1 Falling Edge	0	55	ns	(Note 1)
	$f_U = f_C/8$	External UART Clock Input Frequency		2.5**	MHz	
	f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz	
Timers	$f_{XIN} = f_C/22$	External Timer Input Frequency		0.91	MHz	
	$t_{XIN} = t_C$	Pulse Width for Timer Inputs	100		ns	
MICROWIRE/ PLUS	t_{UWS}	MICROWIRE Setup Time—Master —Slave	100 20		ns	
	t_{UWH}	MICROWIRE Hold Time—Master —Slave	20 50		ns	
	t_{UWV}	MICROWIRE Output Valid Time—Master —Slave		50 150	ns	
External Hold	$t_{SALE} = \frac{3}{4} t_C + 40$	\overline{HLD} Falling Edge before ALE Rising Edge	115		ns	
	$t_{HWP} = t_C + 10$	\overline{HLD} Pulse Width	110		ns	
	$t_{HAE} = t_C + 100$	\overline{HLDA} Falling Edge after \overline{HLD} Falling Edge		200	ns	(Note 3)
	$t_{HAD} = \frac{3}{4} t_C + 85$	\overline{HLDA} Rising Edge after \overline{HLD} Rising Edge		160	ns	
	$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after \overline{HLDA} Falling Edge		116	ns	(Note 5)
	$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable after \overline{HLDA} Rising Edge	116		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	(Note 6)
	t_{DRDY}	\overline{RDRDY} Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH}	Input Data Hold after Rising Edge of \overline{UWR}	20		ns	
	t_A	\overline{WRRDY} Delay from Rising Edge of \overline{UWR}		70	ns	

**This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

20 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 thru Figure 5) $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46083/HPC46003, $-40^\circ C$ to $+85^\circ C$ for HPC36083/HPC36003, $-40^\circ C$ to $+105^\circ C$ for HPC26083/HPC26003, $-55^\circ C$ to $+125^\circ C$ for HPC16083/HPC16003 (Continued)

	Symbol and Formula	Parameter	Min	Max	Units	Note
Address Cycles	$t_{DC1ALER}$	Delay from CK1 Rising Edge to ALE Rising Edge	0	35	ns	(Note 1)
	$t_{DC1ALEF}$	Delay from CK1 Rising Edge to ALE Falling Edge	0	35	ns	(Note 1)
	$t_{DC2ALER} = \frac{1}{4}t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		45	ns	
	$t_{DC2ALEF} = \frac{1}{4}t_C + 20$	Delay from CK2 Rising Edge to ALE Falling Edge		45	ns	(Note 2)
	$t_{LL} = \frac{1}{2}t_C - 9$	ALE Pulse Width	41		ns	
	$t_{ST} = \frac{1}{4}t_C - 7$	Setup of Address Valid before ALE Falling Edge	18		ns	
	$t_{HP} = \frac{1}{4}t_C - 5$	Hold of Address Valid after ALE Falling Edge	20		ns	
Read Cycles	$t_{ARR} = \frac{1}{4}t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	20		ns	
	$t_{ACC} = t_C + WS - 55$	Data Input Valid after Address Output Valid		145	ns	(Note 2)
	$t_{RD} = \frac{1}{2}t_C + WS - 65$	Data Input Valid after \overline{RD} Falling Edge		95	ns	
	$t_{RW} = \frac{1}{2}t_C + WS - 10$	\overline{RD} Pulse Width	140		ns	
	$t_{DR} = \frac{3}{4}t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	60	ns	
	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	85		ns	
Write Cycles	$t_{ARW} = \frac{1}{2}t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	45		ns	
	$t_{WW} = \frac{3}{4}t_C + WS - 15$	\overline{WR} Pulse Width	160		ns	
	$t_V = \frac{1}{2}t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	145		ns	
	$t_{HW} = \frac{1}{4}t_C - 5$	Hold of Data Valid after \overline{WR} Rising Edge	20		ns	
Ready Input	$t_{DAR} = \frac{1}{4}t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns	
	$t_{RWP} = t_C$	RDY Pulse Width	100		ns	

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CK1 having 50% duty cycle and with less than 15 pF load on CKO with rise and fall times (t_{CK1R} and t_{CK1F}) on CK1 input less than 2.5 ns.

Note 2: Do not design with these parameters unless CK1 is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CK1 or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{HLD} falling edge occurring at the latest time it can be accepted during the present CPU cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} as long as $(3t_C + 4WS + 72t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait state and ready input.

Note 4: WS (t_{WAIT}) x (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 20$ MHz, with one wait programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

Note 6: This is guaranteed by design and not tested.

30 MHz

AC Electrical Characteristics (Continued)

(See Notes 1 and 4 and Figure 1 thru Figure 5) $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46083/HPC46003, $-40^\circ C$ to $+85^\circ C$ for HPC36083/HPC36003, $-40^\circ C$ to $+105^\circ C$ for HPC26083/HPC26003, $-55^\circ C$ to $+125^\circ C$ for HPC16083/HPC16003

	Symbol and Formula	Parameter	Min	Max	Units	Note
Clocks	f_C	CKI Operating Frequency	2	30	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	33	500	ns	
	t_{CKIH}	CKI High Time	15		ns	
	t_{CKIL}	CKI Low Time	16.6		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	66		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	66		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CK1 Falling Edge	0	55	ns	(Note 1)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CK1 Falling Edge	0	55	ns	(Note 1)
	$f_U = f_C/8$	External UART Clock Input Frequency		3.75**	MHz	
	f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.875	MHz	
Timers	$f_{XIN} = f_C/22$	External Timer Input Frequency		1.364	MHz	
	$t_{XIN} = t_C$	Pulse Width for Timer Inputs	66		ns	
MICROWIRE/ PLUS	t_{UWS}	MICROWIRE Setup Time—Master —Slave	100 20		ns	
	t_{UWH}	MICROWIRE Hold Time—Master —Slave	20 50		ns	
	t_{UWV}	MICROWIRE Output Valid Time—Master —Slave		50 150	ns	
External Hold	$t_{SALE} = 3/4 t_C + 40$	HLD Falling Edge before ALE Rising Edge	90		ns	
	$t_{HWP} = t_C + 10$	HLD Pulse Width	76		ns	
	$t_{HAE} = t_C + 85$	HLDA Falling Edge after HLD Falling Edge		151	ns	(Note 3)
	$t_{HAD} = 3/4 t_C + 85$	HLDA Rising Edge after HLD Rising Edge		135	ns	
	$t_{BF} = 1/2 t_C + 66$	Bus Float after HLDA Falling Edge		99	ns	(Note 5)
	$t_{BE} = 1/2 t_C + 66$	Bus Enable after HLDA Rising Edge	99		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	(Note 3)
	t_{DRDY}	RDRDY Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH}	Input Data Hold after Rising Edge of \overline{UWR}	15		ns	
	t_A	WRRDY Delay from Rising Edge of \overline{UWR}		70	ns	

**This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

30 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 thru Figure 5) $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46083/HPC46003, $-40^\circ C$ to $+85^\circ C$ for HPC36083/HPC36003, $-40^\circ C$ to $+105^\circ C$ for HPC26083/HPC26003, $-55^\circ C$ to $+125^\circ C$ for HPC16083/HPC16003 (Continued)

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Address Cycles	$t_{DC1ALER}$	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns	(Notes 1, 2)
	$t_{DC1ALEF}$	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns	(Notes 1, 2)
	$t_{DC2ALER} = \frac{1}{4} t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		37	ns	(Note 2)
	$t_{DC2ALEF} = \frac{1}{4} t_C + 20$	Delay from CK2 Falling Edge to ALE Falling Edge		37	ns	(Note 2)
	$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	24		ns	
	$t_{ST} = \frac{1}{4} t_C - 7$	Setup of Address Valid before ALE Falling Edge	9		ns	
Read Cycles	$t_{VP} = \frac{1}{4} t_C - 5$	Hold of Address Valid after ALE Falling Edge	11		ns	
	$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	12		ns	
	$t_{ACC} = t_C + WS - 32$	Data Input Valid after Address Output Valid		100	ns	(Note 2)
	$t_{RD} = \frac{1}{2} t_C + WS - 39$	Data Input Valid after \overline{RD} Falling Edge		60	ns	
	$t_{RW} = \frac{1}{2} t_C + WS - 14$	\overline{RD} Pulse Width	85		ns	
	$t_{DR} = \frac{3}{4} t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	35	ns	
Write Cycles	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	51		ns	
	$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	28		ns	
	$t_{WW} = \frac{3}{4} t_C + WS - 15$	\overline{WR} Pulse Width	101		ns	
	$t_V = \frac{1}{2} t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	94		ns	
Ready Input	$t_{HW} = \frac{1}{4} t_C - 10$	Hold of Data Valid after \overline{WR} Rising Edge	7		ns	
	$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		33	ns	
	$t_{RWP} = t_C$	RDY Pulse Width	66		ns	

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CKI having 50% duty cycle and with less than 15 pF load on CKO with rise and fall times (t_{CKIR} and t_{CKIL}) on CKI input less than 2.5 ns.

Note 2: Do not design with these parameters unless CKI is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CKI or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{HLD} falling edge occurring at the latest time it can be accepted during the present CPU cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} as long as $(3t_C + 4WS + 72t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: $WS \times t_{WAIT}$ (number of pre-programmed wait states). Minimum and maximum values are calculated from maximum operating frequency, $t_C = 30$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

Note 6: This is guaranteed by design and not tested.

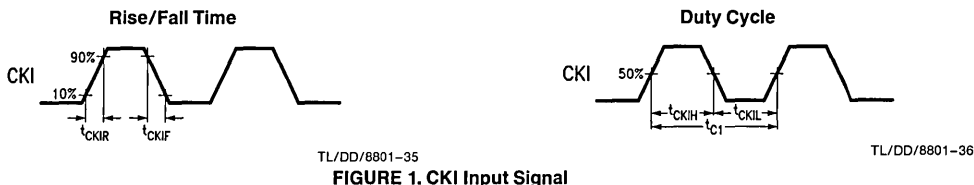


FIGURE 1. CKI Input Signal

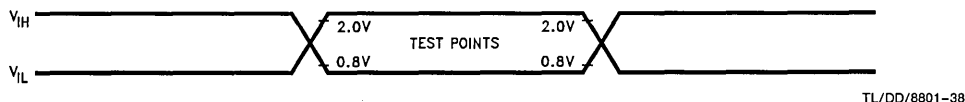


FIGURE 2. Input and Output for AC Tests

Timing Waveforms

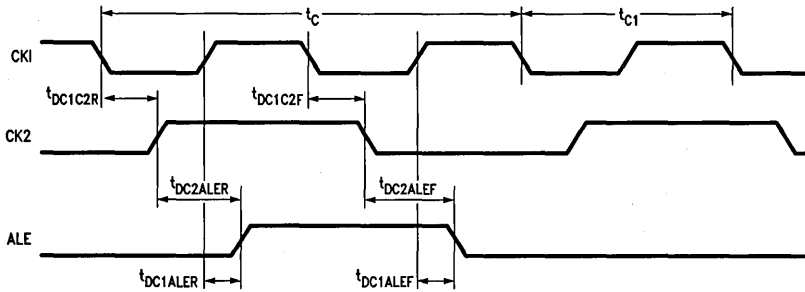


FIGURE 3. CK1, CK2, ALE Timing Diagram

TL/DD/8801-33

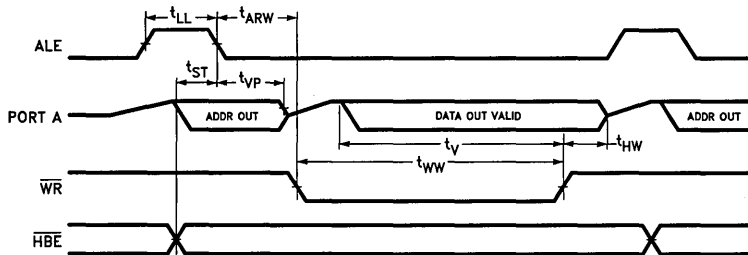


FIGURE 4. Write Cycle

TL/DD/8801-3

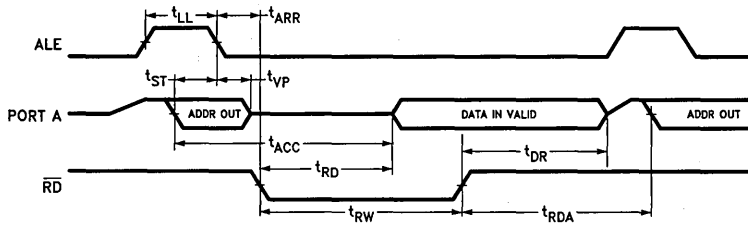


FIGURE 5. Read Cycle

TL/DD/8801-4

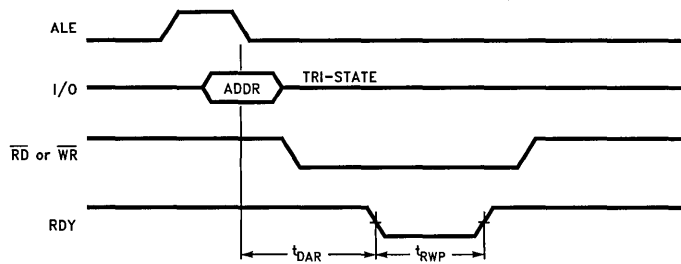


FIGURE 6. Ready Mode Timing

TL/DD/8801-5

Timing Waveforms (Continued)

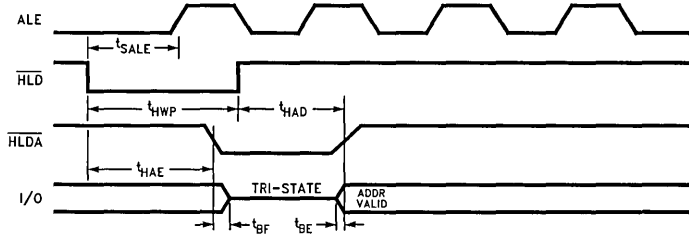


FIGURE 7. Hold Mode Timing

TL/DD/8801-6

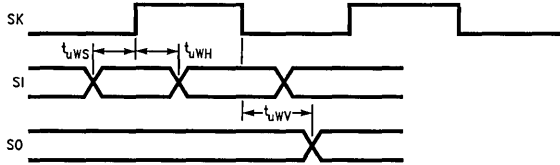


FIGURE 8. MICROWIRE Setup/Hold Timing

TL/DD/8801-37

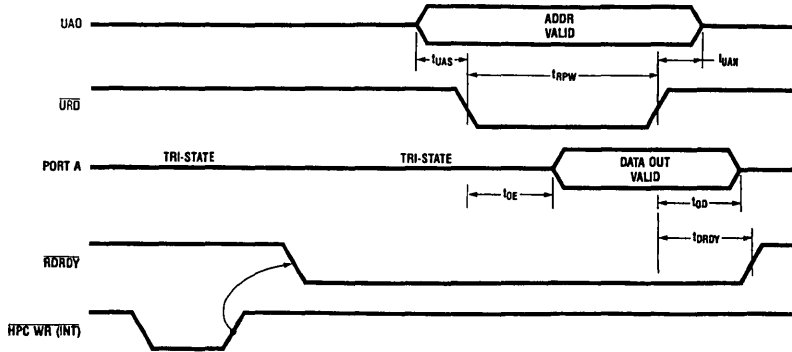


FIGURE 9. UPI Read Timing

TL/DD/8801-9

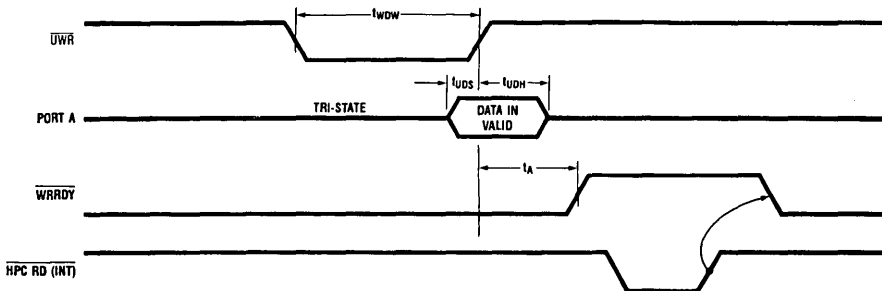


FIGURE 10. UPI Write Timing

TL/DD/8801-10

The following is the Military 883 Electrical Specification for HPC16083 and HPC16003. For latest information on RETS 16083X contact NSC local sales office.

DC Electrical Specifications Test Conditions $V_{CC} = 5V \pm 10\%$ (Unless Otherwise Specified) (Note 1)

Symbol	Parameter	Conditions	SBGRP 1 + 25°C		SBGRP 2 + 125°C		SBGRP 3 - 55°C		Units	Notes
			Min	Max	Min	Max	Min	Max		
V_{IH1}	Logical "1" Input Voltage	RESET, NMI, CKI and \overline{WO}	0.9		0.9		0.9		V	
V_{IH2}		$B_{10}-B_{13}, B_{15}$	(V_{CC})		(V_{CC})		(V_{CC})		V	
V_{IH3}		All Inputs except Port A	0.7		0.7		0.7		V	
V_{IH3}		Port A, $V_{CC} = 5.5V$	4.65		4.65		4.65		V	(Note 2)
V_{IH3}		Port A, $V_{CC} = 4.5V$	3.95		3.95		3.95		V	(Note 2)
V_{IL1}	Logical "0" Input Voltage	RESET, NMI, CKI and \overline{WO}		0.1		0.1		0.1	V	
V_{IL2}		All Inputs except Port A		(V_{CC})		(V_{CC})		(V_{CC})	V	
V_{IL3}		Port A, $V_{CC} = 5.5V$		0.2		0.2		0.2	V	
V_{IL3}		Port A, $V_{CC} = 4.5V$		(V_{CC})		(V_{CC})		(V_{CC})	V	(Note 3)
V_{IL3}		Port A, $V_{CC} = 4.5V$		0.7		0.7		0.7	V	(Note 3)
V_{IL3}		Port A, $V_{CC} = 4.5V$		0.5		0.5		0.5	V	(Note 3)
V_{OH2}	Logical "1" Output Voltage	$I_{OH} = -7$ mA ($A_0-A_{15}, B_{10}-B_{12}, B_{15}, CK2$)	2.4		2.4		2.4		V	
V_{OH3}		$I_{OH3} = -1.6$ mA ($B_0-B_9, B_{13}-B_{14}, P_0-P_3$), \overline{WO} (Open Drain)	2.4		2.4		2.4		V	
V_{OH4}		$I_{OH} = -6$ mA (ST1, ST2)	2.4		2.4		2.4		V	
V_{OH5}		$I_{OH} = -1$ mA ($A_0-A_{15}, B_{10}-B_{12}, B_{15}$)	2.4		2.4		2.4		V	
V_{OH5}		When Used as an External Address/Data Bus	2.4		2.4		2.4		V	
V_{OL2}	Logical "0" Output Voltage	$I_{OL} = 3$ mA (CK2, $A_0-A_{15}, B_{10}-B_{12}, B_{15}$)		0.4		0.4		0.4	V	
V_{OL3}		$I_{OL} = 0.5$ mA ($B_0-B_9, B_{13}-B_{14}, P_0-P_3$)		0.4		0.4		0.4	V	
V_{OL4}		\overline{WO} (Open Drain)		0.4		0.4		0.4	V	
V_{OL5}		$I_{OL} = 1.6$ mA (ST1, ST2)		0.4		0.4		0.4	V	
V_{OL5}		$I_{OL} = 3$ mA ($A_0-A_{15}, B_{10}-B_{12}, B_{15}$)		0.4		0.4		0.4	V	
V_{OL5}	When Used as an External Address/Data Bus		0.4		0.4		0.4	V		
I_{OZ}	TRI-STATE Leakage	$V_{SS} \leq V_{IN} \leq V_{CC}$ (\overline{WO} , Port A, Port B), $V_{CC} = 5.5V$		± 5		± 5		± 5	μA	
I_{L1}	Input Leakage Current	$V_{SS} \leq V_{IN} \leq V_{CC}$, $V_{CC} = 5.5V$ ($I_1-I_6, D_0-D_7, CK1, RESET, EXM, EI$)		± 2		± 2		± 2	μA	(Note 7)
I_{LI2}	Input Pullup Current	$V_{IN} = 0$ ($I_0, I_7, RDY/HLD, EXUI$), $V_{CC} = 5.5V$	-50	-3	-50	-3	-50	-3	μA	(Note 7)
I_{LI3}	Port B_{12} Pulldown during Reset	$V_{IN} = V_{CC}$, Port B_{12} , $V_{CC} = 5.5V$	1	7	1	7	1	7	mA	
VRAM	RAM Keep Alive Voltage	Test Duration is 10 ms	2.5		2.5		2.5		V	
I_{CC1}	Supply Current Dynamic	$F_{IN} = 20$ MHz, $\overline{RESET} = V_{SS}$, $I_{OH} = 0$ mA, $I_{OL} = 0$ mA, $V_{CC} = 5.5V$		55		55		55	mA	
I_{CC2}	Idle Mode Current	$F_{IN} = 20$ MHz, External Clock		3.5		3.5		3.5	mA	
I_{CC}	Halt Mode Current	$NMI = V_{CC}$		2		2		2	mA	
CI/O	Input/Output Capacitance	$f_{test} = 1.0$ MHz, I/O Pin to Ground		20					pF	(Note 4)
CI	Input Capacitance	$f_{test} = 1.0$ MHz, Input Pin to Ground	SBGRP4							
				10						pF

Note 1: Electrical end point testing (when required) for Groups C & D shall consist only of subgroups 1, 2, 9 and 10.

Note 2: Port A V_{IH} test limit includes 700 mV offset caused by output loads being on during Data Drive Time.

Note 3: Port A V_{IL} test limit includes 400 mV offset caused by output loads being on during Data Drive Time.

Note 4: Verified at initial qual only.

Note 7: Future revisions of this device will not have pullups on pins I_0, I_7 which will be tested to I_{L1} conditions.

AC Electrical Specifications Test Conditions $V_{CC} = 4.5V$ and $5.5V$ (Unless Otherwise Specified) (Note 1)

Symbol	Parameter	Conditions	SBGRP 9 + 25°C		SBGRP 10 + 125°C		SBGRP 11 - 55°C		Units	Notes
			Min	Max	Min	Max	Min	Max		
$f_C =$ CKI Freq.	Operating Frequency		2	20	2	20	2	20	MHz	(Note 5)
$t_{CI} = 1/FC$	Clock Period		50		50		50		ns	(Note 5)
$t_C = 2/FC$	Timing Cycle		100		100		100		ns	(Note 5)
$t_{LL} = \frac{1}{2}t_C - 9$	ALE Pulse Width		41		41		41		ns	(Note 6)
$t_{ST} = \frac{1}{4}t_C - 7$	Address Valid to ALE Falling Edge		18		18		18		ns	(Note 6)
$t_{WAIT} = t_C = WS$	Wait State Period		100		100		100		ns	(Note 5)
$FMW = 0.0625 f_C$	External MICROWIRE/PLUS CLK Input Frequency			1.25		1.25		1.25	MHz	(Note 6)
$f_U = 0.125 f_C$	External UART Clock Input Frequency			2.5		2.5		2.5	MHz	(Note 5)
t_{DCIC2}	CK2 Delay From CK1			55		55		55	ns	(Note 6)
$t_{ARR} = \frac{1}{4}t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge		20		20		20		ns	(Note 6)
$t_{RW} = \frac{1}{2}t_C + WS - 10$	\overline{RD} Pulse Width		140		140		140		ns	(Note 6)
$t_{DR} = 3.4 t_C - 15$	Data Hold after Rising Edge of \overline{RD}		0	60	0	60	0	60	ns	(Note 6)
$t_{RD} = \frac{1}{2}t_C + WS - 65$	\overline{RD} Falling Edge to Data in Valid			85		85		85	ns	(Note 6)
$t_{RDA} = t_C - 15$	\overline{RD} Rising Edge to Address Valid		85		85		85		ns	(Note 6)
$t_{VP} = \frac{1}{4}t_C - 5$	Address Hold from ALE Falling Edge		20		20		20		ns	(Note 6)
$t_{ARW} = \frac{1}{2}t_C - 5$	ALE Trailing Edge to \overline{WR} Falling Edge		45		45		45		ns	(Note 6)
$t_{WW} = \frac{3}{4}t_C + WS - 15$	\overline{WR} Pulse Width		160		160		160		ns	(Note 6)
$t_{HW} = \frac{1}{4}t_C - 5$	Data Hold after Trailing Edge of \overline{WR}		20		20		20		ns	(Note 6)
$t_V = \frac{1}{2}t_C + WS - 5$	Data Valid before Rising Edge of \overline{WR}		145		145		145		ns	(Note 6)
$t_{DAR} = \frac{1}{4}t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY			75		75		75	ns	(Note 6)

AC Electrical Specifications

Test Conditions $V_{CC} = 4.5V$ and $5.5V$ (Unless Otherwise Specified) (Note 1)

(Continued)

Symbol	Parameter	Conditions	SBGRP 9 + 25°C		SBGRP 10 + 125°C		SBGRP 11 - 55°C		Units	Notes
			Min	Max	Min	Max	Min	Max		
$t_{RWP} = t_C$	RDY Pulse Width		100		100		100		ns	(Note 6)
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of \overline{HLD} to to Rising Edge of ALE		115		115		115		ns	(Note 6)
$t_{HWP} = t_C + 10$	\overline{HLD} Pulse Width		110		110		110		ns	(Note 6)
$t_{HAD} = \frac{3}{4} t_C + 85$	Rising Edge on \overline{HLD} to Rising Edge on \overline{HLDA}			160		160		160	ns	(Note 6)
$t_{HAE} = t_C + 100$	Falling Edge on \overline{HLD} to Falling Edge on \overline{HLDA}			200		200		200	ns	(Note 6)
$t_{BF} = \frac{1}{2} t_C + 66$	BUS Float before Falling Edge on \overline{HLDA}			116		116		116	ns	(Note 6)
$t_{BE} = \frac{1}{2} t_C + 66$	BUS Enable from Rising Edge of \overline{HLDA}		116		116		116		ns	(Note 6)
t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}		10		10		10		ns	(Note 6)
t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}		10		10		10		ns	(Note 6)
t_{RPW}	\overline{URD} Pulse Width		100		100		100		ns	(Note 6)
t_{OE}	\overline{URD} Falling Edge to Data Out Valid			60		60		60	ns	(Note 6)
t_{RDRDY}	\overline{RDY} Delay from Rising Edge of \overline{URD}			70		70		70	ns	(Note 6)
t_{WDW}	\overline{UWR} Pulse Width		40		40		40		ns	(Note 6)
t_{UDS}	Data Invalid before Trailing Edge of \overline{UWR}		10		10		10		ns	(Note 6)
t_{UDH}	Data In Hold after Rising Edge of \overline{UWR}		15		15		15		ns	(Note 6)
t_A	\overline{WRRDY} Delay from Rising Edge of \overline{UWR}			70		70		70	ns	(Note 6)

Note 1: Electrical end point testing (when required) for groups C & D shall consist only of subgroups 1, 2, 9 and 10.

Note 5: Tested in functional patterns. Not directly measured.

Note 6: $C_L = 70$ pF. Input and output levels are per DC characteristics.

Pin Descriptions

The HPC16083 is available in 68-pin PLCC, LDCC, PGA, and 80-pin PQFP packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	\overline{HLDA}	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	\overline{WRRDY}	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output

Pin Descriptions (Continued)

- B14: TS3 Timer Synchronous Output
 B15: $\overline{RD\overline{RDY}}$ Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

- B10: ALE Address Latch Enable Output
 B11: \overline{WR} Write Output
 B12: \overline{HBE} High Byte Enable Output/Input (sampled at reset)
 B15: \overline{RD} Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

- I0:
 I1: NMI Nonmaskable Interrupt Input
 I2: INT2 Maskable Interrupt/Input Capture/ \overline{URD}
 I3: INT3 Maskable Interrupt/Input Capture/ \overline{UWR}
 I4: INT4 Maskable Interrupt/Input Capture
 I5: SI MICROWIRE/PLUS Data Input
 I6: RDX UART Data Input
 I7:

Port D is an 8-bit input port that can be used as general purpose digital inputs.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

- V_{CC1} and V_{CC2} Positive Power Supply
 GND Ground for On-Chip Logic
 DGND Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

- CKI The Chip System Clock Input
 CKO The Chip System Clock Output (inversion of CKI)
 Pins CKI and CKO are usually connected across an external crystal.

- CK2 Clock Output (CKI divided by 2)

OTHER PINS

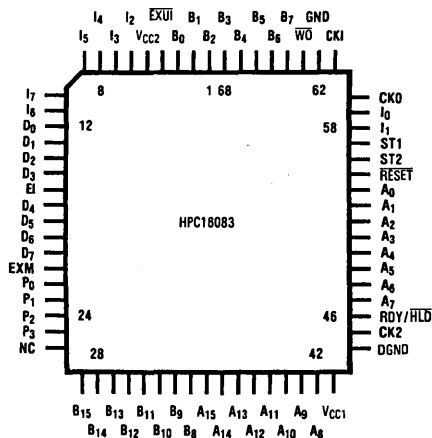
- \overline{WO} This is an active low open drain output that signals an illegal situation has been detected by the Watch Dog logic.
 ST1 Bus Cycle Status Output: indicates first opcode fetch.
 ST2 Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
 \overline{RESET} is an active low input that forces the chip to start and sets the ports in a TRI-STATE mode.

$\overline{RDY}/\overline{HLD}$ has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.

- NC (no connection) do not connect anything to this pin.
 EXM External memory enable (active high) disables internal ROM and maps it to external memory.
 EI External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.
 \overline{EXUI} External interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 (Active Low).

Connection Diagrams

Plastic and Ceramic Leaded Chip Carriers



TL/DD/8801-11

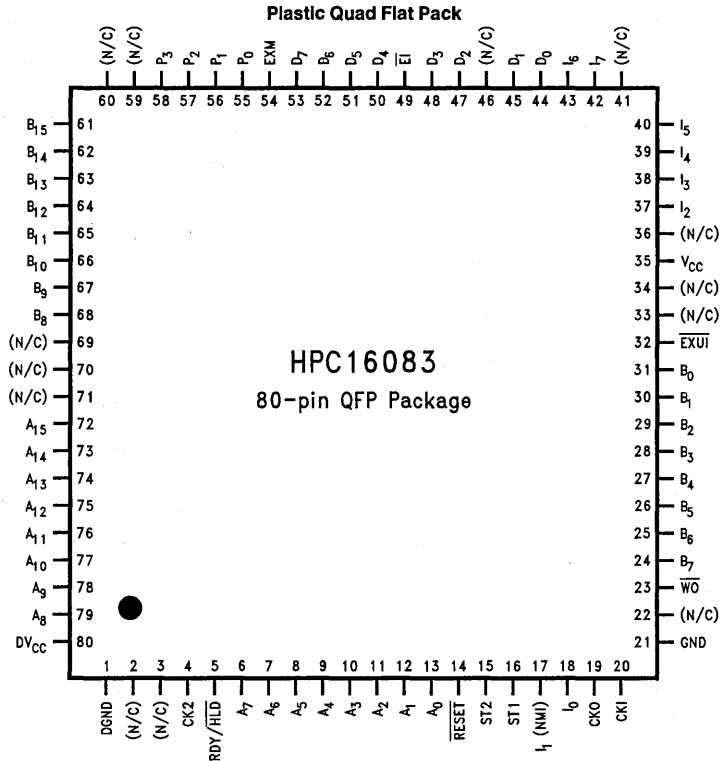
Top View

Order Number HPC16083XXX/L20, HPC16083XXX/L30,
 HPC16003EL20, or HPC16003EL30
 See NS Package Number EL68A

Order Number HPC16083XXX/V20, HPC26083XXX/V20,
 HPC36083XXX/V20, HPC46083XXX/V20,
 HPC16083XXX/V30, HPC26083XXX/V30,
 HPC36083XXX/V30, HPC16003V20, HPC26003V20,
 HPC36003V20, HPC46003V20, HPC16003V30,
 HPV26003V30, HPC36003V30 or HPC46003V30
 See NS Package Number V68A

Note: XXX designates the unique ROM code of a masked device.

Connection Diagrams (Continued)

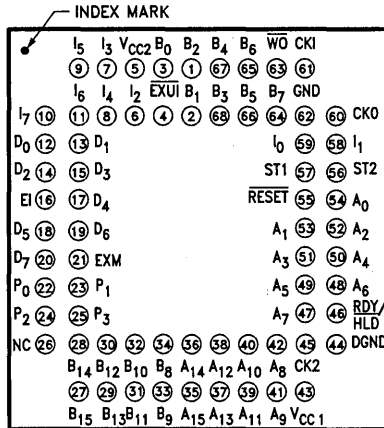


Top View

TL/DD/8801-34

Order Number HPC46083XXX/F20, HPC46083XXX/F30, HPC46003VF20 or HPC46003VF30
See NS Package Number VF80B

Pin Grid Array Pinout



Top View

TL/DD/8801-12

(looking down on component side of PC Board)

Order Number HPC16083XXX/U20, HPC16083XXX/U30, HPC16003U20 or HPC16003U30
See NS Package Number U68A

Note: XXX designates the unique ROM code of a masked device.

Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 11*), consists of a data register and a direction register. Port B (see *Figures 12, 13, 14*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.

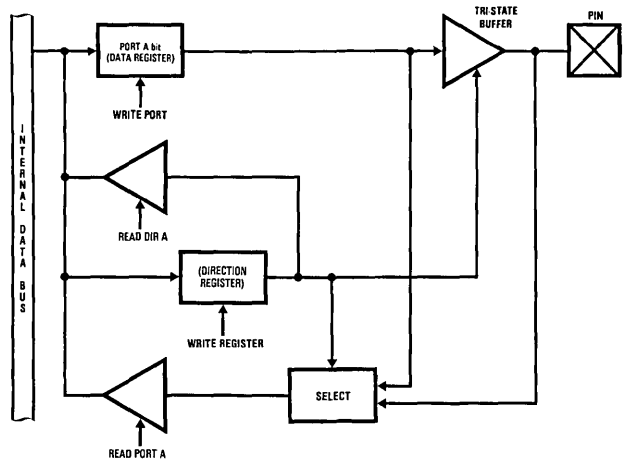


FIGURE 11. Port A: I/O Structure

TL/DD/8801-13

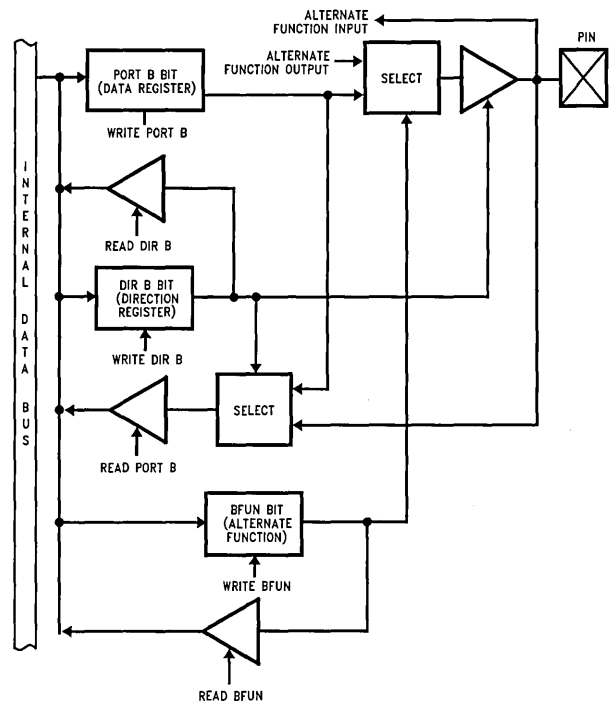
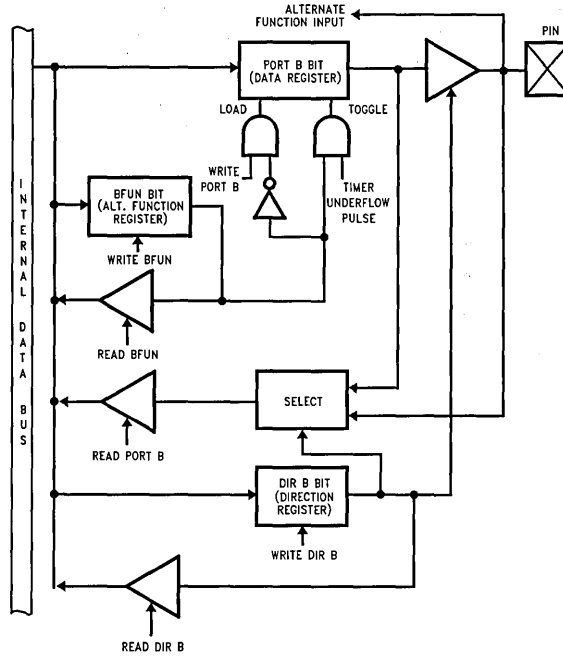


FIGURE 12. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

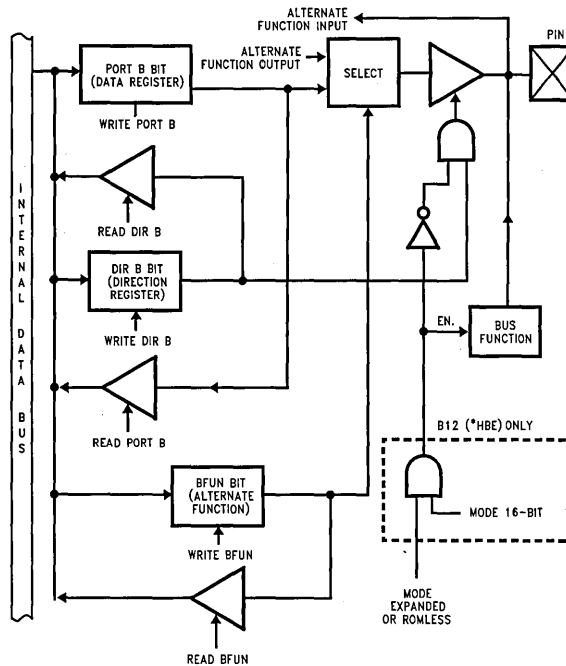
TL/DD/8801-14

Ports A & B (Continued)



TL/DD/8801-15

FIGURE 13. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)



TL/DD/8801-16

FIGURE 14. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC16083 has four operating modes. The ROMless HPC16003 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC16083 is E000 to FFFF (8k bytes). The HPC16003 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC16003 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range and on-chip RAM and Register range, and the "illegal address detection" feature of the WATCHDOG logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the WATCHDOG logic is disabled. The EA bit should be set to "1" by software when using the HPC16003 to disable the "illegal address detection" feature of WATCHDOG.

All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, \overline{RD} , \overline{WR} and HBE. The High Byte Enable pin (\overline{HBE}) is used in 16-bit mode to select high order memory bytes. The \overline{RD} and \overline{WR} signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling \overline{HBE} high at reset. If \overline{HBE} is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC16083 and HPC16003.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC16083 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC16083 functions as a self-contained microcomputer (see *Figure 15*) with all memory (RAM and

ROM) on-chip. It can address internal memory only, consisting of 8k bytes of ROM (E000 to FFFF) and 256 bytes of on-chip RAM and registers (0000 to 01FF). The "illegal address detection" feature of the WATCHDOG is enabled in the Single-Chip Normal mode and a WATCHDOG Output (\overline{WO}) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC16083 to address external memory in addition to the on-chip ROM and RAM (see Table II). WATCHDOG illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC16083 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 8k bytes of external memory may be used with the HPC16083 (see Table II). The WATCHDOG circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

EXPANDED ROMLESS MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of WATCHDOG is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

TABLE II. HPC16083 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	E000:FFFF on-chip
Expanded Normal	0	1	E000:FFFF on-chip 0200:DFFF off-chip
Single-Chip ROMless	1	0	E000:FFFF off-chip
Expanded ROMless	1	1	0200:FFFF off-chip

Note: In all operating modes, the on-chip RAM and Registers (0000:01FF) may be accessed.

HPC16003 Operating Modes

EXPANDED ROMLESS MODE (HPC16003)

Because the HPC16003 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC16003 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between F000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the WATCHDOG logic.

TABLE III. HPC16003 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0200:FFFF off-chip

Note: The on-chip RAM and Registers (0000:01FF) of the HPC16003 may be accessed at all times.

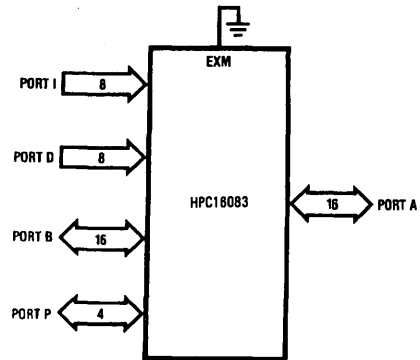


FIGURE 15. Single-Chip Mode

TL/DD/8801-17

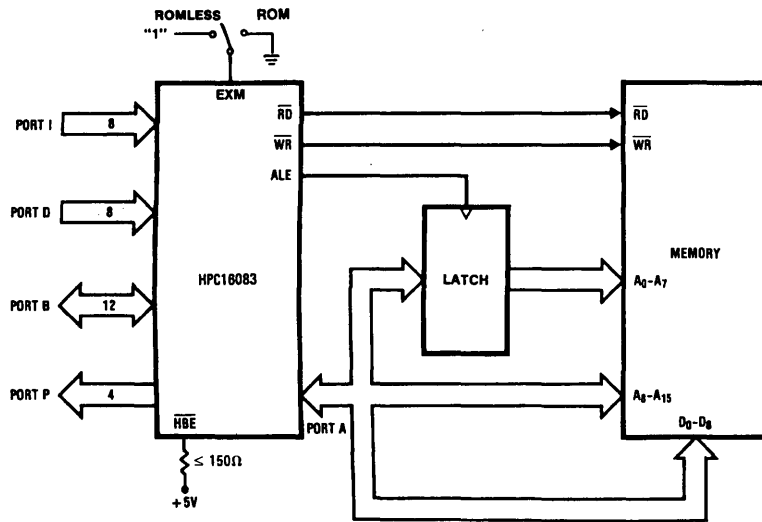


FIGURE 16. 8-Bit External Memory

TL/DD/8801-18

HPC16003 Operating Modes (Continued)

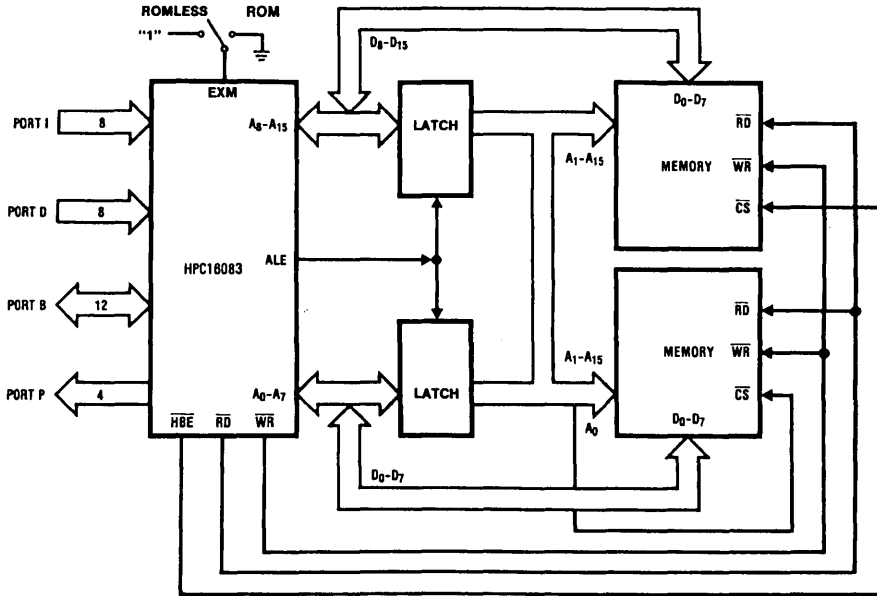


FIGURE 17. 16-Bit External Memory

TL/DD/8801-19

Wait States

The internal ROM can be accessed at the maximum operating frequency with one wait state. With 0 wait states, internal ROM accesses are limited to $\frac{2}{3} f_C$ max.

The HPC16083 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC16083: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16083 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16083 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the RESET or the NMI. The RESET input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16083 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC16083 to resume normal operation.

HPC16083 Interrupts

Complex interrupt handling is easily accomplished by the HPC16083's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table IV.

TABLE IV. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
FFFF:FFFE	RESET	0
FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
FFFB:FFFA	External interrupt on I2 pin	2
FFF9:FFF8	External interrupt on I3 pin	3
FFF7:FFF6	External interrupt on I4 pin	4
FFF5:FFF4	Overflow on internal timers	5
FFF3:FFF2	Internal on the UART transmit/receive complete or external on EXUI	6
FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC16083 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table IV. The interrupt on $\overline{\text{RESET}}$ has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the $\overline{\text{RESET}}$, which is serviced immediately. $\overline{\text{RESET}}$ and $\overline{\text{EXUI}}$ are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt ($\overline{\text{EXUI}}$) is shared with the UART interrupt. This interrupt is level-low sensitive. To select this interrupt disable the ERI and ETI UART interrupt bits in the ENUI register. To select the UART interrupt leave this pin floating or tie it high.

Interrupt Control Registers

The HPC16083 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

$\overline{\text{RESET}}$ and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the

interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16083 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 18* shows the Interrupt Enable Logic.

RESET

The $\overline{\text{RESET}}$ input initializes the processor and sets ports A and B in the TRI-STATE condition and port P in the LOW state. $\overline{\text{RESET}}$ is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between E000 and FFFF when using the HPC16003.

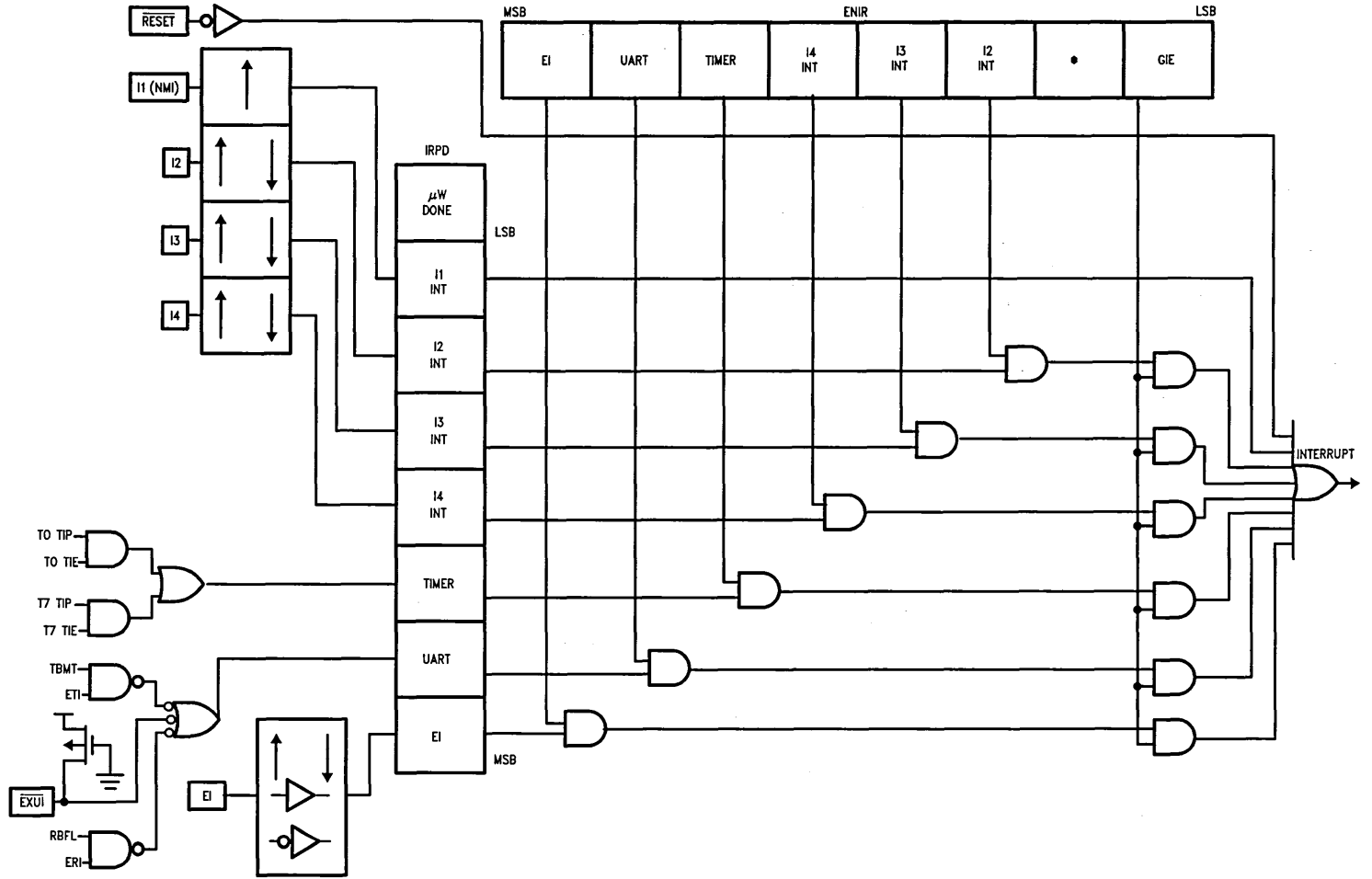


FIGURE 18. Block Diagram of Interrupt Logic

Timer Overview

The HPC16083 contains a powerful set of flexible timers enabling the HPC16083 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16083 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for WATCHDOG logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see Figure 19).

The HPC16083 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see Figure 20).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under

software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

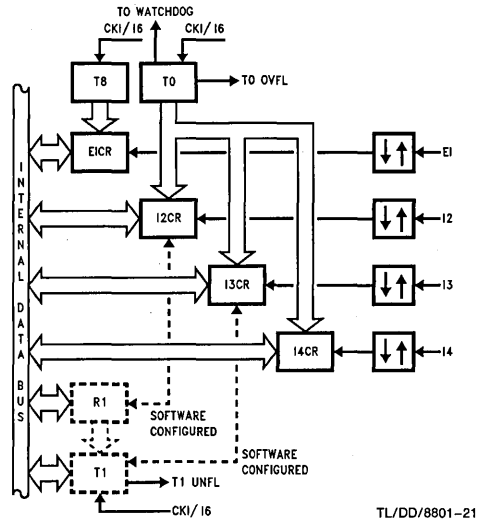


FIGURE 19. Timers T0, T1 and T8 with Four Input Capture Registers

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16083 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see Figure 20). Timer/register pairs 4-7 form four identical units which can generate synchronous outputs on port P (see Figure 21).

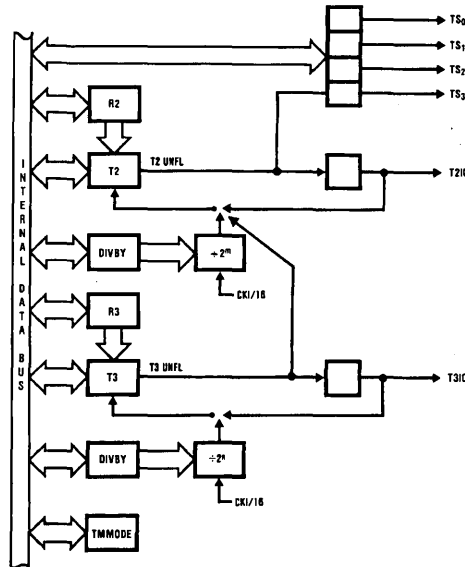
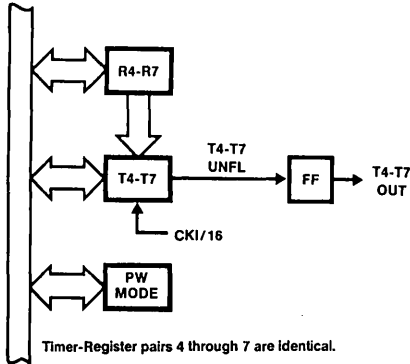


FIGURE 20. Timers T2-T3 Block

Timer Overview (Continued)



Timer-Register pairs 4 through 7 are identical.

TL/DD/8801-23

FIGURE 21. Timers T4-T7 Block

Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to 1/2 the frequency of the source used for clocking the timer.

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16083.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.



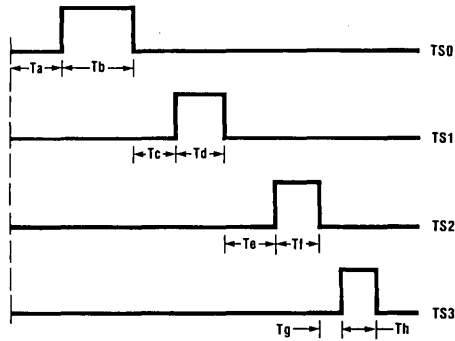
TL/DD/8801-24

FIGURE 22. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0-TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. Figure 23 is an example of synchronous pulse train generation.

WATCHDOG Logic

The WATCHDOG Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the WATCHDOG logic are potentially infinite loops and illegal addresses. Should the



TL/DD/8801-25

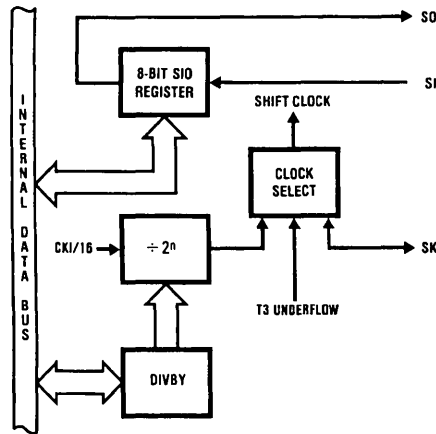
FIGURE 23. Synchronous Pulse Generation

WATCHDOG register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the WATCHDOG Output (WO) pin low. The WO pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

*Note: See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see Figure 24). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.



TL/DD/8801-26

FIGURE 24. MICROWIRE/PLUS

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS Operation

The HPC16083 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16083 is the master or slave. The shift clock is generated when the HPC16083 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16083 is configured as a slave. When the HPC16083 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz. The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 25 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based sys-

tem could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC16083 microcontrollers interconnected to other MICROWIRE peripherals. HPC16083 #1 is set up as the master and initiates all data transfers. HPC16083 #2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of a LCD display controlled by the COP472 display driver. The data to be displayed is sent serially to the COP472 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC16083 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.

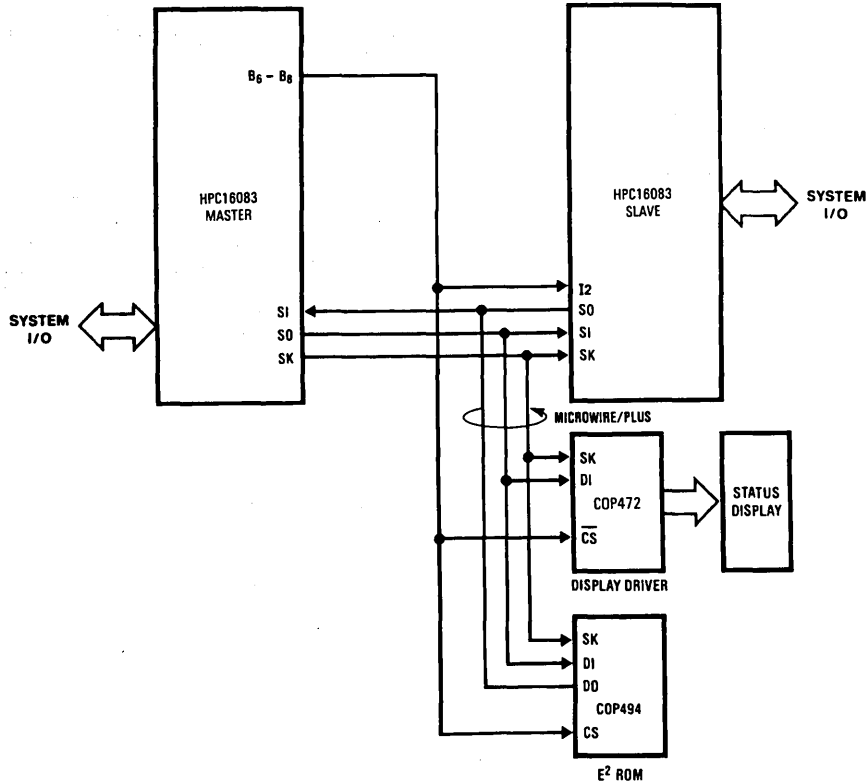


FIGURE 25. MICROWIRE/PLUS Application

TL/DD/8801-27

HPC16083 UART

The HPC16083 contains a software programmable UART. The UART (see Figure 26) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENUI register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC16083 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16083 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16083 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16083 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

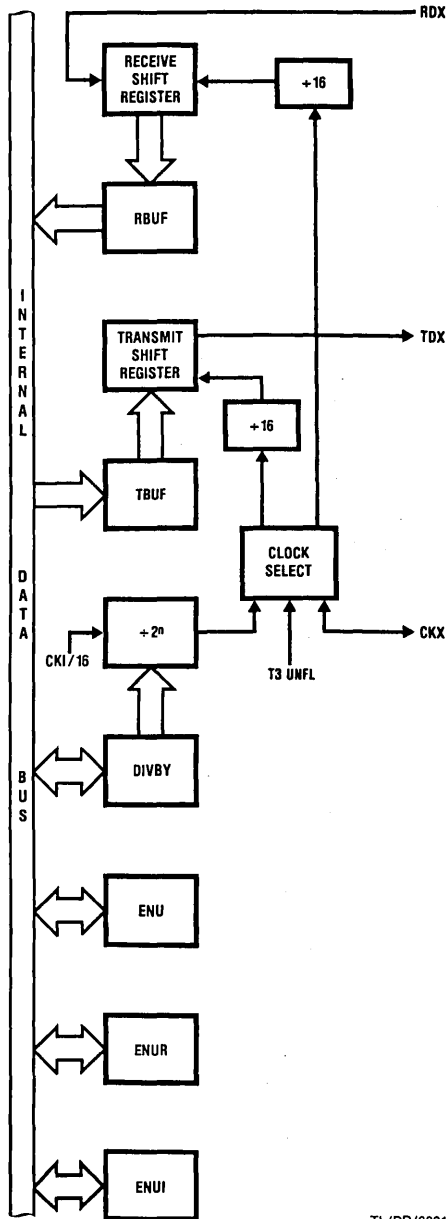


FIGURE 26. UART Block Diagram

TL/DD/8801-28

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC16083 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC16083's and set up systems with very high data exchange rates. Another area of application could be where a HPC16083 is programmed as an intelligent peripheral to a host system such as the Series 32000[®] microprocessor. *Figure 27* illustrates how a HPC16083 could be used as an intelligent peripheral for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe (\overline{URD}), a Write Strobe (\overline{UWR}), a Read Ready Line (\overline{RDRDY}), a Write Ready Line (\overline{WRRDY}) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The \overline{URD} and \overline{UWR} inputs may be used to interrupt the HPC16083. The \overline{RDRDY} and \overline{WRRDY} outputs may be used to interrupt the host processor.

The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC16083 is the data bus. UPI can only be used if the HPC16083 is in the Single-Chip mode.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16083 supports shared memory access with two pins. The pins are the $\overline{RDY}/\overline{HLD}$ input pin and the \overline{HLDA} output pin. The user can software select either the Hold or Ready function by the state of a control bit. The \overline{HLDA} output is multiplexed onto port B.

The host uses DMA to interface with the HPC16083. The host initiates a data transfer by activating the \overline{HLD} input of the HPC16083. In response, the HPC16083 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal (\overline{HLDA}) from the HPC16083 indicating that the system bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16083 resumes normal operations.

Figure 28 illustrates an application of the shared memory interface between the HPC16083 and a Series 32000 system. To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus.

Memory

The HPC16083 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 8 kbytes of ROM and 256 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16083 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC16083 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table V.

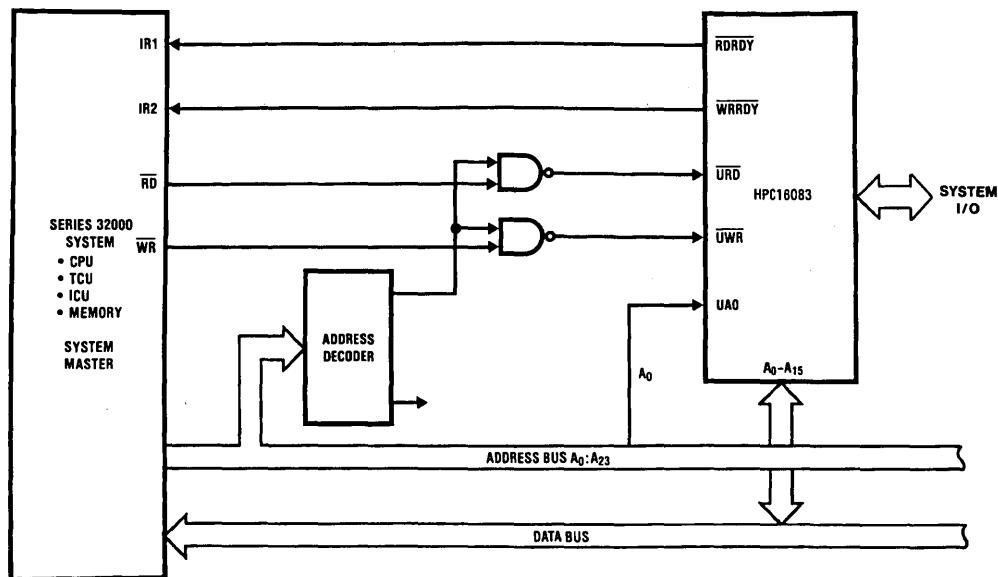


FIGURE 27. HPC16083 as a Peripheral: (UPI Interface to Series 32000 Application)

TL/DD/8801-29

Shared Memory Support (Continued)

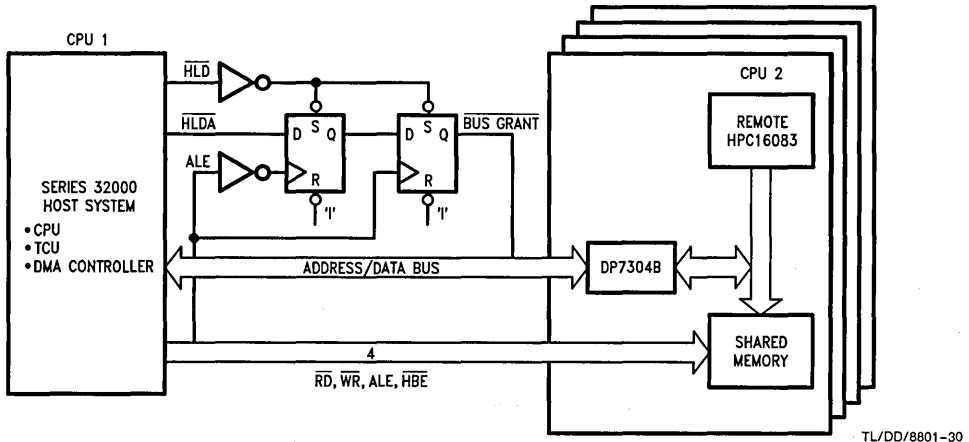


FIGURE 28. Shared Memory Application: HPC16083 Interface to Series 32000 System

TL/DD/8801-30

TABLE V. HPC16083 Memory Map

FFFF:FF0 FFEF:FFD0 FFCF:FFCE : : E001:E000	Interrupt Vectors JSRP Vectors On-Chip ROM	USER MEMORY
DFFF:DFFE : : 0201:0200	External Expansion Memory	
01FF:01FE : : 01C1:01C0	On-Chip RAM	USER RAM
0195:0194	WATCHDOG Address	WATCHDOG Logic
0192 0191:0190 018F:018E 018D:018C 018B:018A 0189:0188 0187:0186 0185:0184 0183:0182 0181:0180	T0CON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/ R1 I3CR Register/ T1 I4CR Register	Timer Block T0:T3
015E:015F 015C 0153:0152 0151:0150 014F:014E 014D:014C 014B:014A 0149:0148 0147:0146 0145:0144 0143:0142 0141:0140	EICR EICON Port P Register PWMODE Register R7 Register T7 Timer R6 Register T6 Timer R5 Register T5 Timer R4 Register T4 Timer	Timer Block T4:T7

0128 0126 0124 0122 0120	ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART
0104	Port D Input Register	
00F5:00F4 00F3:00F2 00F1:00F0	BFUN Register DIR B Register DIR A Register / IBUF	PORTS A & B CONTROL
00E6	UPIC Register	UPI CONTROL
00E3:00E2 00E1:00E0	Port B Port A / OBUF	PORTS A & B
00DE:00DF 00DD:00DC 00D8 00D6 00D4 00D2 00D0	(reserved) HALT Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00CF:00CE 00CD:00CC 00CB:00CA 00C9:00C8 00C7:00C6 00C5:00C4 00C3:00C2 00C0	X Register B Register K Register A Register PC Register SP Register (reserved) PSW Register	HPC CORE REGISTERS
00BF:00BE : : 0001:0000	On-Chip RAM	USER RAM

HPC16083/HPC26083/HPC36083/HPC46083/HPC16003/HPC26003/HPC36003/HPC46003

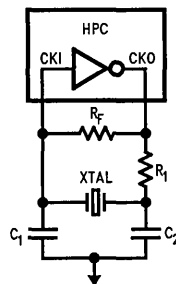
Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused outputs should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least $1 \mu F$ and bypass their outputs with a $10 \mu F$ to $50 \mu F$ tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a $10 \mu F$ to $20 \mu F$ tantalum electrolytic capacitor or a $50 \mu F$ to $100 \mu F$ aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.



TL/DD/8801-40

FIGURE 29. Recommended Crystal Circuit

A recommended crystal oscillator circuit to be used with the HPC is shown below. See table for recommended component values. The recommended values given in the table below have yielded consistent results and are made to match a crystal with a 18 pF load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within $1''$ distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A $1.0 \mu F$, a $0.1 \mu F$, and a $0.001 \mu F$ dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

HPC Oscillator Table

XTAL Frequency (MHz)	$R_1 (\Omega)$
≤ 2	1500
4	1200
6	910
8	750
10	600
12	470
14	390
16	300
18	220
20	180
22	150
24	120
26	100
28	75
30	62

$R_F = 3.3 \text{ M}\Omega$

$C_1 = 27 \text{ pF}$

$C_2 = 33 \text{ pF}$

XTAL Specifications: The crystal used was an M-TRON Industries MP-1 Series XTAL. "AT" cut parallel resonant

$C_L = 18 \text{ pF}$

Series Resistance is:

$25 \Omega @ 25 \text{ MHz}$

$40 \Omega @ 10 \text{ MHz}$

$600 \Omega @ 2 \text{ MHz}$

HPC16083 CPU

The HPC16083 CPU has a 16-bit ALU and six 16-bit registers

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16083 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	$MA + Meml \rightarrow MA$ carry $\rightarrow C$
ADC	Add with carry	$MA + Meml + C \rightarrow MA$ carry $\rightarrow C$
ADDS	Add short imm8	$MA + imm8 \rightarrow MA$ carry $\rightarrow C$
DADC	Decimal add with carry	$MA + Meml + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
SUBC	Subtract with carry	$MA - Meml + C \rightarrow MA$ carry $\rightarrow C$
DSUBC	Decimal subtract w/carry	$MA - Meml + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
MULT	Multiply (unsigned)	$MA * Meml \rightarrow MA \& X, 0 \rightarrow K, 0 \rightarrow C$
DIV	Divide (unsigned)	$MA / Meml \rightarrow MA, rem. \rightarrow X, 0 \rightarrow K, 0 \rightarrow C$
DIVD	Divide Double Word (unsigned)	$(X \& MA) / Meml \rightarrow MA, rem \rightarrow X, 0 \rightarrow K, carry \rightarrow C$
IFEQ	If equal	Compare MA & Meml, Do next if equal
IFGT	If greater than	Compare MA & Meml, Do next if MA > Meml
AND	Logical and	$MA \text{ and } Meml \rightarrow MA$
OR	Logical or	$MA \text{ or } Meml \rightarrow MA$
XOR	Logical exclusive-or	$MA \text{ xor } Meml \rightarrow MA$
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	$Mem + 1 \rightarrow Mem$
DECSZ	Decrement, skip if 0	$Mem - 1 \rightarrow Mem$, Skip next if Mem = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Mem1 → MA
ST	Load, incr/decr X	Mem(X) → A, X ± 1 (or 2) → X
X	Store to Memory	A → Mem
	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ± 1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP
POP	Pop Stack to Memory	SP - 2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ± 1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A, B ± 1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B, imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A - 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → [SP], SP + 2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → [SP], SP + 2 → SP, PC + # → PC (# is +1025 to -1023)
JSRL	Jump subroutine long	PC → [SP], SP + 2 → SP, PC + # → PC
JP	Jump relative short	PC + # → PC (# is +32 to -31)
JMP	Jump relative	PC + # → PC (# is +257 to -255)
JMPL	Jump relative long	PC + # → PC
JID	Jump indirect at PC + A	PC + A + 1 → PC then Mem(PC) + PC → PC
JIDW		
NOP	No Operation	PC + 1 → PC
RET	Return	SP - 2 → SP, [SP] → PC
RETSK	Return then skip next	SP - 2 → SP, [SP] → PC, & skip
RETI	Return from interrupt	SP - 2 → SP, [SP] → PC, interrupt re-enabled

Note: W is 16-bit word of memory

MA is Accumulator A or direct memory (8 or 16-bit)

Mem is 8-bit byte or 16-bit word of memory

Mem1 is 8- or 16-bit memory or 8 or 16-bit immediate data

imm is 8-bit or 16-bit immediate data

imm8 is 8-bit immediate data only

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir.		Direct	Indir.	Index	Immed.	Direct		Immed.	
	(B)	(X)					*	**	*	**
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B-)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X-)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16083 has been designed to be extremely code-efficient. The HPC16083 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16083, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16083 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of the 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into the table; the assembler can give this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16083 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16083 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16083 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

HPC MICROCONTROLLER DEVELOPMENT SYSTEM

The HPC microcontroller development system is an in-system emulator (ISE) designed to support the entire family of HPC Microcontrollers. The complete package of hardware and software tools combined with a host system provides a powerful system for design, development and debug of HPC based designs. Software tools are available for IBM® PC/AT® (MS-DOS, PC-DOS) and for UNIX® based multi-user Sun® SPARCstation (SunOSTM).

The stand alone units comes complete with a power supply and external emulation POD. This unit can be connected to various host systems through an RS-232 link. The software package includes an ANSI compatible C-Compiler, Linker, Assembler and librarian package. Source symbolic debug capability is provided through a user friendly MS-windows 3.0 interface for IBM PC/AT environments and through a line debugger under Sunview for Sun SPARCstations.

The ISE provides fully transparent in-system emulation at speeds up to 20 MHz 1 waitstate. A 2k word (48-bit wide) trace buffer gives trace trigger and non intrusive monitoring of the system. External triggering is also available through an external logic interface socket on the POD. Direct EPROM programming can be done through the use of externally mounted EPROM socket. Form-Fit-Function emulator programming is supported by a programming board included with the system. Comprehensive on-line help and diagnostics features reduce user's design and debug time. 8 hardware breakpoints (Address/range), 64k bytes of user memory, and break on external events are some of the other features offered.

Hewlett Packard model HP64775 Emulator/Analyzer providing in-system emulation for up to 30 MHz 1 waitstate is also available. Contact your local sales office for technical details and support.

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum require-

ment for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MDS-DIAL-A-HLP

Information system package contains:
 DIAL-A-HELPER Users Manual
 Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MDS, he can leave messages on our electronic bulletin board, which we will respond to.

Development Tools Selection Table

Product	Order Number	Description	Includes	Manual Number
HPC16003/16083	HPC-DEV-ISE1 HPC-DEV-ISE1-E	HPC In-System Emulator HPC In-System Emulator for Europe and South East Asia	HPC MDS User's Manual MDS Comm User's Manual HPC Emulator Programmer User's Manual HPC16083/16004/16064 Manual	420420184-001 424420188-001 420421313-001 424410897-001
	HPC-DEV-IBMA	Assembler/Linker/Library Package for IBM PC/AT	HPC Assembler/Linker Librarian User's Manual	424410836-001
	HPC-DEV-IBMC	C Compiler/Assembler/Linker/Library Package for IBM PC/AT	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-0 424410836-001
	HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC/AT C Compiler/Assembler/Linker/Library Package for IBM PC/AT	Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424420189-001 424410883-001 424410836-001
	HPC-DEV-SUNC HPC-DEV-SUNDB	C Compiler/Assembler/Linker Library Package for SUN SPARCstation Source/Symbolic Debugger for Sun SPARCstation C Compiler/Assembler/Linker Library Package	HPC C Compiler User's Manual HCP Assembler/Linker/Library User's Manual Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	

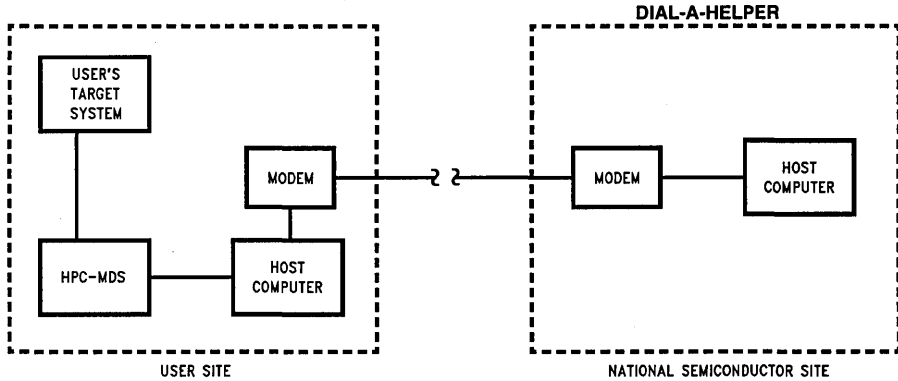
COMPLETE SYSTEM

HPC16003/16083	HPC-DEV-SYS1 HPC-DEV-SYS1-E	HPC In-System Emulator with C Compiler/Assembler/Linker/Library and Source Symbolic Debugger Same for Europe and South East Asia		
----------------	--------------------------------	--	--	--

VAX™ UNIX will be supported in the near future. Contact field sales for more information.

Development Support (Continued)

Voice: (408) 721-5582
 Modem: (408) 739-1162
 Baud: 300 or 1200 Baud
 Set-Up: Length: 8-bit
 Parity: None
 Stop Bit: 1
 Operation: 24 hrs, 7 days

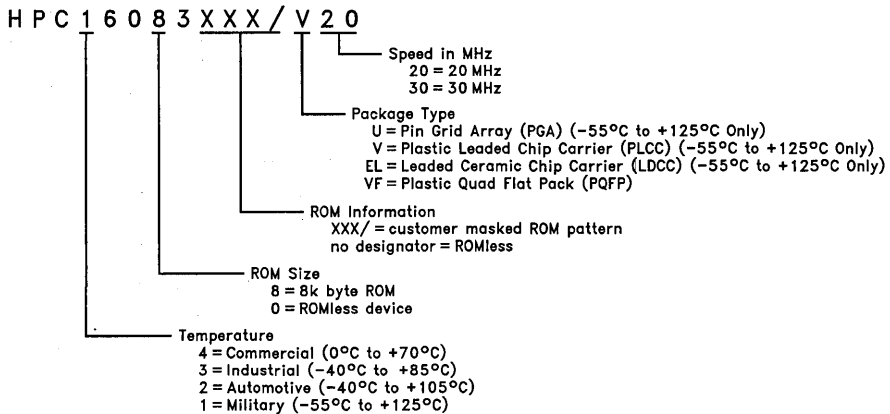


TL/DD/8801-32

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16083 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/8801-31

FIGURE 30. HPC Family Part Numbering Scheme

Examples

- HPC46003V20 — ROMless, Commercial temp. (0°C to 70°C), PLCC
- HPC16083XXX/U20 — 8k masked ROM, Military temp. (-55°C to +125°C), PGA
- HPC26083XXX/V20 — 8k masked ROM, Automotive temp. (-40°C to +105°C), PLCC



HPC36164/46164, HPC36104/46104

High-Performance microController with A/D

General Description

The HPC46164 and HPC46104 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC46164 has 16k bytes of on-chip ROM. The HPC46104 has no on-chip ROM and is intended for use with external memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICRO-WIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC46164" is used throughout this data-sheet to refer to the HPC46164 and HPC46104 devices unless otherwise specified.

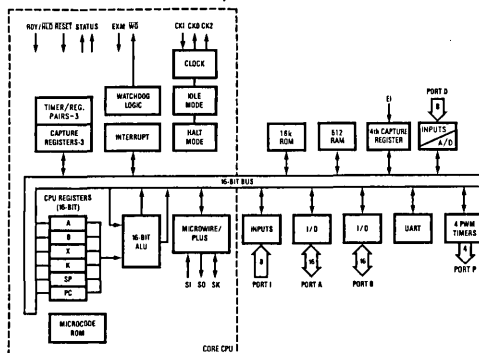
The HPC46164 and HPC46104 have, as an on-board peripheral, an 8-channel 8-bit Analog-to-Digital Converter. This A/D converter can operate in a single-ended mode where the analog input voltage is applied across one of the eight input channels (D0–D7) and AGND. The A/D converter can also operate in differential mode where the analog input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel pairs in differential mode.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available only in an 80-pin PQFP package.

Features

- HPC family—core features:
 - 16-bit architecture, both byte and word
 - 16-bit data bus, ALU, and registers
 - 64k bytes of external direct memory addressing
 - FAST—200 ns for fastest instruction when using 20.0 MHz clock, 134 ns at 30.0 MHz
 - High code efficiency—most instructions are single byte
 - 16 x 16 multiply and 32 x 16 divide
 - Eight vectored interrupt sources
 - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
 - MICROWIRE/PLUS serial I/O interface
 - CMOS—very low power with two power save modes: IDLE and HALT
- A/D—8-channel 8-bit analog-to-digital converter with conversion time
- Minimum 7.5 μ s for single conversion
- A/D—supports conversions in "quiet mode"
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 16k bytes of ROM, 512 bytes of RAM on-chip
- ROMless version available (HPC46104)
- Commercial (0°C to +70°C) and industrial (-40°C to +85°C) temperature ranges

Block Diagram (HPC46164 with 16k ROM shown)



TL/DD/9682-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

V_{CC} with Respect to GND -0.5V to 7.0V
All Other Pins ($V_{CC} + 0.5$)V to (GND - 0.5)V

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

$V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46164/HPC46104, $-40^\circ C$ to $+85^\circ C$ for HPC36164/HPC36104

Symbol	Parameter	Test Conditions	Min	Max	Units
I _{CC1}	Supply Current	$V_{CC} = 5.5V, f_{in} = 30$ MHz (Note 1)		65	mA
		$V_{CC} = 5.5V, f_{in} = 20$ MHz (Note 1)		47	mA
		$V_{CC} = 5.5V, f_{in} = 2.0$ MHz (Note 1)		15	mA
I _{CC2}	IDLE Mode Current	$V_{CC} = 5.5V, f_{in} = 30$ MHz (Note 1)		5	mA
		$V_{CC} = 5.5V, f_{in} = 20$ MHz (Note 1)		3	mA
		$V_{CC} = 5.5V, f_{in} = 2.0$ MHz (Note 1)		1	mA
I _{CC3}	HALT Mode Current	$V_{CC} = 5.5V, f_{in} = 0$ kHz (Note 1)		300	μA
		$V_{CC} = 2.5V, f_{in} = 0$ kHz (Note 1)		100	μA

INPUT VOLTAGE LEVELS FOR SCHMITT TRIGGERED INPUTS RESET, NMI, \overline{WO} ; AND ALSO CKI

V _{IH1}	Logic High		0.9 V _{CC}		V
V _{IL1}	Logic Low			0.1 V _{CC}	V

ALL OTHER INPUTS

V _{IH2}	Logic High (except Port D)		0.7 V _{CC}	$V_{CC} + 0.3$	V
V _{IL2}	Logic Low (except Port D)		GND - 0.3	0.2 V _{CC}	V
V _{IH3}	Logic High (Port D Only)		0.7 V _{CC}	V _{CC}	V
V _{IL3}	Logic Low (Port D Only)		GND	0.2 V _{CC}	V
I _{LI1}	Input Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 2	μA
I _{LI2}	Input Leakage Current RDY/HLD, EXU \overline{I}	$V_{IN} = 0$	-3	-50	μA
I _{LI3}	Input Leakage Current B12	RESET = 0, $V_{IN} = V_{CC}$	0.5	7	μA
C _I	Input Capacitance	(Note 2)		10	pF
C _{IO}	I/O Capacitance	(Note 2)		20	pF

OUTPUT VOLTAGE LEVELS

V _{OH1}	Logic High (CMOS)	I _{OH} = -10 μA (Note 2)	$V_{CC} - 0.1$		V
V _{OL1}	Logic Low (CMOS)	I _{OH} = 10 μA (Note 2)		0.1	V
V _{OH2}	Port A/B Drive, CK2 (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅)	I _{OH} = -7 mA	2.4		V
V _{OL2}		I _{OL} = 3 mA		0.4	V
V _{OH3}	Other Port Pin Drive, \overline{WO} (open drain) (B ₀ -B ₉ , B ₁₃ , B ₁₄ , P ₀ -P ₃)	I _{OH} = -1.6 mA (except \overline{WO})	2.4		V
V _{OL3}		I _{OL} = 0.5 mA		0.4	V
V _{OH4}	ST1 and ST2 Drive	I _{OH} = -6 mA	2.4		V
V _{OL4}		I _{OL} = 1.6 mA		0.4	V
V _{OH5}	Port A/B Drive (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅) When Used as External Address/Data Bus	I _{OH} = -1 mA	2.4		V
V _{OL5}		I _{OL} = 3 mA		0.4	V
V _{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V _{CC}	V
I _{OZ}	TRI-STATE [®] Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 5	μA

Note 1: I_{CC1}, I_{CC2}, I_{CC3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC1} is measured with RESET = V_{SS}. I_{CC3} is measured with NMI = V_{CC} and A/D inactive. CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns. V_{REF} = AGND = GND.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

20 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and *Figure 1* through *Figure 5*.) $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46164 and $-40^\circ C$ to $+85^\circ C$ for HPC36164.

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Clocks	f_C	CKI Operating Frequency	2	20	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	50	500	ns	
	t_{CKIH}	CKI High Time	22.5		ns	
	t_{CKIL}	CKI Low Time	22.5		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	100		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	100		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CKI Falling Edge	0	55	ns	(Note 1)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CKI Falling Edge	0	55	ns	(Note 1)
Timers	$f_{XIN} = f_C/22$	External Timer Input Frequency		0.91	MHz	
	$t_{XIN} = t_C$	Pulse Width for Timer Inputs	100		ns	
MICROWIRE/PLUS	t_{UWS}	MICROWIRE Setup Time Master Slave	100 20		ns	
	t_{UWH}	MICROWIRE Hold Time Master Slave	20 50		ns	
	t_{UWV}	MICROWIRE Output Valid Time Master Slave		50 150	ns	
External Hold	$t_{SALE} = \frac{3}{4} t_C + 40$	\overline{HLD} Falling Edge before ALE Rising Edge	115		ns	
	$t_{HWP} = t_C + 10$	\overline{HLD} Pulse Width	110		ns	
	$t_{HAE} = t_C + 100$	$\overline{HLD\bar{A}}$ Falling Edge after \overline{HLD} Falling Edge		200	ns	(Note 3)
	$t_{HAD} = \frac{3}{4} t_C + 85$	$\overline{HLD\bar{A}}$ Rising Edge after \overline{HLD} Rising Edge		160	ns	
	$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after $\overline{HLD\bar{A}}$ Falling Edge		116	ns	(Note 5)
	$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable after $\overline{HLD\bar{A}}$ Rising Edge	116		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	(Note 6)
	t_{DRDY}	\overline{RDRDY} Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH}	Input Data Hold after Rising Edge of \overline{UWR}	20		ns	
	t_A	\overline{WRRDY} Delay from Rising Edge of \overline{UWR}		70	ns	

*This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

20 MHz (Continued)

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 through Figure 5.) $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46164 and -40°C to $+85^\circ\text{C}$ for HPC36164.

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Address Cycles	$t_{DC1ALER}$	Delay from CK1 Rising Edge to ALE Rising Edge	0	35	ns	(Notes 1, 2)
	$t_{DC1ALEF}$	Delay from CK1 Rising Edge to ALE Falling Edge	0	35	ns	(Notes 1, 2)
	$t_{DC2ALER} = \frac{1}{4} t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		45	ns	(Note 2)
	$t_{DC2ALEF} = \frac{1}{4} t_C + 20$	Delay from CK2 Falling Edge to ALE Falling Edge		45	ns	(Note 2)
	$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns	
	$t_{ST} = \frac{1}{4} t_C - 7$	Setup of Address Valid before ALE Falling Edge	18		ns	
	$t_{VP} = \frac{1}{4} t_C - 5$	Hold of Address Valid after ALE Falling Edge	20		ns	
Read Cycles	$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	20		ns	
	$t_{ACC} = t_C + WS - 55$	Data Input Valid after Address Output Valid		145	ns	(Note 6)
	$t_{RD} = \frac{1}{2} t_C + WS - 65$	Data Input Valid after \overline{RD} Falling Edge		85	ns	
	$t_{RW} = \frac{1}{2} t_C + WS - 10$	\overline{RD} Pulse Width	140		ns	
	$t_{DR} = \frac{3}{4} t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	60	ns	
	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	85		ns	
Write Cycles	$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	45		ns	
	$t_{WW} = \frac{3}{4} t_C + WS - 15$	\overline{WR} Pulse Width	160		ns	
	$t_V = \frac{1}{2} t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	145		ns	
	$t_{HW} = \frac{1}{4} t_C - 5$	Hold of Data Valid after \overline{WR} Rising Edge	20		ns	
Ready Input	$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns	
	$t_{RWP} = t_C$	RDY Pulse Width	100		ns	

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CK1 having 50% duty cycle and with less than 15 pF load on CKO with rise and fall times (t_{CK1R} and t_{CK1L}) on CK1 input less than 2.5 ns.

Note 2: Do not design with these parameters unless CK1 is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CK1 or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{HLD} falling edge occurring at the latest time it can be accepted during the present CPU cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} may be as long as $(3 t_C + 4WS + 72 t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: WS (t_{WAIT}) \times (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 20$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

Note 6: This is guaranteed by design and not tested.

A/D Converter Specifications

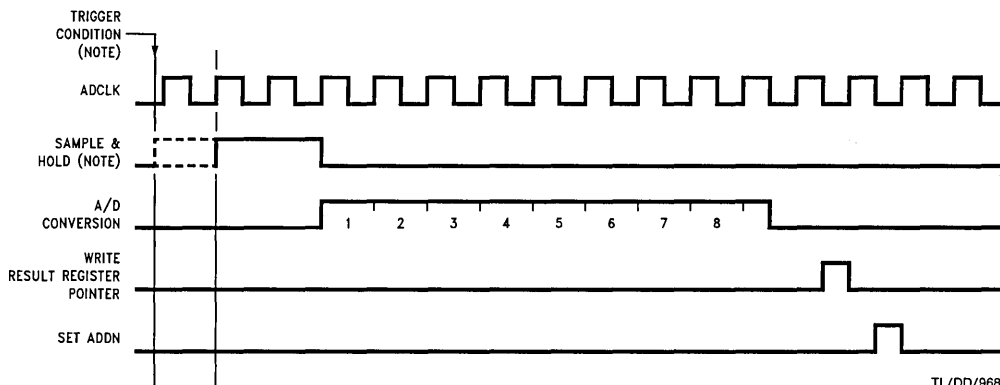
$V_{CC} = 5V \pm 10\% (V_{SS} - 0.05V) \leq$ Any Input $\leq (V_{CC} + 0.05V)$, $f_C = 20$ MHz and Prescalar = $f_C/12$.

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{CC} = 5.5V, V_{REF} = 5V,$ $V_{CC} = 5V, V_{REF} = 5V$ and $V_{CC} = 4.5V, V_{REF} = 4.5V$			± 2	LSB
Non-Linearity	$V_{CC} = 5.5V, V_{REF} = 5V,$ $V_{CC} = 5V, V_{REF} = 5V$ and $V_{CC} = 4.5V, V_{REF} = 4.5V$			$\pm 1/2$	LSB
Differential Non-Linearity	$V_{CC} = 5.5V, V_{REF} = 5V,$ $V_{CC} = 5V, V_{REF} = 5V$ and $V_{CC} = 4.5V, V_{REF} = 4.5V$			$\pm 1/2$	LSB
Input Reference Resistance		1.6		4.8	k Ω
Common Mode Input Range (Note 8)		AGND		V_{REF}	V
DC Common Mode Error				$\pm 1/4$	LSB
Off Channel Leakage Current				± 2	μA
On Channel Leakage Current				± 2	μA
A/D Clock Frequency (Note 8)		0.1		1.67	MHz
Conversion Time (Note 7)		12.5			A/D Clock Cycles

Note 7: Conversion Time includes sample and hold time. See following diagrams.

Note 8: See Prescalar description.

Timing Diagram



TL/DD/9682-11

Note: The trigger condition generated by the start conversion method selected by the SC bits requires one CK2 to propagate through before the trigger condition is known. Once the trigger condition is known, the sample and hold will start at the next rising edge of ADCLK. The figure shows worst case.

30 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 through Figure 5.) $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46164/HPC46104, -55°C to $+125^\circ\text{C}$ for HPC16164/HPC16104.

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Clocks	f_C	CKI Operating Frequency	2	30	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	33	500	ns	
	t_{CKIH}	CKI High Time	15		ns	
	t_{CKIL}	CKI Low Time	16.6		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	66		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	66		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CKI Falling Edge	0	55	ns	(Note 1)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CKI Falling Edge	0	55	ns	(Note 1)
	$f_U = f_C/8$ f_{MW}	External UART Clock Input Frequency External MICROWIRE/PLUS Clock Input Frequency		3.75* 1.875	MHz MHz	
Timers	$f_{XIN} = f_C/22$ $t_{XIN} = t_C$	External Timer Input Frequency Pulse Width for Timer Inputs	66	1.36	MHz ns	
	MICROWIRE/PLUS	t_{UWS}	MICROWIRE Setup Time Master Slave	100 20		ns
t_{UWH}		MICROWIRE Hold Time Master Slave	20 50		ns	
t_{UWV}		MICROWIRE Output Valid Time Master Slave		50 150	ns	
External Hold	$t_{SALE} = \frac{3}{4}t_C + 40$	HLDA Falling Edge before ALE Rising Edge	90		ns	
	$t_{HWP} = t_C + 10$	HLDA Pulse Width	76		ns	
	$t_{HAE} = t_C + 85$	HLDA Falling Edge after HLD Falling Edge		151	ns	(Note 3)
	$t_{HAD} = \frac{3}{4}t_C + 85$	HLDA Rising Edge after HLD Rising Edge		135	ns	
	$t_{BF} = \frac{1}{2}t_C + 66$	Bus Float after HLDA Falling Edge		99	ns	(Note 5)
	$t_{BE} = \frac{1}{2}t_C + 66$	Bus Enable after HLDA Rising Edge	99		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	(Note 6)
	t_{DRDY}	\overline{RDRDY} Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH}	Input Data Hold after Rising Edge of \overline{UWR}	20		ns	
	t_A	\overline{WRRDY} Delay from Rising Edge of \overline{UWR}		70	ns	

*This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

30 MHz (Continued)

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 through Figure 5.) $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46164/HPC46104, -55°C to $+125^\circ\text{C}$ for HPC16164/HPC16104.

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Address Cycles	$t_{DC1ALER}$	Delay from CK1 Rising Edge to ALE Rising Edge	0	35	ns	(Notes 1, 2)
	$t_{DC1ALEF}$	Delay from CK1 Rising Edge to ALE Falling Edge	0	35	ns	(Notes 1, 2)
	$t_{DC2ALER} = \frac{1}{4} t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		37	ns	(Note 2)
	$t_{DC2ALEF} = \frac{1}{4} t_C + 20$	Delay from CK2 Falling Edge to ALE Falling Edge		37	ns	(Note 2)
	$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	24		ns	
	$t_{ST} = \frac{1}{4} t_C - 7$	Setup of Address Valid before ALE Falling Edge	9		ns	
	$t_{HP} = \frac{1}{4} t_C - 5$	Hold of Address Valid after ALE Falling Edge	11		ns	
Read Cycles	$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	11		ns	
	$t_{ACC} = t_C + WS - 32$	Data Input Valid after Address Output Valid		100	ns	(Note 6)
	$t_{RD} = \frac{1}{2} t_C + WS - 39$	Data Input Valid after \overline{RD} Falling Edge		60	ns	
	$t_{RW} = \frac{1}{2} t_C + WS - 14$	\overline{RD} Pulse Width	85		ns	
	$t_{DR} = \frac{3}{4} t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	35	ns	
	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	51		ns	
Write Cycles	$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	28		ns	
	$t_{WW} = \frac{3}{4} t_C + WS - 15$	\overline{WR} Pulse Width	101		ns	
	$t_V = \frac{1}{2} t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	94		ns	
	$t_{HW} = \frac{1}{4} t_C - 10$	Hold of Data Valid after \overline{WR} Rising Edge	7		ns	
Ready Input	$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		33	ns	
	$t_{RWP} = t_C$	RDY Pulse Width	66		ns	

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CK1 having 50% duty cycle and with less than 15 pF load on CK0 with rise and fall times (t_{CKIR} and t_{CKIF}) on CK1 input less than 2.5 ns.

Note 2: Do not design with these parameters unless CK1 is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CK1 or CK0 is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is specified for case with \overline{HLD} falling edge occurring at the latest time it can be accepted during the present CPU cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} may be as long as $(3 t_C + 4WS + 72 t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: $WS (t_{WAIT}) \times$ (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 30$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

Note 6: This is guaranteed by design and not tested.

CKI Input Signal Characteristics

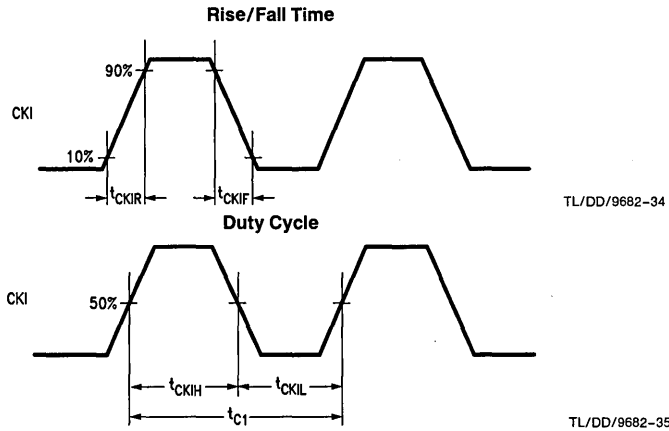
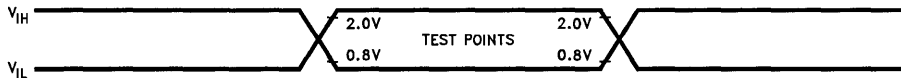


FIGURE 1. CKI Input Signal



TL/DD/9682-40

Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

FIGURE 2. Input and Output for AC Tests

Timing Waveforms

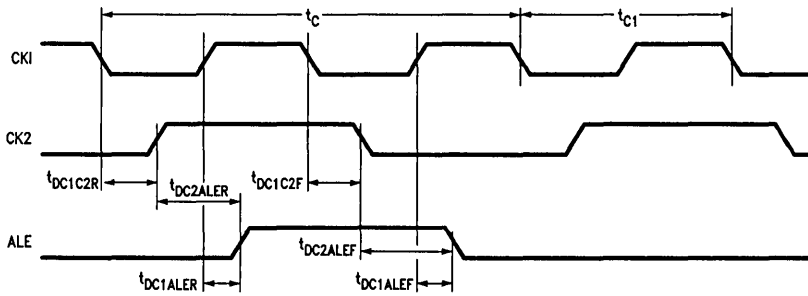


FIGURE 3. CK1, CK2, ALE Timing Diagram

TL/DD/9682-2

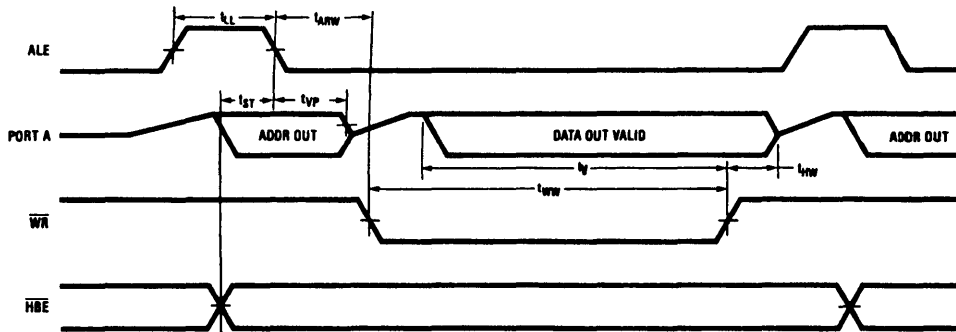


FIGURE 4. Write Cycle

TL/DD/9682-3

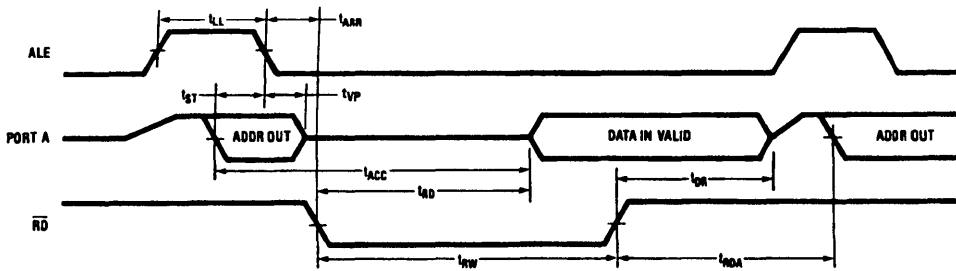


FIGURE 5. Read Cycle

TL/DD/9682-4

Timing Waveforms (Continued)

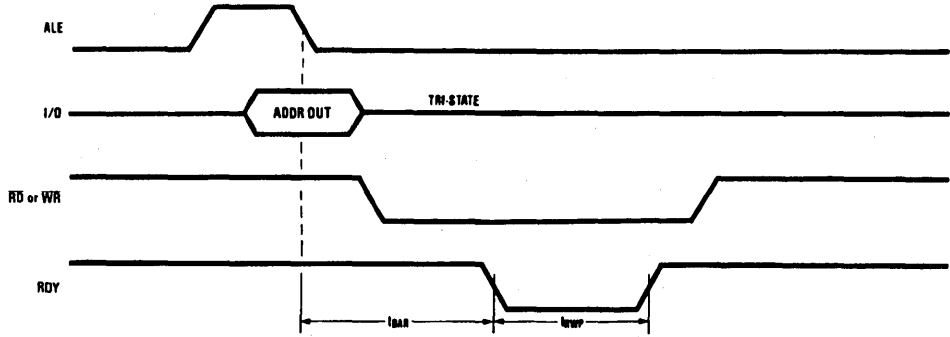


FIGURE 6. Ready Mode Timing

TL/DD/9682-5

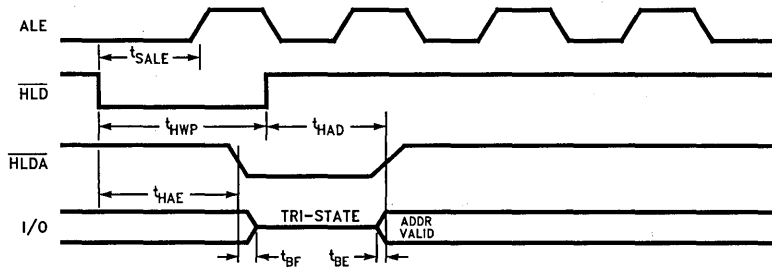


FIGURE 7. Hold Mode Timing

TL/DD/9682-6

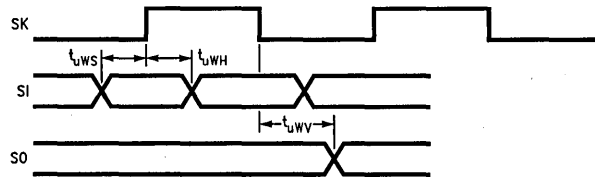


FIGURE 8. MICROWIRE Setup/Hold Timing

TL/DD/9682-39

Timing Waveforms (Continued)

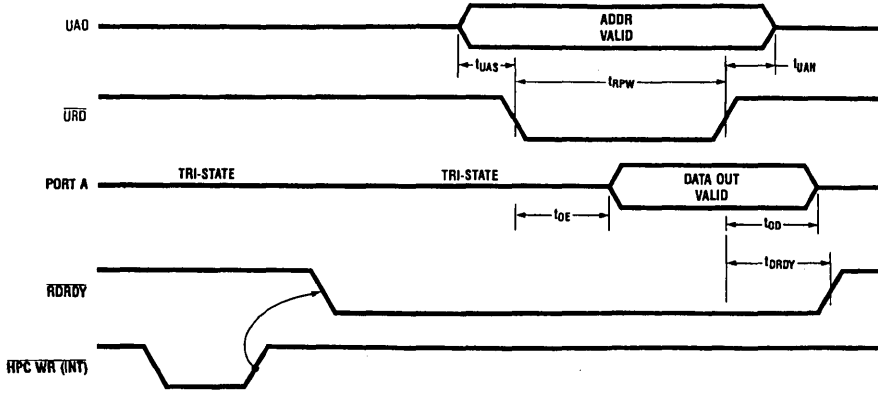


FIGURE 9. UPI Read Timing

TL/DD/9682-9

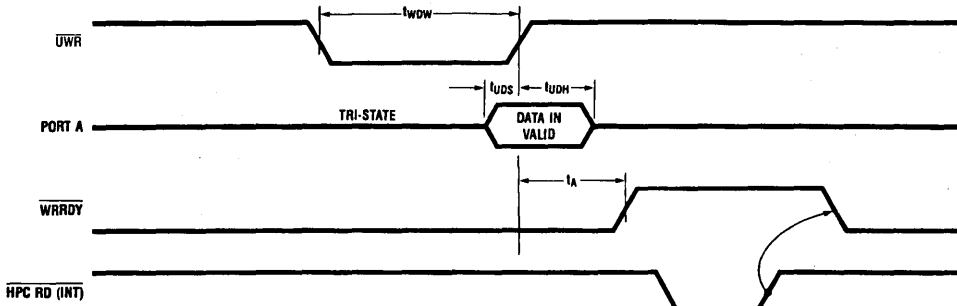


FIGURE 10. UPI Write Timing

TL/DD/9682-10

Pin Descriptions

The HPC46164 is available only in an 80-pin PQFP package.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	HLDA	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	WRDY	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	RDRDY	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	WR	Write Output
B12:	HBE	High Byte Enable Output/Input (sampled at reset)
B15:	RD	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/ \overline{URD}
I3:	INT3	Maskable Interrupt/Input Capture/ \overline{UWR}
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		External Start A/D Conversion

Port D is an 8-bit input port that can be used as general purpose digital inputs or as analog channel inputs for the A/D converter. These functions of Port D are mutually exclusive and under the control of software.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V _{CC1} and V _{CC2}	Positive Power Supply
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI	The Chip System Clock Input
CKO	The Chip System Clock Output (inversion of CKI)

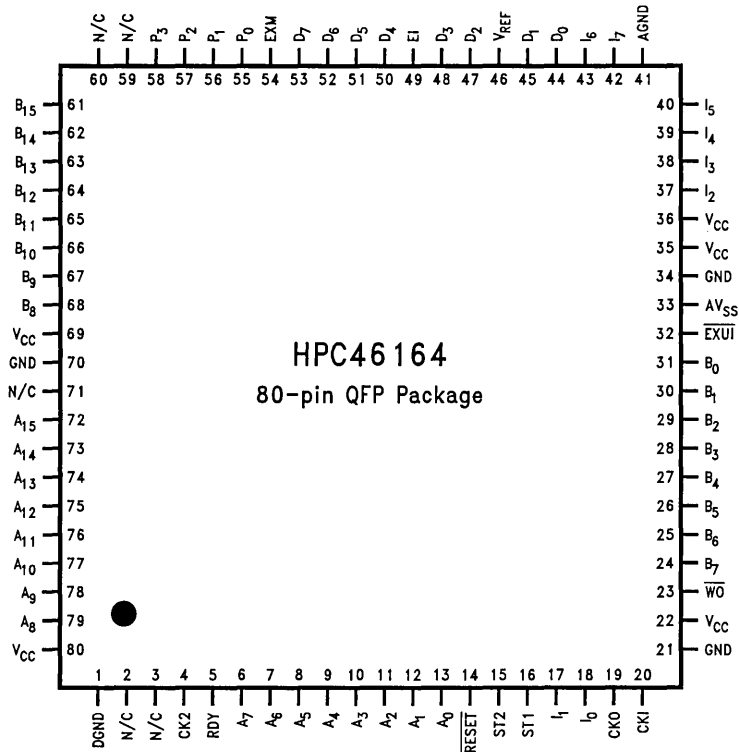
Pins CKI and CKO are usually connected across an external crystal.

CK2	Clock Output (CKI divided by 2)
-----	---------------------------------

OTHER PINS

\overline{WO}	This is an active low open drain output that signals an illegal situation has been detected by the WATCHDOG logic.
ST1	Bus Cycle Status Output: indicates first opcode fetch.
ST2	Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
\overline{RESET}	Active low input that forces the chip to restart and sets the ports in a TRI-STATE mode.
RDY/ \overline{HLD}	Selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.
V _{REF}	A/D converter reference voltage input.
EXM	External memory enable (active high) disables internal ROM and maps it to external memory.
EI	External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.
\overline{EXUI}	External active low interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2.

Connection Diagram



HPC46164
80-pin QFP Package

Top View

TL/DD/9682-45

Order Number HPC46064XXX/F20, HPC46064XXX/F30,
HPC46004VF20 or HPC46004VF30
See NS Package Number VF80B

Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 11*) consists of a data register and a direction register. Port B (see *Figures 12, 13 and 14*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.

Ports A & B (Continued)

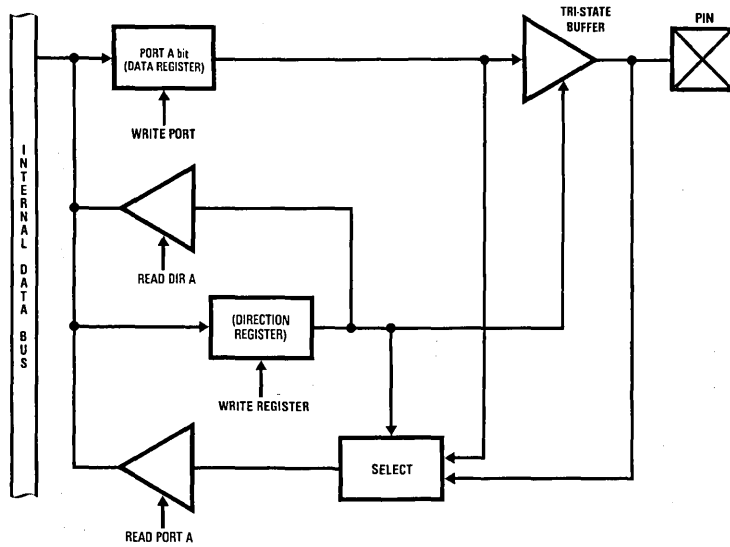


FIGURE 11. Port A: I/O Structure

TL/DD/9682-13

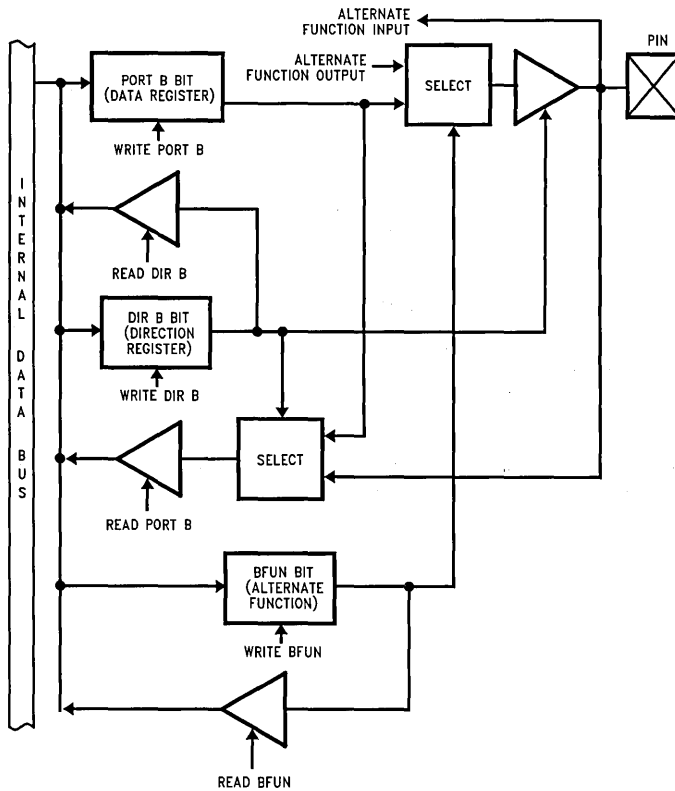
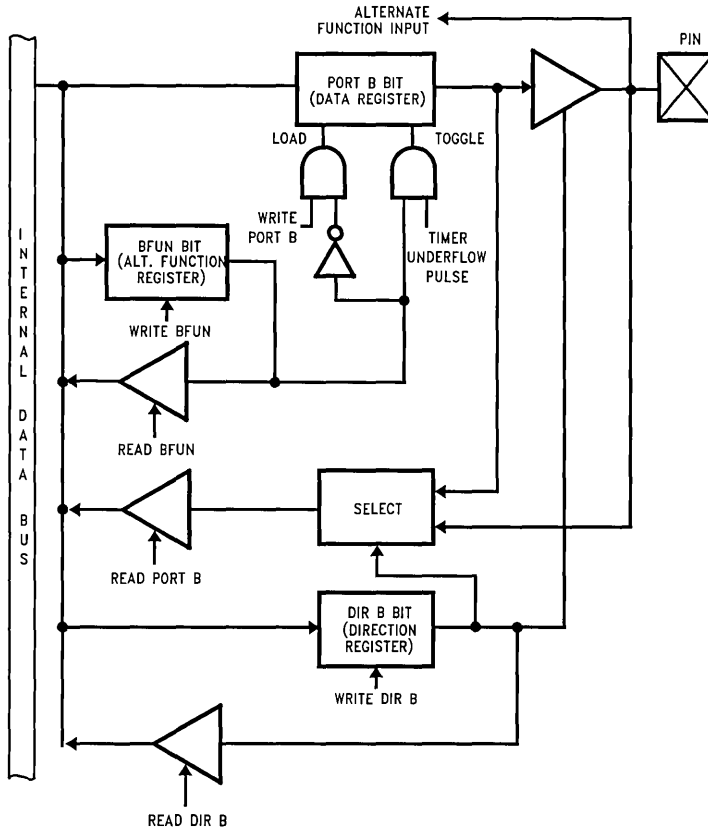


FIGURE 12. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

TL/DD/9682-14

Ports A & B (Continued)



TL/DD/9682-15

FIGURE 13. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)

Ports A & B (Continued)

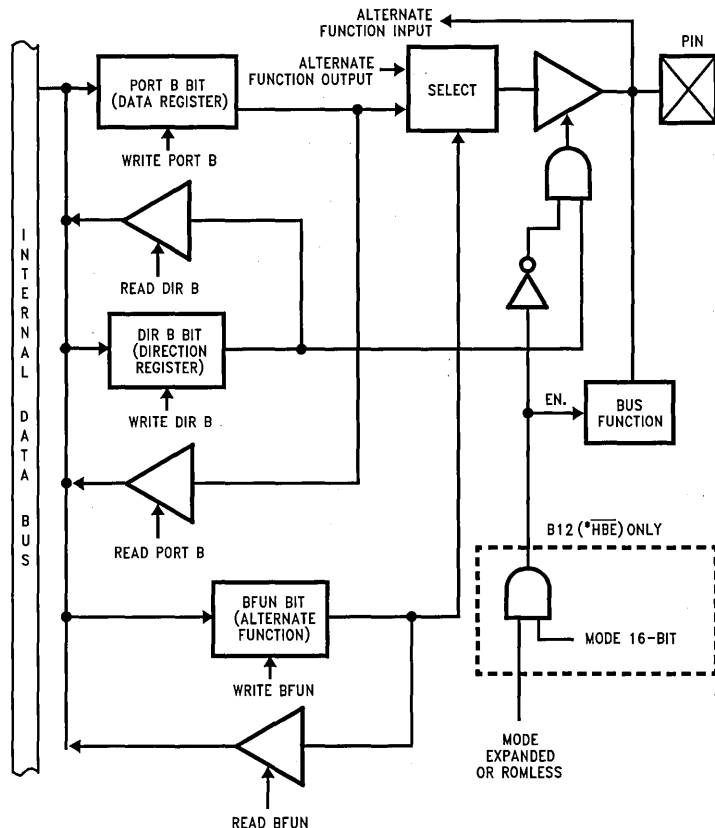


FIGURE 14. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

TL/DD/9682-16

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC46164 and HPC46104 have four operating modes. The ROMless HPC46104 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC46164 is C000 to FFFF (16k bytes). The HPC46104 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC46104 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range

and on-chip RAM and Register range, and the "illegal address detection" feature of the WATCHDOG logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the WATCHDOG logic is disabled. The EA bit should be set to "1" by software when using the HPC46104 to disable the "illegal address detection" feature of WATCHDOG.

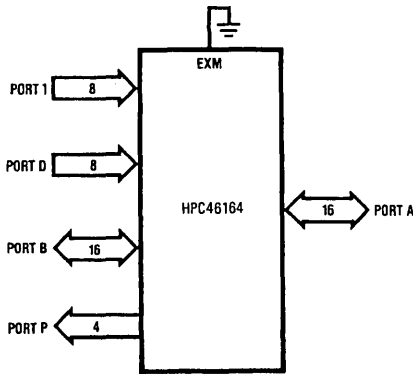
All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, \overline{RD} , \overline{WR} and \overline{HBE} . The High Byte Enable pin (\overline{HBE}) is used in 16-bit mode to select high order memory bytes. The \overline{RD} and \overline{WR} signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling \overline{HBE} high at reset. If \overline{HBE} is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC46164 and HPC46104.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC46164 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC46164 functions as a self-contained microcomputer (see Figure 15) with all memory (RAM and ROM) on-chip. It can address internal memory only, consisting of 16k bytes of ROM (C000 to FFFF) and 512 bytes of on-chip RAM and Registers (0000 to 02FF). The "illegal address detection" feature of the WATCHDOG is enabled in the Single-Chip Normal mode and a WATCHDOG Output ($\overline{W0}$) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.



TL/DD/9682-17

FIGURE 15. Single-Chip Mode

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC46164 to address external memory in addition to the

on-chip ROM and RAM (see Table I). WATCHDOG illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC46164 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 16k of external memory may be used with the HPC46164 (see Table I). The WATCHDOG circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

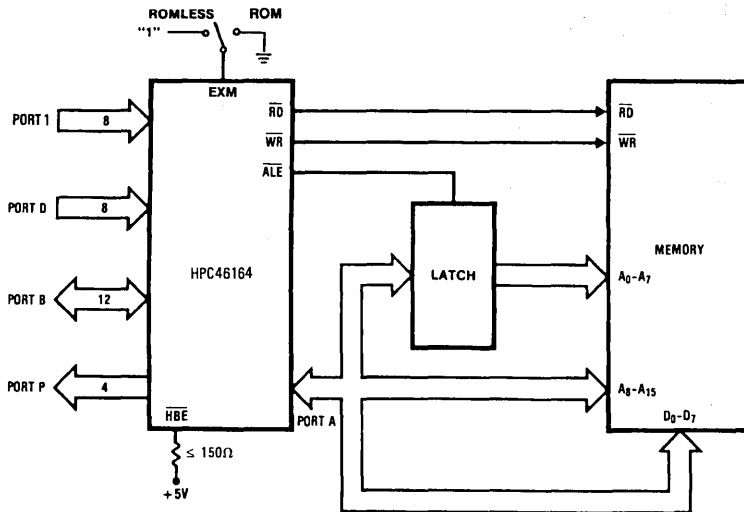
TABLE I. HPC46164 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	C000:FFFF on-chip
Expanded Normal	0	1	C000:FFFF on-chip 0300:BFFF off-chip
Single-Chip ROMless	1	0	C000:FFFF off-chip
Expanded ROMless	1	1	0300:FFFF off-chip

Note: In all operating modes, the on-chip RAM and Registers (0000:02FF) may be accessed.

EXPANDED ROMLESS MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of WATCHDOG is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.



TL/DD/9682-18

FIGURE 16. 8-Bit External Memory

HPC46164 Operating Modes (Continued)

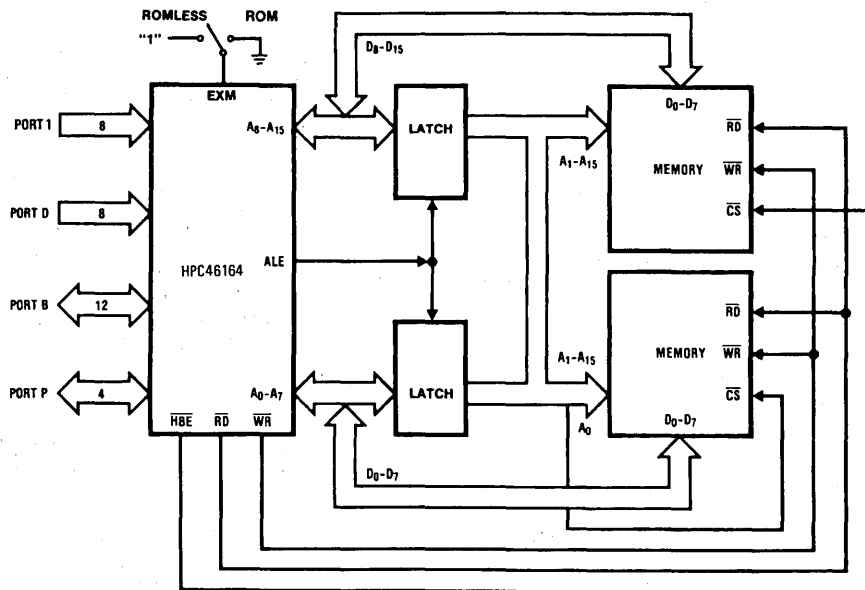


FIGURE 17. 16-Bit External Memory

TL/DD/9682-19

HPC46104 Operating Modes

EXPANDED ROMLESS MODE

Because the HPC46104 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC46104 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between C000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the WATCHDOG logic.

TABLE II. HPC46104 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0300:FFFF off-chip

Note: The on-chip RAM and Registers (0000:02FF) of the HPC46104 may be accessed at all times.

Wait States

The internal ROM can be accessed at the maximum operating frequency with one wait state. With 0 wait states, internal ROM accesses are limited to $\frac{2}{3} f_C$ max. The HPC46164 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC46164: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC46164 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC46164 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the RESET or the NMI. The RESET input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC46164 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC46164 to resume normal operation.

HPC46164 Interrupts

Complex interrupt handling is easily accomplished by the HPC46164's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table III.

TABLE III. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
FFFF:FFFE	RESET	0
FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
FFFB:FFFA	External interrupt on I2 pin	2
FFF9:FFF8	External interrupt on I3 pin	3
FFF7:FFF6	External interrupt on I4 pin	4
FFF5:FFF4	Overflow on internal timers	5
FFF3:FFF2	Internal on the UART transmit/receive complete or external on EXUI or A/D converter	6
FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC46164 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table III. The interrupt on Reset has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately. RESET and EXUI are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt (EXUI) is shared with the on-board UART. The EXUI interrupt is level-LOW-sensitive. To select this interrupt, disable the ERI and ETI UART interrupts by resetting these enable bits in the ENUI register. To select the on-board UART interrupt, leave this pin floating.

Interrupt Control Registers

The HPC46164 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled

or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC46164 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 18* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A and B in the TRI-STATE condition and Port P in the LOW state. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between C000 and FFFF when using the HPC46104.

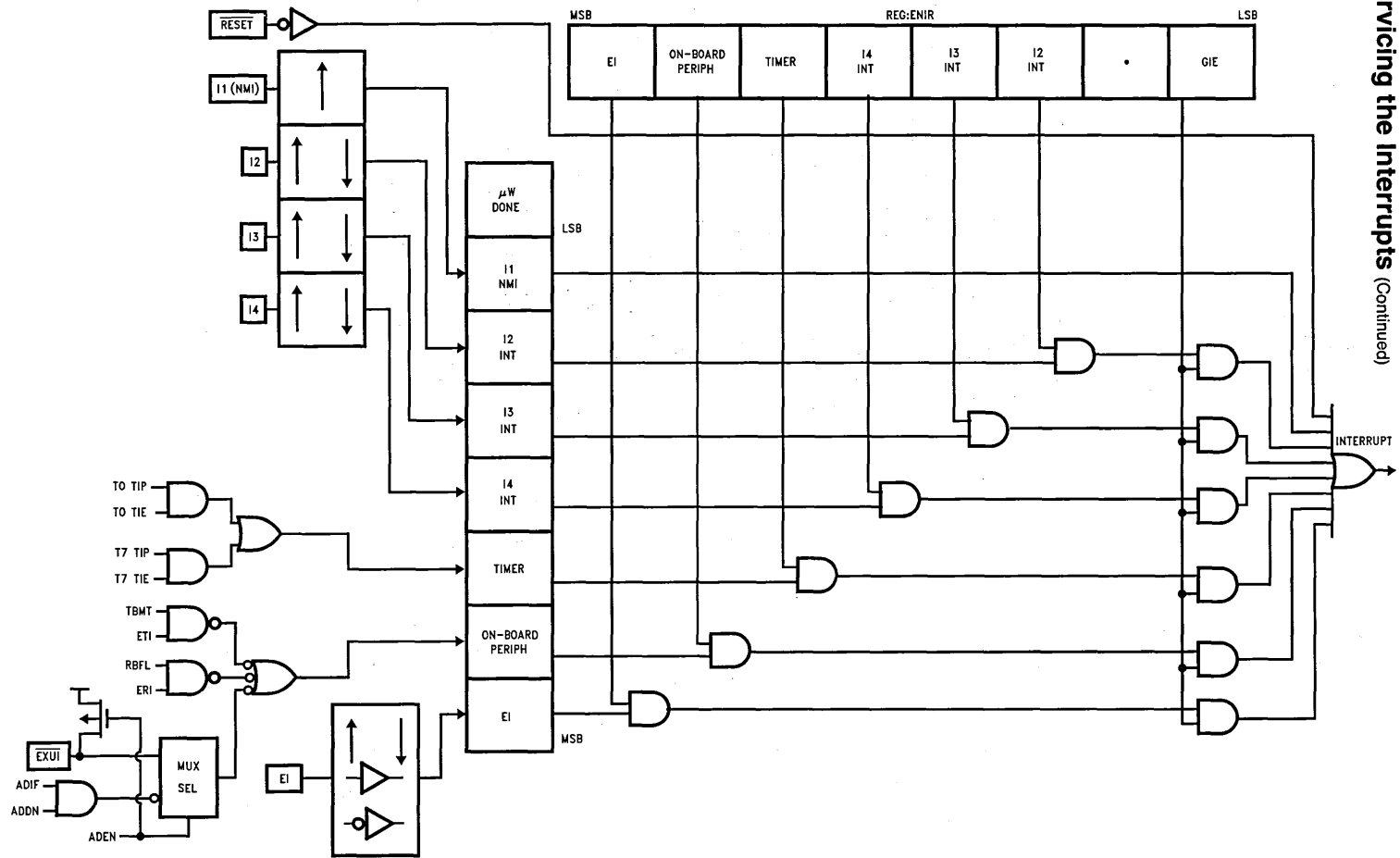


FIGURE 18. Block Diagram of Interrupt Logic

Timer Overview

The HPC46164 contains a powerful set of flexible timers enabling the HPC46164 to perform extensive timer functions not usually associated with microcontrollers. The HPC46164 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for WATCHDOG logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see Figure 19).

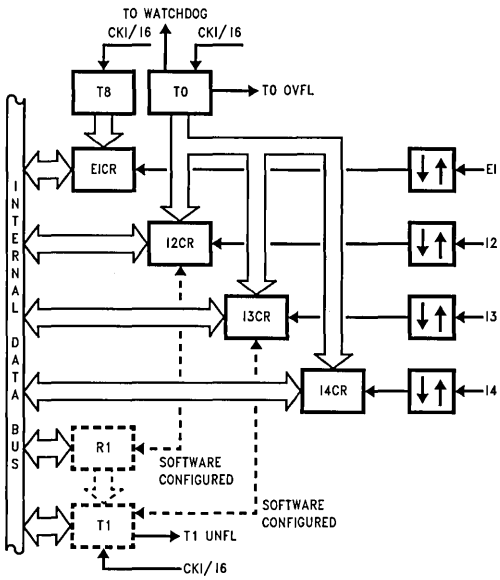


FIGURE 19. Timers T0, T1 and T8 with Four Input Capture Registers

The HPC46164 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records

the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see Figure 20).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC46164 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see Figure 20).

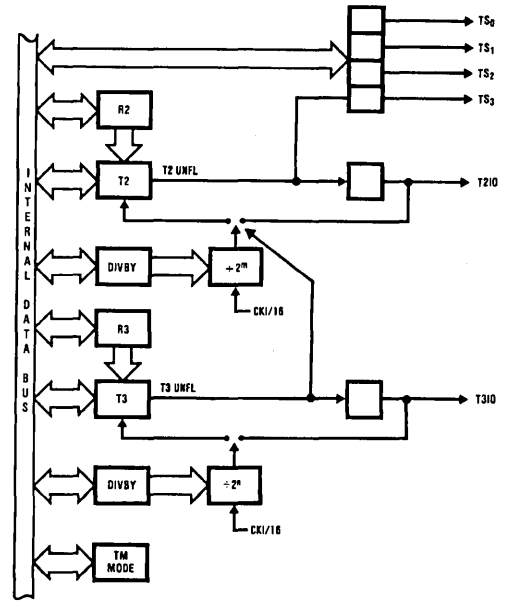


FIGURE 20. Timers T2-T3 Block

Timer Overview (Continued)

Timer/register pairs 4-7 form four identical units which can generate synchronous outputs on port P (see Figure 21). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to $\frac{1}{2}$ the frequency of the source used for clocking the timer.

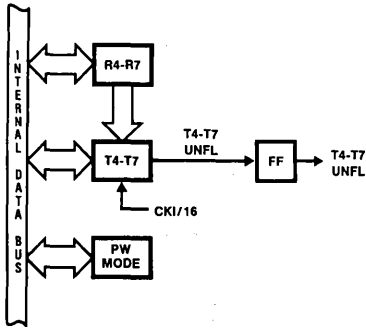


FIGURE 21. Timers T4-T7 Block

TL/DD/9682-23

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC46164.

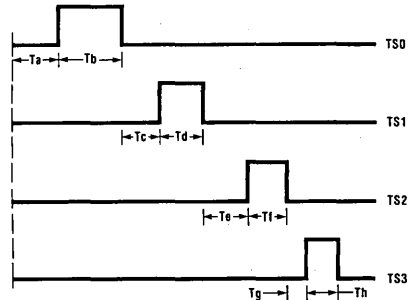
Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.



FIGURE 22. Square Wave Frequency Generation

TL/DD/9682-24

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0-TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. Figure 23 is an example of synchronous pulse train generation.



TL/DD/9682-25

FIGURE 23. Synchronous Pulse Generation

WATCHDOG Logic

The WATCHDOG Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the WATCHDOG logic are potentially infinite loops and illegal addresses. Should the WATCHDOG register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the WATCHDOG Output ($\overline{W0}$) pin low. The $\overline{W0}$ pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

*Note: See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see Figure 24). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS (Continued)

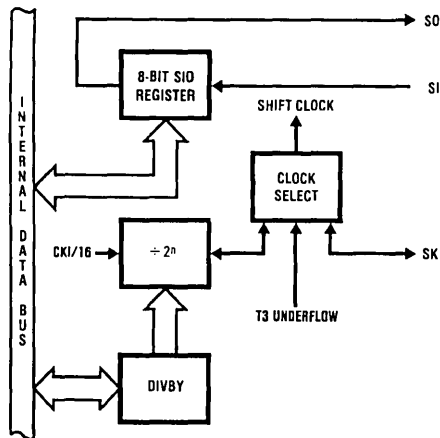


FIGURE 24. MICROWIRE/PLUS

TL/DD/9682-26

MICROWIRE/PLUS Operation

The HPC46164 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC46164 is the master or slave. The shift clock is generated when the HPC46164 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC46164 is configured as a slave. When the HPC46164 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 14 se-

lectable binary steps or T3 underflow from 153 Hz to 1.25 MHz with CKI at 20.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 25 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC46164 microcontrollers interconnected to other MICROWIRE peripherals. HPC46164 # 1 is set up as the master and initiates all data transfers. HPC46164 # 2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of an LCD display controlled by the COP472 display driver. The data to be displayed is sent serially to the COP472 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC46164 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.

MICROWIRE/PLUS Application (Continued)

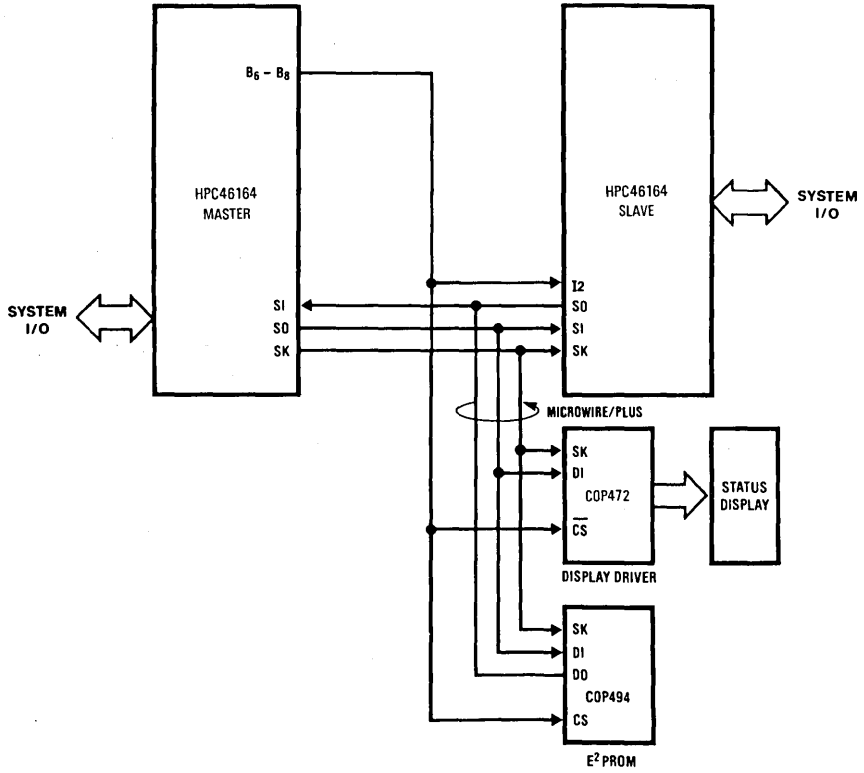


FIGURE 25. MICROWIRE/PLUS Application

TL/DD/9682-27

HPC46164 UART

The HPC46164 contains a software programmable UART. The UART (see *Figure 26*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kbaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC46164 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC46164 UART features a Wake-up Mode of operation. This mode of operation enables the HPC46164 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC46164 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

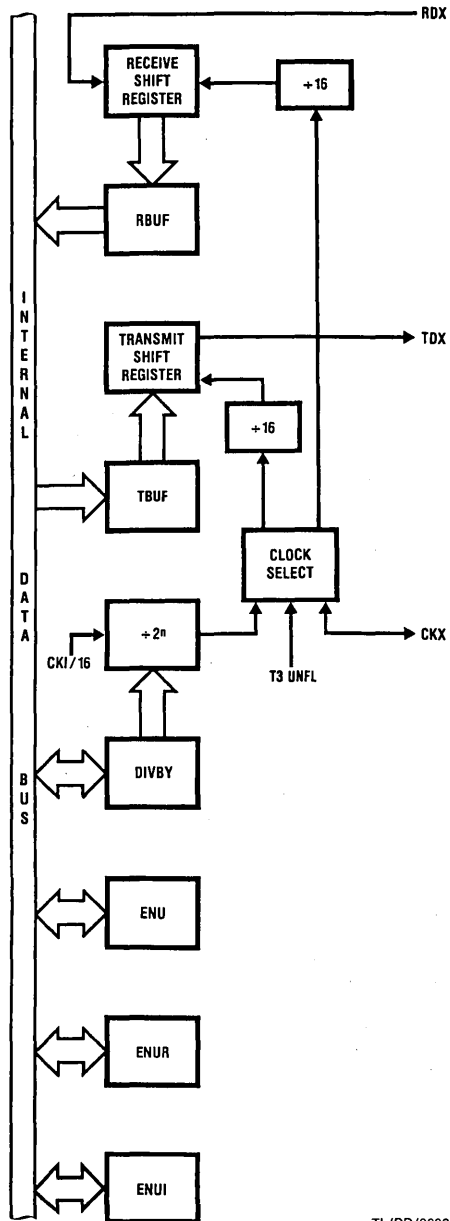


FIGURE 26. UART Block Diagram

TL/DD/9682-28

A/D Converter

The HPC46164 has an on-board eight-channel 8-bit Analog to Digital converter. Conversion is performed using a successive approximation technique. The A/D converter cell can operate in single-ended mode where the input voltage is applied across one of the eight input channels (D0–D7) and AGND or in differential mode where the input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel-pairs in differential mode.

OPERATING MODES

The operating modes of the converter are selected by 4 bits called ADMODE (CR2.4–7) see Table IV. Associated with the eight input channels in single-ended mode are eight result registers, one for each channel. The A/D converter can be programmed by software to convert on any specific channel storing the result in the result register associated with that channel. It can also be programmed to stop after one conversion or to convert continuously. If a brief history of the signal on any specific input channel is required, the converter can be programmed to convert on that channel and store the consecutive results in each of the result registers before stopping. As a final configuration in single-ended mode, the converter can be programmed to convert the signal on each input channel and store the result in its associated result register continuously.

Associated with each even-odd pair of input channels in differential mode of operation are four result register-pairs. The A/D converter performs two conversions on the selected pair of input channels. One conversion is performed assuming the positive connection is made to the even channel and the negative connection is made to the following odd channel. This result is stored in the result register associated with the even channel. Another conversion is performed assuming the positive connection is made to the odd channel and the negative connection is made to the preceding even channel. This result is stored in the result register associated with the odd channel. This technique does not require that the programmer know the polarity of the input signal. If the even channel result register is nonzero (meaning the odd channel result register is zero), then the input signal is positive with respect to the odd channel. If the odd channel result register is non-zero (meaning the even channel result register is zero), then the input signal is positive with respect to the even channel.

The same operating modes for single-ended operation also apply when the inputs are taken from channel-pairs in differential mode. The programmer can configure the A/D to con-

vert on any selected channel-pair and store the result in its associated result register-pair then stop. The A/D can also be programmed to do this continuously. Conversion can also be done on any channel-pair storing the result into four result register-pairs for a history of the differential input. Finally, all input channel-pairs can be converted continuously.

The final mode of operation suppresses the external address/data bus activity during the single conversion modes. These quiet modes of operation utilize the RDY function of the HPC Core to insert wait states in the instruction being executed in order to limit digital noise in the environment due to external bus activity when addressing external memory. The overall effect is to increase the accuracy of the A/D.

CONTROL

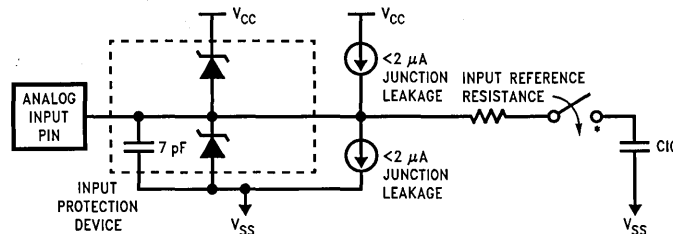
The conversion clock supplied to the A/D converter can be selected by three bits in CR1 used as a prescaler on CKI. These bits can be used to ensure that the A/D is clocked as fast as possible when different external crystal frequencies are used. Controlling the starting of conversion cycles in each of the operating modes can be done by four different methods. The method is selected by two bits called SC (CR3.0–1). Conversion cycles can be initiated through software by resetting a bit in a control register, through hardware by an underflow of Timer T2, or externally by a rising or falling edge of a signal input on I7.

INTERRUPTS

The A/D converter can interrupt the HPC when it completes a conversion cycle if one of the noncontinuous modes has been selected. If one of the cycle modes was selected, then the converter will request an interrupt after eight conversions. If one of the one-shot modes was selected, then the converter will request an interrupt after every conversion. When this interrupt is generated, the HPC vectors to the on-board peripheral interrupt vector location at address FFF2. The service routine must then determine if the A/D converter requested the interrupt by checking the A/D done flag which doubles as the A/D interrupt pending flag.

Analog Input and Source Resistance Considerations

Figure 27 shows the A/D pin model for the HPC46164 in single ended mode. The differential mode has similar A/D pin model. The leads to the analog inputs should be kept as short as possible. Both noise and digital clock coupling to an A/D input can cause conversion errors. The clock lead should be kept away from the analog input line to reduce coupling. The A/D channel input pins do not have any internal output driver circuitry connected to them because this circuitry would load the analog input signals due to output buffer leakage current.



*The analog switch is closed only during the sample time.

FIGURE 27. Port D Input Structure

A/D Converter (Continued)

TABLE IV. A/D Operating Modes

Mode 0	Single-ended, single channel, single result register, one-shot (default value on power-up)
Mode 1	Single-ended, single channel, single result register, continuous
Mode 2	Single-ended, single channel, multiple result registers, stop after 8
Mode 3	Single-ended, multiple channel, multiple result registers, continuous
Mode 4	Differential, single channel-pair, single result register-pair, one-shot
Mode 5	Differential, single channel-pair, single result register-pair, continuous
Mode 6	Differential, single channel-pair, multiple result register-pairs, stop after 4 pairs
Mode 7	Differential, multiple channel-pair, multiple result register-pairs, continuous
Mode 8	Single-ended, single channel, single result register, one-shot (default value on power-up), quiet address/data bus
Mode C	Differential, single channel-pair, single result register-pair, one-shot, quiet address/data bus

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC46164 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC46164's and set up systems with very high data exchange rates. Another area of application could be where an HPC46164 is programmed as an intelligent peripheral to a host system such as the Series 32000[®] microprocessor. Figure 28 illustrates how an HPC46164 could be used as an intelligent peripheral for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe (\overline{URD}), a Write Strobe (\overline{UWR}), a Read Ready Line (\overline{RDRDY}), a Write Ready Line (\overline{WRRDY}) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The \overline{URD} and \overline{UWR} inputs may be used to interrupt the HPC46164. The \overline{RDRDY} and \overline{WRRDY} outputs may be used to interrupt the host processor.

The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC46164 is the data bus. UPI can only be used if the HPC46164 is in the Single-Chip mode.

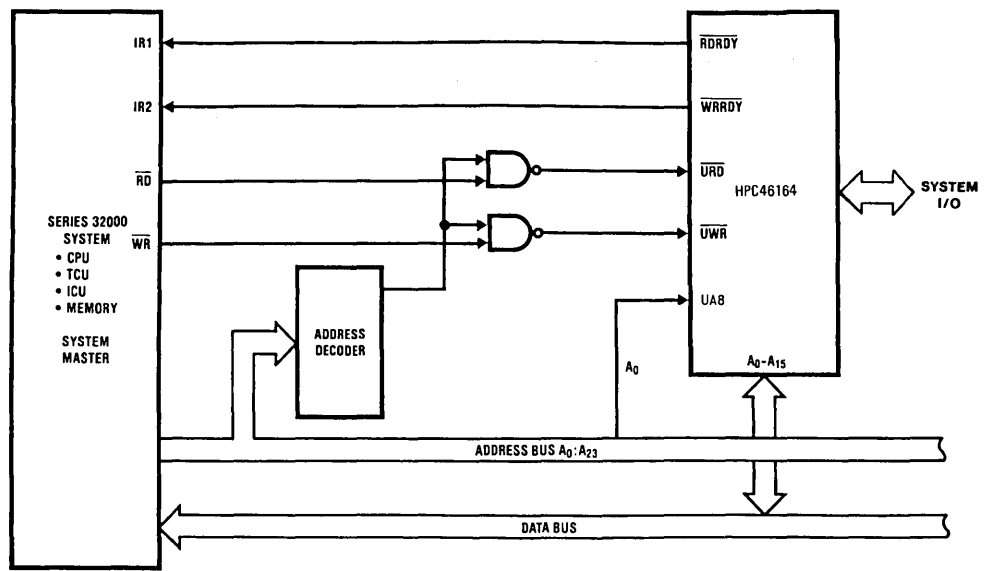


FIGURE 28. HPC46164 as a Peripheral: (UPI Interface to Series 32000 Application)

TL/DD/9682-30

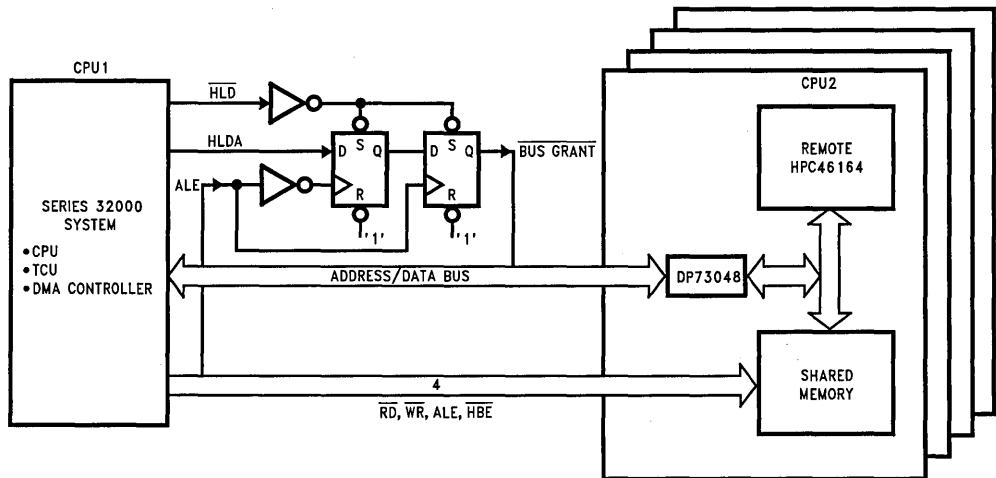
Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC46164 supports shared memory access with two pins. The pins are the RDY/HLD input pin and the HLDA output pin. The user can software select either the Hold or Ready function by the state of a control bit. The HLD output is multiplexed onto port B.

The host uses DMA to interface with the HPC46164. The host initiates a data transfer by activating the HLD input of

the HPC46164. In response, the HPC46164 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal (HLDA) from the HPC46164 indicating that the system bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC46164 resumes normal operations.

To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus. Figure 29 illustrates an application of the shared memory interface between the HPC46164 and a Series 32000 system.



TL/DD/9682-31

FIGURE 29. Shared Memory Application: HPC46164 Interface to Series 32000 System

Memory

The HPC46164 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 16 kbytes of ROM and 512 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed

directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC46164 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC46164 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table V.

TABLE V. HPC46164 Memory Map

FFFF:FFF0 FFEF:FFD0 FFCF:FFFE : : E001:C000	Interrupt Vectors JSRP Vectors } On-Chip ROM*	USER MEMORY	011F:011E 011D:011C 011B:011A 0119:0118 0117:0116 0115:0114 0113:0112 0111:0110 0106	A/D Result Register 7 A/D Result Register 6 A/D Result Register 5 A/D Result Register 4 A/D Result Register 3 A/D Result Register 2 A/D Result Register 1 A/D Result Register 0 A/D Control Register 3	A to D Registers
BFFF:BFFE : : 0301:0300	} External Expansion Memory		0104	Port D Input Register	
02FF:02FE : : 01C1:01C0	} On-Chip RAM	USER RAM	0102 0100	A/D Control Register 2 A/D Control Register 1	A to D Registers
0195:0194	WATCHDOG Address	WATCHDOG Logic	00F5:00F4 00F3:00F2 00F1:00F0	BFUN Register DIR B Register DIR A Register / IBUF	PORTS A & B CONTROL
0192 0191:0190 018F:018E 018D:018C 018B:018A 0189:0188 0187:0186 0185:0184 0183:0182 0181:0180	T0CON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/ R1 I3CR Register/ T1 I4CR Register	Timer Block T0:T3	00E6	UPIC Register	UPI CONTROL
015E:015F 015C 0153:0152 0151:0150 014F:014E 014D:014C 014B:014A 0149:0148 0147:0146 0145:0144 0143:0142 0141:0140	EICR EICON Port P Register PWMODE Register R7 Register T7 Timer R6 Register T6 Timer R5 Register T5 Timer R4 Register T4 Timer	Timer Block T4:T7	00E3:00E2 00E1:00E0	Reserved Port B Port A / OBUF	PORTS A & B
0128 0126 0124 0122 0120	ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART	00DE 00DD:00DC 00D8 00D6 00D4 00D2 00D0	Reserved HALT Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	PORT CONTROL & INTERRUPT CONTROL REGISTERS
			00CF:00CE 00CD:00CC 00CB:00CA 00C9:00C8 00C7:00C6 00C5:00C4 00C3:00C2 00C0	X Register B Register K Register A Register PC Register SP Register Reserved PSW Register	HPC CORE REGISTERS
			00BF:00BE : : 0001:0000	On-Chip RAM	USER RAM

*Note: The HPC46164 On-Chip ROM is on addresses C000:FFFF and the External Expansion Memory is 0300:BFFF. The HPC46104 have no On-Chip ROM, External Memory is 0300:FFFF.

Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused output should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least 1 μF and bypass their outputs with a 10 μF to 50 μF tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a 10 μF to 20 μF tantalum electrolytic capacitor or a 50 μF to 100 μF aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.

A recommended crystal oscillator circuit to be used with the HPC is shown in *Figure 30*. See table for recommended component values. The recommended values given in Table VI have yielded consistent results and are made to match a crystal with a 20 pF load capacitance, with some small allowance for layout capacitance.

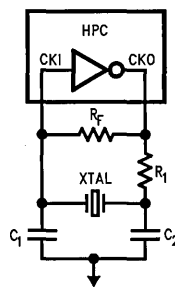
A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within "1" distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the

chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A 1.0 μF , a 0.1 μF , and a 0.001 μF dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

TABLE VI. HPC Oscillator Table

XTAL Freq (MHz)	R_1 (Ω)
≤ 2	1500
4	1200
6	910
8	750
10	600
12	470
14	390
16	300
18	220
20	180
22	150
24	120
26	100
28	75
30	62



$R_f = 3.3 \text{ M}\Omega$

$C_1 = 27 \text{ pF}$

$C_2 = 33 \text{ F}$

XTAL Specifications: The crystal used was an M-TRON Industries MP-1 Series XTAL. "AT" cut, parallel resonant

$C_L = 18 \text{ pF}$

Series Resistance is

25 Ω @ 25 MHz

40 Ω @ 10 MHz

600 Ω @ 2 MHz

TL/DD/9682-41

FIGURE 30. Recommended Crystal Circuit

HPC46164 CPU

The HPC46164 CPU has a 16-bit ALU and six 16-bit registers:

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

HPC46164 CPU (Continued)

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC46164 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	$MA + Mem1 \rightarrow MA$ carry $\rightarrow C$
ADC	Add with carry	$MA + Mem1 + C \rightarrow MA$ carry $\rightarrow C$
ADDS	Add short imm8	$A + imm8 \rightarrow A$ carry $\rightarrow C$
DADC	Decimal add with carry	$MA + Mem1 + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
SUBC	Subtract with carry	$MA - Mem1 + C \rightarrow MA$ carry $\rightarrow C$
DSUBC	Decimal subtract w/carry	$MA - Mem1 + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
MULT	Multiply (unsigned)	$MA * Mem1 \rightarrow MA \& X, 0 \rightarrow K, 0 \rightarrow C$
DIV	Divide (unsigned)	$MA / Mem1 \rightarrow MA, rem. \rightarrow X, 0 \rightarrow K, 0 \rightarrow C$
DIVD	Divide Double Word (unsigned)	$X \& MA / Mem1 \rightarrow MA, rem \rightarrow X, 0 \rightarrow K, Carry \rightarrow C$
IFEQ	If equal	Compare MA & Mem1, Do next if equal
IFGT	If greater than	Compare MA & Mem1, Do next if MA > Mem1
AND	Logical and	$MA \text{ and } Mem1 \rightarrow MA$
OR	Logical or	$MA \text{ or } Mem1 \rightarrow MA$
XOR	Logical exclusive-or	$MA \text{ xor } Mem1 \rightarrow MA$
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	$Mem + 1 \rightarrow Mem$
DECSZ	Decrement, skip if 0	$Mem - 1 \rightarrow Mem$, Skip next if Mem = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Mem1 → MA
	Load, incr/decr X	Mem(X) → A, X ± 1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ± 1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP
POP	Pop Stack to Memory	SP - 2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ± 1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A, B ± 1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B, imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A - 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP), SP + 2 → SP W(table #) → PC
JSR	Jump subroutine relative	PC → W(SP), SP + 2 → SP, PC + # → PC (# is +1025 to -1023)
JSRL	Jump subroutine long	PC → W(SP), SP + 2 → SP, PC + # → PC
JP	Jump relative short	PC + # → PC (# is +32 to -31)
JMP	Jump relative	PC + # → PC (# is +257 to -255)
JMPL	Jump relative long	PC + # → PC
JID	Jump indirect at PC + A	PC + A + 1 → PC
JIDW		then Mem(PC) + PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP - 2 → SP, W(SP) → PC
RETSK	Return then skip next	SP - 2 → SP, W(SP) → PC, & skip
RETI	Return from interrupt	SP - 2 → SP, W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory

MA is Accumulator A or direct memory (8- or 16-bit)

Mem is 8-bit byte or 16-bit word of memory

Mem1 is 8- or 16-bit memory or 8- or 16-bit immediate data

imm is 8-bit or 16-bit immediate data

imm8 is 8-bit immediate data only

Memory Usage

Number of Bytes for Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A					To Direct Memory				
	Reg Indir. (B)	(X)	Direct	Indir	Index	Immed.	Direct * **		Immed. * **	
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address

**16-bit direct address

Instructions that Modify Memory Directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B-)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X-)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC	A	1
RC	A	1
IFC	A	1
IFNC	A	1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC46164 has been designed to be extremely code-efficient. The HPC46164 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC46164, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC46164 are single-byte. There are two especially code-saving instructions: JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of the 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into this table; the assembler can give this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC46164 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC46164 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC46164 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

HPC MICROCONTROLLER DEVELOPMENT SYSTEM

The HPC microcontroller development system is an in-system emulator (ISE) designed to support the entire family of HPC Microcontrollers. The complete package of hardware and software tools combined with a host system provides a powerful system for design, development and debug of HPC based designs. Software tools are available for IBM PC-AT® (MS-DOS, PC-DOS) and for Unix based multi-user Sun SparcStation (SunOSTM).

The stand alone units comes complete with a power supply and external emulation POD. This unit can be connected to various host systems through an RS-232 link. The software package includes an ANSI compatible C-Compiler, Linker, Assembler and librarian package. Source symbolic debug capability is provided through a user friendly MS-windows 3.0 interface for IBM PC-AT environment and through a line debugger under Sunview for Sun SparcStations.

The ISE provides fully transparent in-system emulation at speeds up to 20 MHz 1 waitstate. A 2k word (48-bit wide) trace buffer gives trace trigger and non intrusive monitoring of the system. External triggering is also available through an external logic interface socket on the POD. Direct EPROM programming can be done through the use of externally mounted EPROM socket. Form-Fit-Function emulator programming is supported by a programming board included with the system. Comprehensive on-line help and diagnostics features reduced user's design and debug time. 8 hardware breakpoints (Address/range), 64 kbytes of user memory, and break on external events are some of the other features offered.

Hewlett Packard model HP64775 Emulator/Analyzer providing in-system emulation for up to 30 MHz 1 waitstate is also available. Contact your local sales office for technical details and support.

Development Support (Continued)**Development Tools Selection Table**

Product	Order Part Number	Description	Includes	Manual Number
HPC16104/ 16164	HPC-DEV-ISE4 HPC-DEV-ISE-E	HPC In-System Emulator HPC In-System Emulator for Europe and South East Asia	HPC MDS User's Manual MDS Comm User's Manual HPC Emulator Programmer HPC16104/16164 Manual	420420184-001 424420188-001 420421313-001
	HPC-DEV-IBMA	Assembler/Linker/ Library Package for IBM PC-AT	HPC Assembler/Linker Librarian User's Manual	424410836-001
	HPC-DEV-IBMC	C Compiler/Assembler/ Linker/Library Package for IBM PC-AT	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-001 424410836-001
	HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC-AT C Compiler/Assembler/Linker Library Package for IBM PC-AT	Source/Symbolic Debugger User's Manual	424420189-001
			HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-001 424410836-001
HPC-DEV-SUNC HPC-DEV-SUNDB	C Compiler/Assembler/Linker Library Package for Sun SparcStation Source/Symbolic Debugger for Sun SparcStation C Compiler/Assembler/Linker Library Package	HPC Compiler User's Manual HPC Assembler/Linker/Library User's Manual Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual		
Complete System: HPC16104/ 16164	HPC-DEV-SYS4	HPC In-System Emulator with C Compiler/Assembler/ Linker/Library and Source Symbolic Debugger		
	HPC-DEV-SYS4-E	Same for Europe and South East Asia		

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Micro-

controller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

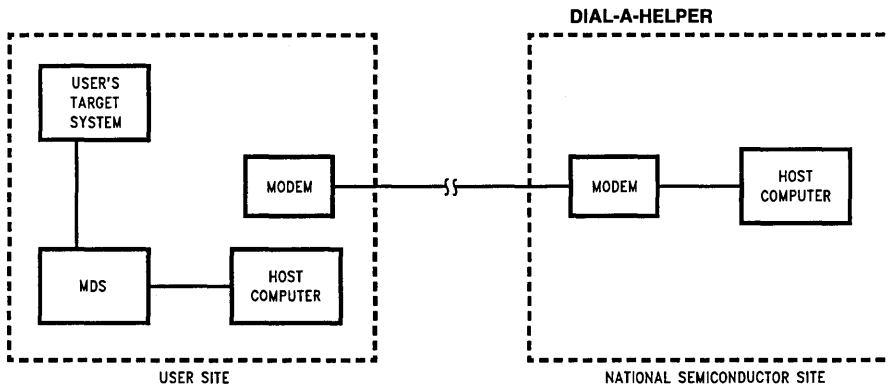
Order P/N: MDS-DIAL-A-HLP

Information System Package Contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MDS, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582
 Modem: (408) 739-1162
 Baud: 300 or 1200 baud
 Set-Up: Length: 8-Bit
 Parity: None
 Stop Bit: 1
 Operation: 24 Hrs. 7 Days

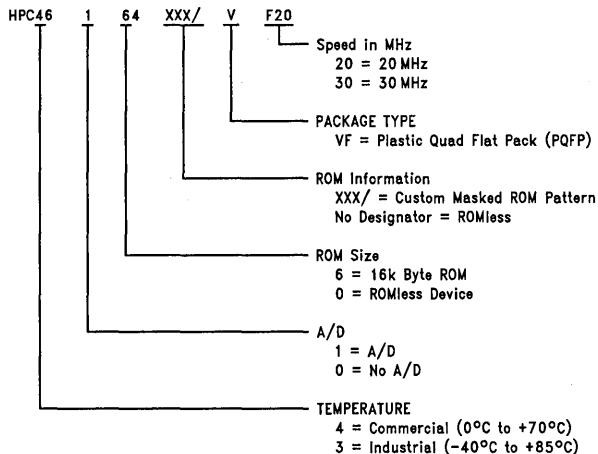


TL/DD/9682-37

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC46164 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/9682-46



HPC16064/26064/36064/46064/16004/26004/ 36004/46004 High-Performance microController

General Description

The HPC46064 and HPC46004 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC46064 has 16k bytes of on-chip ROM. The HPC46004 has no on-chip ROM and is intended for use with external memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

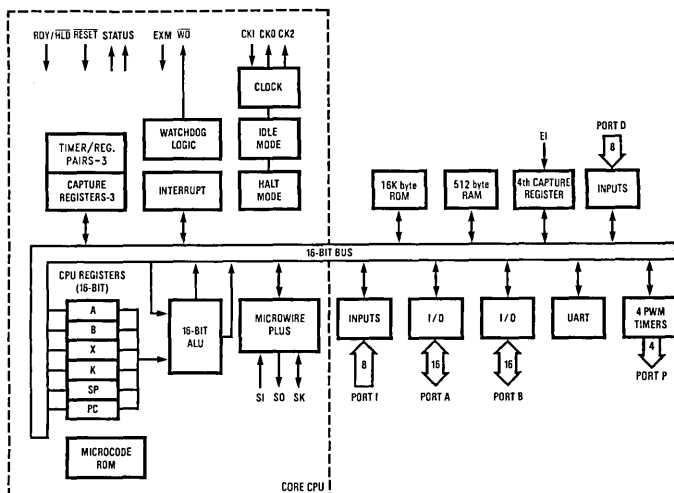
The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC46064" is used throughout this data-sheet to refer to the HPC46064 and HPC46004 devices unless otherwise specified.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LDCC, PGA and 80-pin PQFP package.

Features

- HPC family—core features:
 - 16-bit architecture, both byte and word
 - 16-bit data bus, ALU, and registers
 - 64k bytes of external direct memory addressing
 - FAST—200 ns for fastest instruction when using 20.0 MHz clock, 134 ns at 30.0 MHz
 - High code efficiency—most instructions are single byte
 - 16 x 16 multiply and 32 x 16 divide
 - Eight vectored interrupt sources
 - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
 - MICROWIRE/PLUS serial I/O interface
 - CMOS—very low power with two power save modes: IDLE and HALT
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 16k bytes of ROM, 512 bytes of RAM on-chip
- ROMless version available (HPC46004)
- Commercial (0°C to +70°C), industrial (-40°C to +85°C), automotive (-40°C to +105°C) and military (-55°C to +125°C) temperature ranges

Block Diagram (HPC46064 with 16k ROM shown)



TL/DD/11372-1

HPC16064/26064/36064/46064/16004/26004/36004/46004

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

V_{CC} with Respect to GND -0.5V to 7.0V
 All Other Pins ($V_{CC} + 0.5$)V to (GND - 0.5)V
 Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

$V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46064/46004, -40°C to $+85^\circ\text{C}$ for HPC36064/36004, -40°C to $+105^\circ\text{C}$ for HPC26064/26004, -55°C to $+125^\circ\text{C}$ for HPC16064/16004

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC1}	Supply Current	$V_{CC} = 5.5V, f_{in} = 30\text{ MHz}$ (Note 1)		65	mA
		$V_{CC} = 5.5V, f_{in} = 20\text{ MHz}$ (Note 1)		47	mA
		$V_{CC} = 5.5V, f_{in} = 2.0\text{ MHz}$ (Note 1)		10	mA
I_{CC2}	IDLE Mode Current	$V_{CC} = 5.5V, f_{in} = 30\text{ MHz}$ (Note 1)		5	mA
		$V_{CC} = 5.5V, f_{in} = 20\text{ MHz}$ (Note 1)		3.0	mA
		$V_{CC} = 5.5V, f_{in} = 2.0\text{ MHz}$ (Note 1)		1	mA
I_{CC3}	HALT Mode Current	$V_{CC} = 5.5V, f_{in} = 0\text{ kHz}$ (Note 1)		300	μA
		$V_{CC} = 2.5V, f_{in} = 0\text{ kHz}$ (Note 1)		100	μA

INPUT VOLTAGE LEVELS FOR SCHMITT TRIGGERED INPUTS, RESET, NMI, AND $\overline{W0}$; AND ALSO CKI

V_{IH1}	Logic High		$0.9 V_{CC}$		V
V_{IL1}	Logic Low			$0.1 V_{CC}$	V

ALL OTHER INPUTS

V_{IH2}	Logic High		$0.7 V_{CC}$		V
V_{IL2}	Logic Low			$0.2 V_{CC}$	V
I_{L11}	Input Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 2	μA
I_{L12}	Input Leakage Current RDY/HLD, EXUI	$V_{IN} = 0$	-3	-50	μA
I_{L13}	Input Leakage Current B12	$\overline{\text{RESET}} = 0, V_{IN} = V_{CC}$	0.5	7	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF

OUTPUT VOLTAGE LEVELS

V_{OH1}	Logic High (CMOS)	$I_{OH} = -10\ \mu\text{A}$ (Note 2)	$V_{CC} - 0.1$		V
V_{OL1}	Logic Low (CMOS)	$I_{OH} = 10\ \mu\text{A}$ (Note 2)		0.1	V
V_{OH2}	Port A/B Drive, CK2 (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅)	$I_{OH} = -7\text{ mA}$	2.4		V
V_{OL2}		$I_{OL} = 3\text{ mA}$		0.4	V
V_{OH3}	Other Port Pin Drive, $\overline{W0}$ (open drain) (B ₀ -B ₉ , B ₁₃ , B ₁₄ , P ₀ -P ₃)	$I_{OH} = -1.6\text{ mA}$ (except $\overline{W0}$)	2.4		V
V_{OL3}		$I_{OL} = 0.5\text{ mA}$		0.4	V
V_{OH4}	ST1 and ST2 Drive	$I_{OH} = -6\text{ mA}$	2.4		V
V_{OL4}		$I_{OL} = 1.6\text{ mA}$		0.4	V
V_{OH5}	Port A/B Drive (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅) When Used as External Address/Data Bus	$I_{OH} = -1\text{ mA}$	2.4		V
V_{OL5}		$I_{OL} = 3\text{ mA}$		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE® Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 5	μA

Note 1: I_{CC1} , I_{CC2} , I_{CC3} measured with no external drive (I_{OH} and $I_{OL} = 0$, I_{IH} and $I_{IL} = 0$). I_{CC1} is measured with $\overline{\text{RESET}} = V_{SS}$. I_{CC3} is measured with $\text{NMI} = V_{CC}$. CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

20 MHz
AC Electrical Characteristics

 (See Notes 1 and 4 and Figure 1 through Figure 5). $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46064/46004, -40°C to $+85^\circ\text{C}$ for HPC36064/36004, -40°C to $+105^\circ\text{C}$ for HPC26064/26004, -55°C to $+125^\circ\text{C}$ for HPC16064/16004

	Symbol and Formula	Parameter and Notes	Min	Max	Units	Notes
Clocks	f_C	CKI Operating Frequency	2	20	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	50	500	ns	
	t_{CKIH}	CKI High Time	22.5		ns	
	t_{CKIL}	CKI Low Time	22.5		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	100		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	100		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CKI Falling Edge	0	55	ns	(Note 1)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CKI Falling Edge	0	55	ns	(Note 1)
	$f_U = f_C/8$ f_{MW}	External UART Clock Input Frequency External MICROWIRE/PLUS Clock Input Frequency		2.5* 1.25	MHz MHz	
Timers	$f_{XIN} = f_C/22$ $t_{XIN} = t_C$	External Timer Input Frequency Pulse Width for Timer Inputs	100	0.91	MHz ns	
	MICROWIRE/PLUS	t_{UWS}	MICROWIRE Setup Time Master Slave	100 20		ns
t_{UWH}		MICROWIRE Hold Time Master Slave	20 50		ns	
t_{UWV}		MICROWIRE Output Valid Time Master Slave		50 150	ns	
External Hold	$t_{SALE} = \frac{3}{4}t_C + 40$	\overline{HLD} Falling Edge before \overline{ALE} Rising Edge	115		ns	
	$t_{HWP} = t_C + 10$	\overline{HLD} Pulse Width	110		ns	
	$t_{HAE} = t_C + 100$	$\overline{HLD\bar{A}}$ Falling Edge after \overline{HLD} Falling Edge		200	ns	(Note 3)
	$t_{HAD} = \frac{3}{4}t_C + 85$	$\overline{HLD\bar{A}}$ Rising Edge after \overline{HLD} Rising Edge		160	ns	
	$t_{BF} = \frac{1}{2}t_C + 66$	Bus Float after $\overline{HLD\bar{A}}$ Falling Edge		116	ns	(Note 5)
	$t_{BE} = \frac{1}{2}t_C + 66$	Bus Enable after $\overline{HLD\bar{A}}$ Rising Edge	116		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	(Note 6)
	t_{DRDY}	\overline{RDRDY} Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH}	Input Data Hold after Rising Edge of \overline{UWR}	20		ns	
t_A	\overline{WRRDY} Delay from Rising Edge of \overline{UWR}		70	ns		

*This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

20 MHz (Continued)

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 through Figure 5). $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46064/46004, $-40^\circ C$ to $+85^\circ C$ for HPC36064/36004, $-40^\circ C$ to $+105^\circ C$ for HPC26064/26004, $-55^\circ C$ to $+125^\circ C$ for HPC16064/16004

	Symbol and Formula	Parameter and Notes	Min	Max	Units	Notes
Address Cycles	$t_{DC1ALER}$	Delay from CK1 Rising Edge to ALE Rising Edge	0	35	ns	(Notes 1, 2)
	$t_{DC1ALEF}$	Delay from CK1 Rising Edge to ALE Falling Edge	0	35	ns	(Notes 1, 2)
	$t_{DC2ALER} = \frac{1}{4} t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		45	ns	(Note 2)
	$t_{DC2ALEF} = \frac{1}{4} t_C + 20$	Delay from CK2 Falling Edge to ALE Falling Edge		45	ns	(Note 2)
	$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns	
	$t_{ST} = \frac{1}{4} t_C - 7$	Setup of Address Valid before ALE Falling Edge	18		ns	
	$t_{VP} = \frac{1}{4} t_C - 5$	Hold of Address Valid after ALE Falling Edge	20		ns	
Read Cycles	$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	20		ns	
	$t_{ACC} = t_C + WS - 55$	Data Input Valid after Address Output Valid		145	ns	(Note 6)
	$t_{RD} = \frac{1}{2} t_C + WS - 65$	Data Input Valid after \overline{RD} Falling Edge		85	ns	
	$t_{RW} = \frac{1}{2} t_C + WS - 10$	\overline{RD} Pulse Width	140		ns	
	$t_{DR} = \frac{3}{4} t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	60	ns	
	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	85		ns	
Write Cycles	$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	45		ns	
	$t_{WW} = \frac{3}{4} t_C + WS - 15$	\overline{WR} Pulse Width	160		ns	
	$t_V = \frac{1}{2} t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	145		ns	
	$t_{HW} = \frac{1}{4} t_C - 5$	Hold of Data Valid after \overline{WR} Rising Edge	20		ns	
Ready Input	$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns	
	$t_{RWP} = t_C$	RDY Pulse Width	100		ns	

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CK1 having 50% duty cycle and with less than 15 pF load on CK0 with rise and fall times (t_{CK1R} and t_{CK1L}) on CK1 input less than 2.5 ns.

Note 2: Do not design with these parameters unless CK1 is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CK1 or CK0 is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{FLD} falling edge occurring at the latest time it can be accepted during the present CPU cycle being executed. If \overline{FLD} falling edge occurs later, t_{HAE} may be as long as $(3 t_C + 4WS + 72 t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: $WS (t_{WAIT}) \times$ (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 20$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

Note 6: This is guaranteed by design and not tested.

30 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and *Figure 1* through *Figure 5*.) $V_{CC} = 5V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46064/46004, -40°C to $+85^\circ\text{C}$ for HPC36064/36004, -40°C to $+105^\circ\text{C}$ for HPC26064/26004, -55°C to $+125^\circ\text{C}$ for HPC16064/16004

	Symbol and Formula	Parameter and Notes	Min	Max	Units	Notes
Clocks	f_C	CKI Operating Frequency	2	30	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	33	500	ns	
	t_{CKIH}	CKI High Time	15		ns	
	t_{CKIL}	CKI Low Time	16.6		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	66		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	66		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CKI Falling Edge	0	55	ns	(Note 1)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CKI Falling Edge	0	55	ns	(Note 1)
	$f_U = f_C/8$ f_{MW}	External UART Clock Input Frequency External MICROWIRE/PLUS Clock Input Frequency		3.75* 1.875	MHz MHz	
Timers	$f_{XIN} = f_C/22$ $t_{XIN} = t_C$	External Timer Input Frequency Pulse Width for Timer Inputs	66	1.36	MHz ns	
	MICROWIRE/PLUS	t_{UWS}	MICROWIRE Setup Time Master Slave	100 20		ns
t_{UWH}		MICROWIRE Hold Time Master Slave	20 50		ns	
t_{UVV}		MICROWIRE Output Valid Time Master Slave		50 150	ns	
External Hold	$t_{SALE} = \frac{3}{4} t_C + 40$	\overline{HLD} Falling Edge before \overline{ALE} Rising Edge	90		ns	
	$t_{HWP} = t_C + 10$	\overline{HLD} Pulse Width	76		ns	
	$t_{HAE} = t_C + 100$	\overline{HLDA} Falling Edge after \overline{HLD} Falling Edge		151	ns	(Note 3)
	$t_{HAD} = \frac{3}{4} t_C + 85$	\overline{HLDA} Rising Edge after \overline{HLD} Rising Edge		135	ns	
	$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after \overline{HLDA} Falling Edge		99	ns	(Note 5)
	$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable after \overline{HLDA} Rising Edge	99		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	(Note 6)
	t_{DRDY}	\overline{RDRDY} Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH}	Input Data Hold after Rising Edge of \overline{UWR}	20		ns	
	t_A	\overline{WRDY} Delay from Rising Edge of \overline{UWR}		70	ns	

*This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

30 MHz (Continued)

AC Electrical Characteristics

(See Notes 1 and 4 and Figure 1 through Figure 5.) $V_{CC} = 5V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46064/46004, -40°C to $+85^\circ\text{C}$ for HPC36064/36004, -40°C to $+105^\circ\text{C}$ for HPC26064/26004, -55°C to $+125^\circ\text{C}$ for HPC16064/16004

	Symbol and Formula	Parameter and Notes	Min	Max	Units	Notes
Address Cycles	$t_{DC1ALER}$	Delay from CK1 Rising Edge to ALE Rising Edge	0	35	ns	(Notes 1, 2)
	$t_{DC1ALEF}$	Delay from CK1 Rising Edge to ALE Falling Edge	0	35	ns	(Notes 1, 2)
	$t_{DC2ALER} = \frac{1}{4}t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		37	ns	(Note 2)
	$t_{DC2ALEF} = \frac{1}{4}t_C + 20$	Delay from CK2 Falling Edge to ALE Falling Edge		37	ns	(Note 2)
	$t_{LL} = \frac{1}{2}t_C - 9$	ALE Pulse Width	24		ns	
	$t_{ST} = \frac{1}{4}t_C - 7$	Setup of Address Valid before ALE Falling Edge	9		ns	
	$t_{VP} = \frac{1}{4}t_C - 5$	Hold of Address Valid after ALE Falling Edge	11		ns	
Read Cycles	$t_{ARR} = \frac{1}{4}t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	11		ns	
	$t_{ACC} = t_C + WS - 32$	Data Input Valid after Address Output Valid		100	ns	(Note 6)
	$t_{RD} = \frac{1}{2}t_C + WS - 39$	Data Input Valid after \overline{RD} Falling Edge		60	ns	
	$t_{RW} = \frac{1}{2}t_C + WS - 14$	\overline{RD} Pulse Width	85		ns	
	$t_{DR} = \frac{3}{4}t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	35	ns	
	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	51		ns	
Write Cycles	$t_{ARW} = \frac{1}{2}t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	28		ns	
	$t_{WW} = \frac{3}{4}t_C + WS - 15$	\overline{WR} Pulse Width	101		ns	
	$t_V = \frac{1}{2}t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	94		ns	
	$t_{HW} = \frac{1}{4}t_C - 10$	Hold of Data Valid after \overline{WR} Rising Edge	7		ns	
Ready Input	$t_{DAR} = \frac{1}{4}t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		33	ns	
	$t_{RWP} = t_C$	RDY Pulse Width	66		ns	

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CK1 having 50% duty cycle and with less than 15 pF load on CKO with rise and fall times (t_{CK1R} and t_{CK1F}) on CK1 input less than 2.5 ns.

Note 2: Do not design with these parameters unless CK1 is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CK1 or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{HLD} falling edge occurring at the latest time it can be accepted during the present CPU cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} may be as long as $(3t_C + 4WS + 72t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: $WS (t_{WAIT}) \times$ (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 20$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

Note 6: This is guaranteed by design and not tested.

CKI Input Signal Characteristics

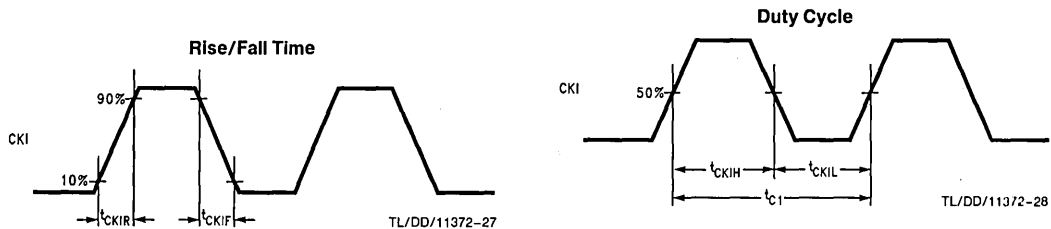
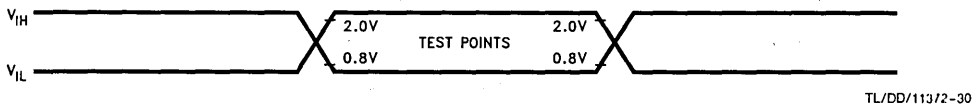


FIGURE 1. CKI Input Signal



Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

FIGURE 2. Input and Output for AC Tests

Timing Waveforms

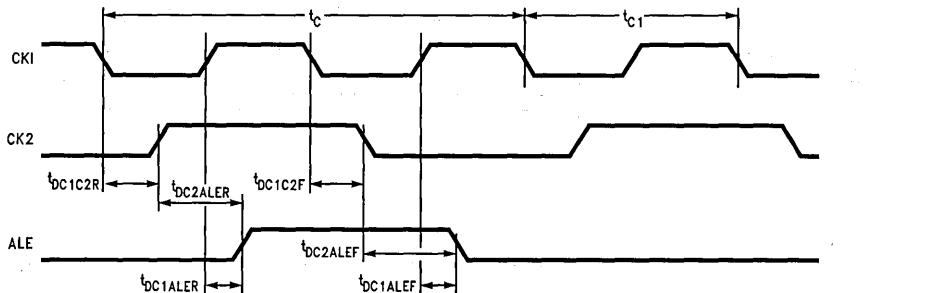


FIGURE 3. CKI, CK2, ALE Timing Diagram

Timing Waveforms (Continued)

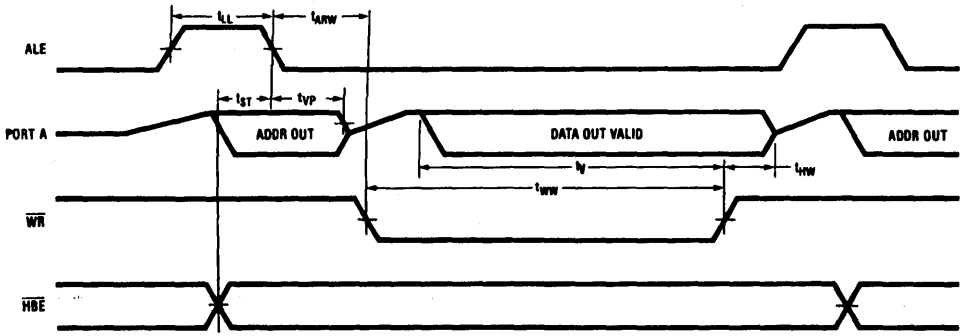


FIGURE 4. Write Cycle

TL/DD/11372-3

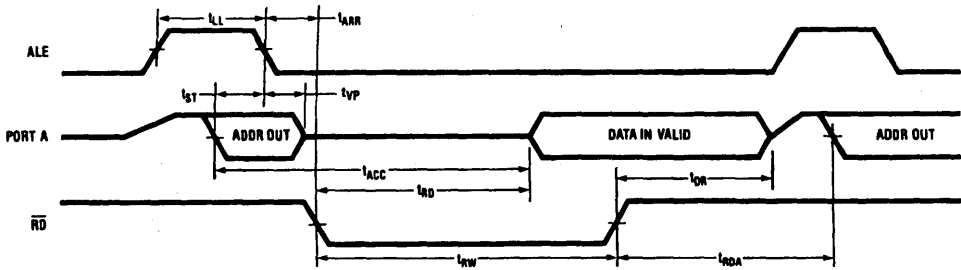


FIGURE 5. Read Cycle

TL/DD/11372-4

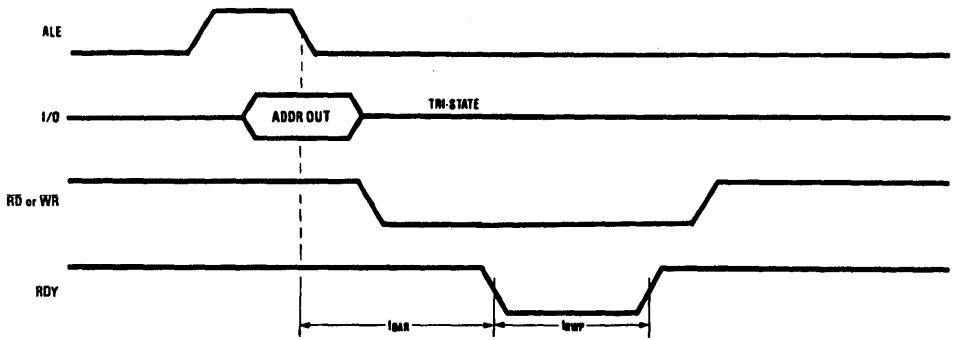


FIGURE 6. Ready Mode Timing

TL/DD/11372-5

Timing Waveforms (Continued)

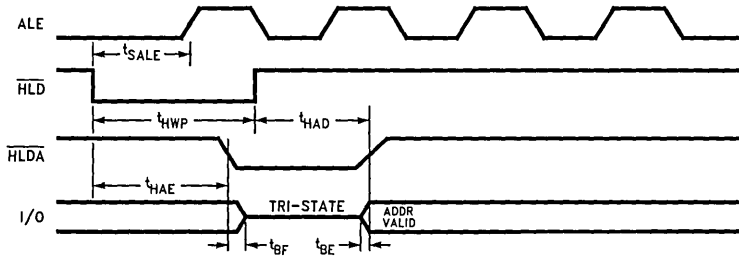


FIGURE 7. Hold Mode Timing

TL/DD/11372-6

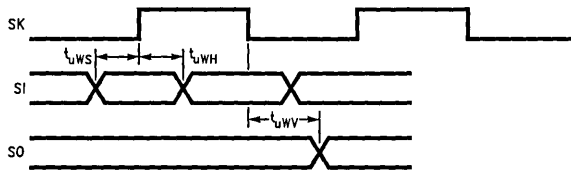


FIGURE 8. MICROWIRE Setup/Hold Timing

TL/DD/11372-29

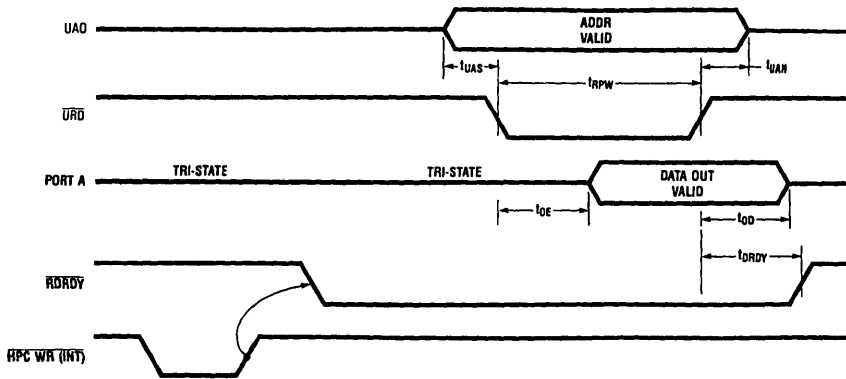


FIGURE 9. UPI Read Timing

TL/DD/11372-7

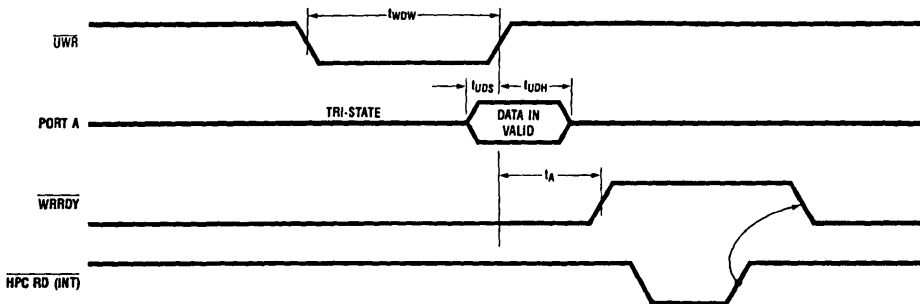


FIGURE 10. UPI Write Timing

TL/DD/11372-8

Pin Descriptions

The HPC46064 is available only in 68-pin PLCC, LDCC, PGA, and 80-pin PQFP packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	H \overline{LDA}	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	W \overline{RRDY}	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	R \overline{DRDY}	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	W \overline{R}	Write Output
B12:	H \overline{BE}	High Byte Enable Output/Input (sampled at reset)
B15:	R \overline{D}	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/ \overline{URD}
I3:	INT3	Maskable Interrupt/Input Capture/ \overline{UWR}
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V $\overline{CC1}$ and V $\overline{CC2}$	Positive Power Supply
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: There are two electrically connected V \overline{CC} pins on the chip, GND and DGND are electrically isolated. Both V \overline{CC} pins and both ground pins must be used.

CLOCK PINS

CKI	The Chip System Clock Input
CKO	The Chip System Clock Output (inversion of CKI)

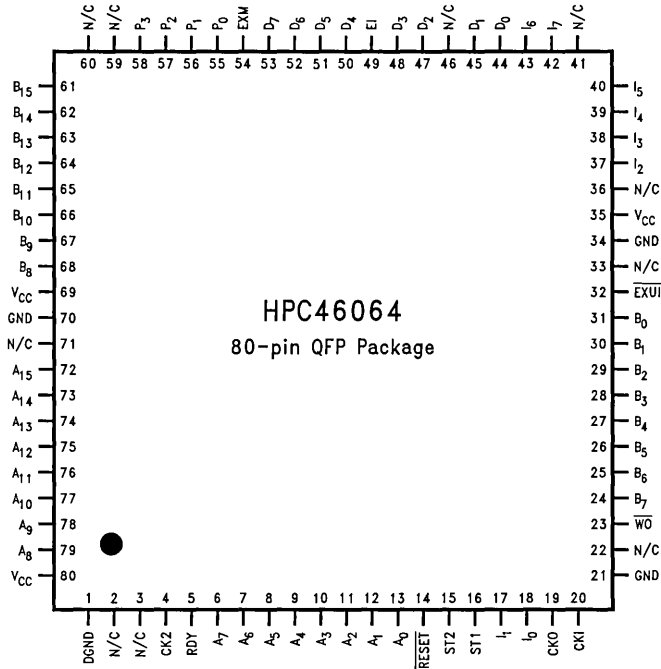
Pins CKI and CKO are usually connected across an external crystal.

CK2	Clock Output (CKI divided by 2)
-----	---------------------------------

OTHER PINS

W \overline{O}	This is an active low open drain output that signals an illegal situation has been detected by the WATCHDOG logic.
ST1	Bus Cycle Status Output: indicates first opcode fetch.
ST2	Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
R \overline{ESET}	Active low input that forces the chip to restart and sets the ports in a TRI-STATE mode.
R $\overline{DY}/\overline{HLD}$	Selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.
N/C	(No connection) do not connect anything to this pin.
EXM	External memory enable (active high) disables internal ROM and maps it to external memory.
EI	External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.
EXUI	External active low interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2.

Connection Diagrams

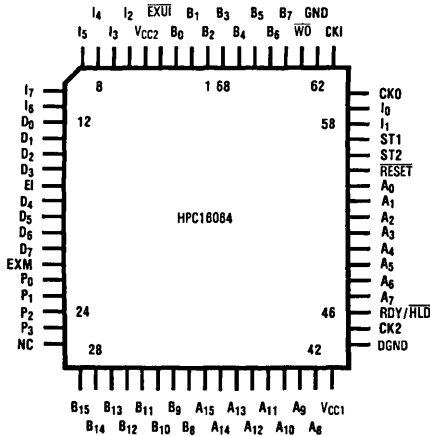


Top View

TL/DD/11372-32

Order Number HPC46064XXX/F20, HPC46064XXX/F30, HPC46004VF20 or HPC46004VF30
See NS Package Number VF80B

Plastic and Ceramic Leaded Chip Carriers



Top View

TL/DD/11372-33

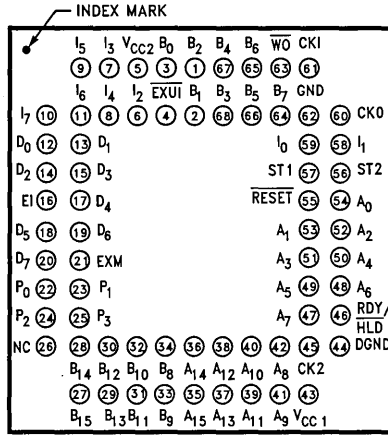
Order Number HPC16064XXX/L20, HPC16064XXX/L30, HPC16004EL20 or HPC16004EL30
See NS Package Number EL68A

Order Number HPC16064XXX/V20, HPC26064XXX/V20, HPC36064XXX/V20, HPC46064XXX/V20,
HPC16064XXX/V30, HPC26064XXX/V30, HPC360864XXX/V30, HPC16004V20, HPC26004V20,
HPC36004V20, HPC46003V20, HPC16004V30, HPV26004V30, HPC36004V30 or HPC46004V30
See NS Package Number V68A

Note: XXX designates the unique ROM code of a masked device.

Connection Diagrams (Continued)

Pin Grid Array Pinout



TL/DD/11372-34

Top View

(looking down on component side of PC Board)

Order Number HPC16064XXX/U20, HPC16064XXX/U30, HPC16004U20 or HPC16004U30
See NS Package Number U68A

Note: XXX designates the unique ROM code of a masked device.

Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 11*) consists of a data register and a direction register. Port B (see *Figures 12, 13 and 14*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.

Ports A & B (Continued)

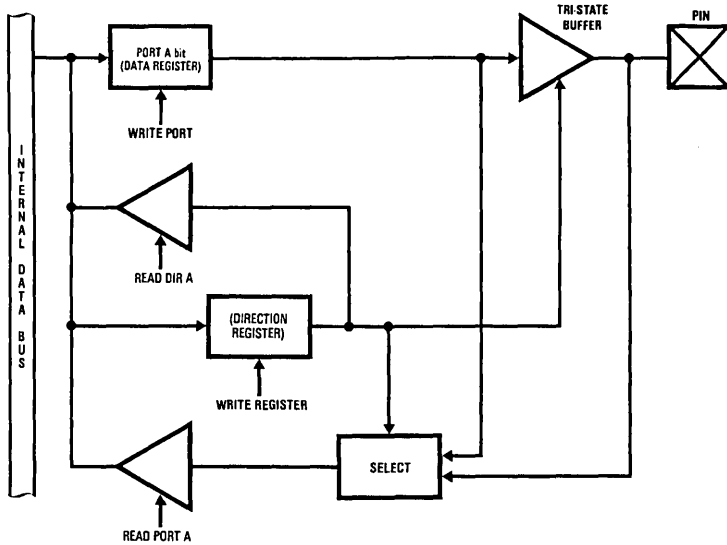


FIGURE 11. Port A: I/O Structure

TL/DD/11372-9

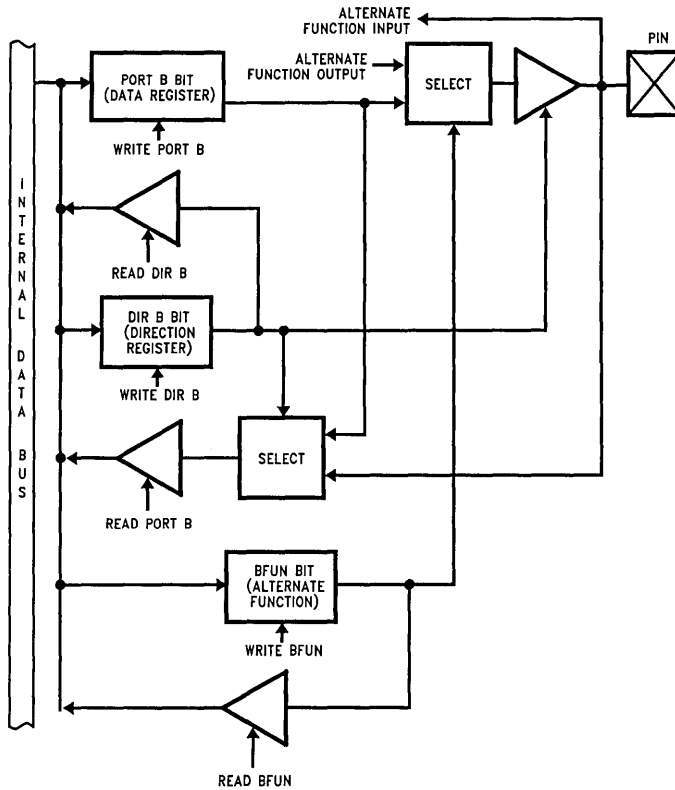
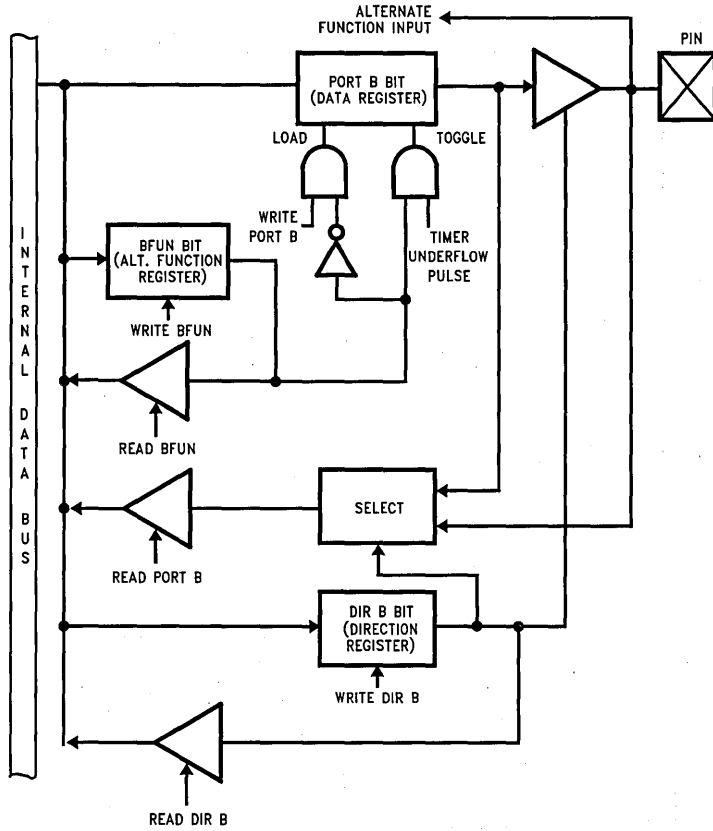


FIGURE 12. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

TL/DD/11372-10

Ports A & B (Continued)



TL/DD/11372-11

FIGURE 13. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)

Ports A & B (Continued)

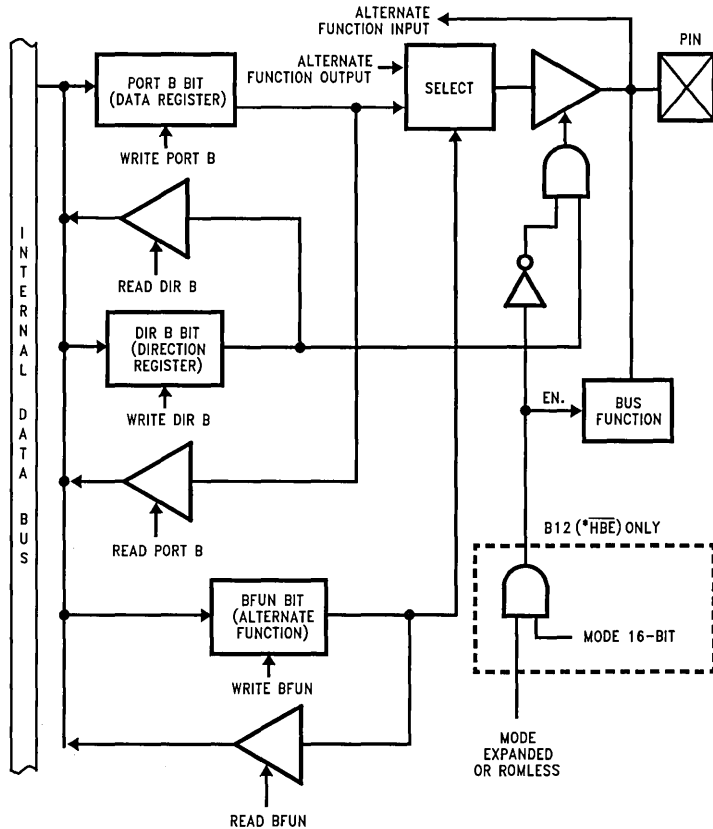


FIGURE 14. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

TL/DD/11372-12

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC46064 and HPC46004 have four operating modes. The ROMless HPC46004 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC46064 is C000 to FFFF (16k bytes). The HPC46004 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC46004 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range

and on-chip RAM and Register range, and the "illegal address detection" feature of the WATCHDOG logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the WATCHDOG logic is disabled. The EA bit should be set to "1" by software when using the HPC46004 to disable the "illegal address detection" feature of WATCHDOG.

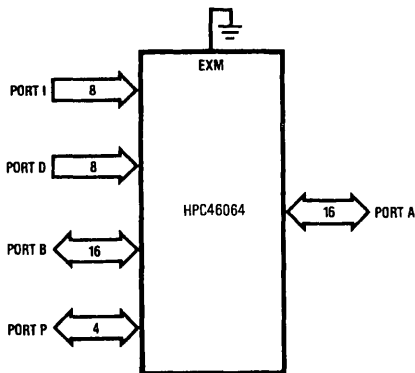
All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, RD, WR and HBE. The High Byte Enable pin (HBE) is used in 16-bit mode to select high order memory bytes. The RD and WR signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling HBE high at reset. If HBE is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC46064 and HPC46004.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC46064 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC46064 functions as a self-contained microcomputer (see Figure 15) with all memory (RAM and ROM) on-chip. It can address internal memory only, consisting of 16k bytes of ROM (C000 to FFFF) and 512 bytes of on-chip RAM and Registers (0000 to 02FF). The "illegal address detection" feature of the WATCHDOG is enabled in the Single-Chip Normal mode and a WATCHDOG Output (WO) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.



TL/DD/11372-13

FIGURE 15. Single-Chip Mode

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC46064 to address external memory in addition to the

on-chip ROM and RAM (see Table I). WATCHDOG illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC46064 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 16k of external memory may be used with the HPC46064 (see Table I). The WATCHDOG circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

TABLE I. HPC46064 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	C000:FFFF on-chip
Expanded Normal	0	1	C000:FFFF on-chip 0300:BFFF off-chip
Single-Chip ROMless	1	0	C000:FFFF off-chip
Expanded ROMless	1	1	0300:FFFF off-chip

Note: In all operating modes, the on-chip RAM and Registers (0000:02FF) may be accessed.

EXPANDED ROMLESS MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of WATCHDOG is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

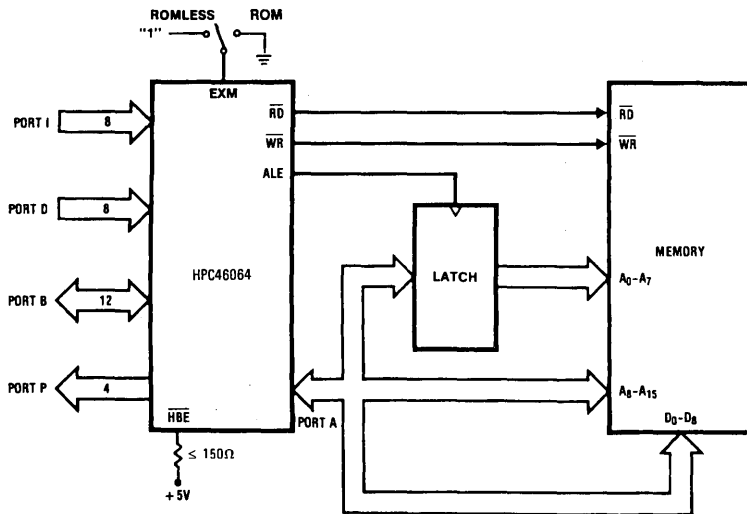


FIGURE 16. 8-Bit External Memory

TL/DD/11372-14

HPC46064 Operating Modes (Continued)

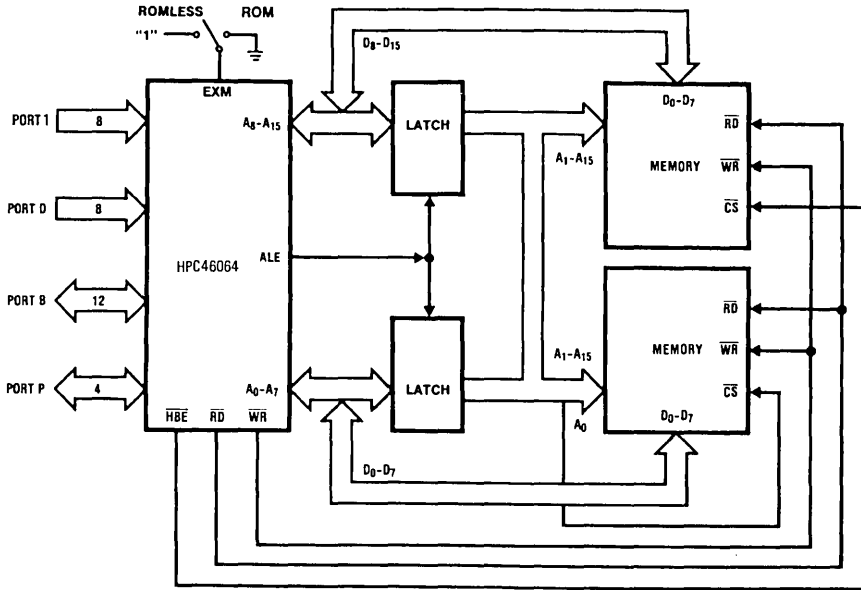


FIGURE 17. 16-Bit External Memory

TL/DD/11372-15

HPC46004 Operating Modes

EXPANDED ROMLESS MODE

Because the HPC46004 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC46004 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between C000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the WATCHDOG logic.

TABLE II. HPC46004 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0300:FFFF off-chip

Note: The on-chip RAM and Registers (0000:02FF) of the HPC46004 may be accessed at all times.

Wait States

The internal ROM can be accessed at the maximum operating frequency with one wait state. With 0 wait states, internal ROM accesses are limited to $\frac{1}{3} f_C$ max. The HPC46064 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC46064: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC46064 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC46064 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the \overline{RESET} or the NMI. The \overline{RESET} input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC46064 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the \overline{RESET} or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC46064 to resume normal operation.

HPC46064 Interrupts

Complex interrupt handling is easily accomplished by the HPC46064's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table III.

TABLE III. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
FFFF:FFFE	RESET	0
FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
FFFB:FFFA	External interrupt on I2 pin	2
FFF9:FFF8	External interrupt on I3 pin	3
FFF7:FFF6	External interrupt on I4 pin	4
FFF5:FFF4	Overflow on internal timers	5
FFF3:FFF2	Internal on the UART transmit/receive complete	6
FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC46064 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table III. The interrupt on Reset has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately. RESET and EXUI are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt (EXUI) is shared with the on-board UART. The EXUI interrupt is level-LOW-sensitive. To select this interrupt, disable the ERI and ETI UART interrupts by resetting these enable bits in the ENUI register. To select the on-board UART interrupt, leave this pin floating.

Interrupt Control Registers

The HPC46064 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled

or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC46064 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 18* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A and B in the TRI-STATE condition and Port P in the LOW state. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between C000 and FFFF when using the HPC46004.

Servicing the Interrupts (Continued)

TL/00/11372-16

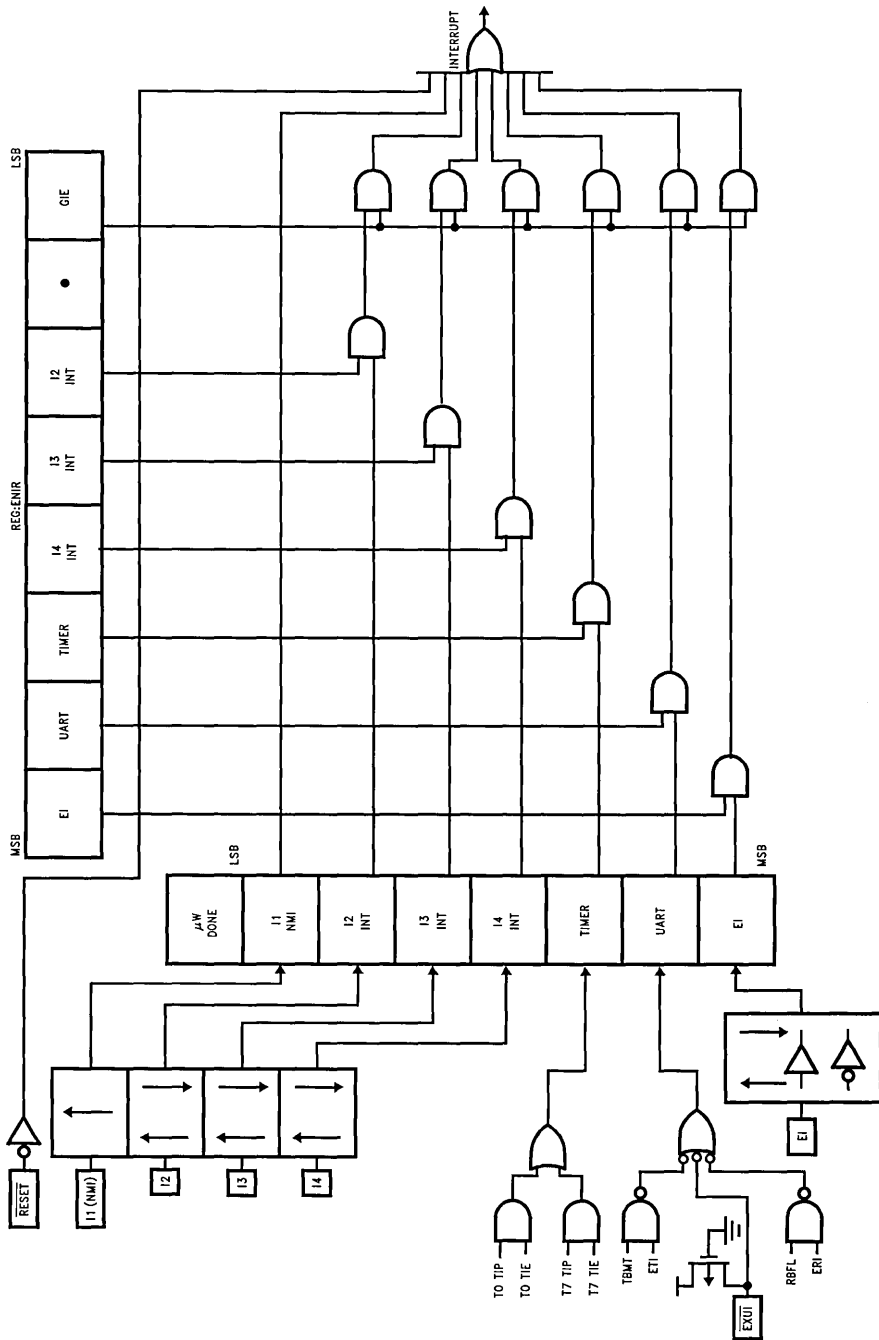


FIGURE 18. Block Diagram of Interrupt Logic

HPC16064/26064/36064/46064/16004/26004/36004/46004

Timer Overview

The HPC46064 contains a powerful set of flexible timers enabling the HPC46064 to perform extensive timer functions not usually associated with microcontrollers. The HPC46064 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for WATCHDOG logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see Figure 19).

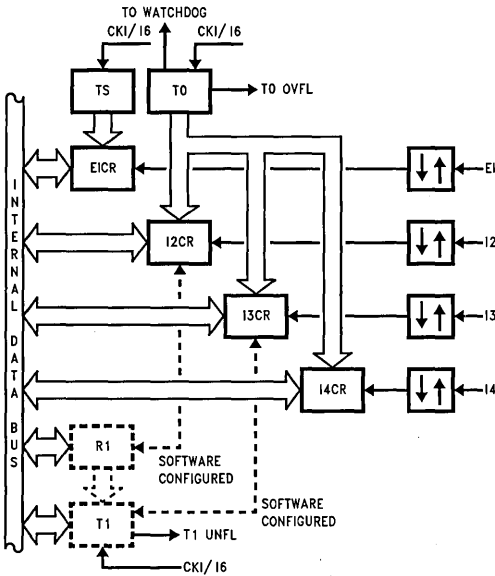


FIGURE 19. Timers T0, T1 and T8 with Four Input Capture Registers

The HPC46064 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records

the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see Figure 20).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC46064 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see Figure 20).

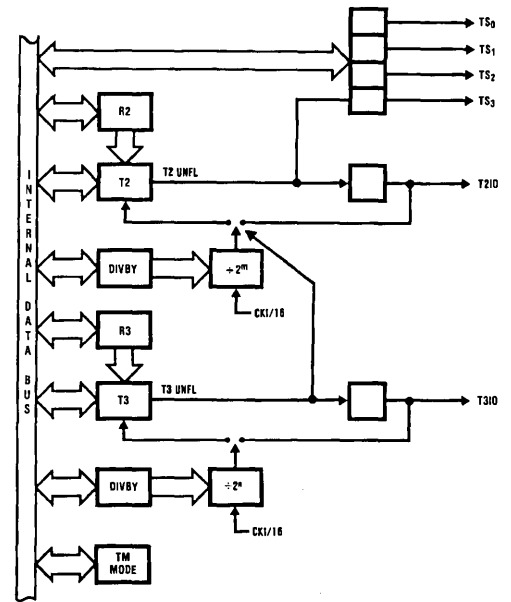


FIGURE 20. Timers T2-T3 Block

Timer Overview (Continued)

Timer/register pairs 4–7 form four identical units which can generate synchronous outputs on port P (see *Figure 21*). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to $\frac{1}{2}$ the frequency of the source used for clocking the timer.

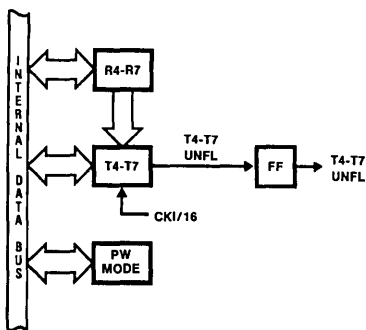


FIGURE 21. Timers T4–T7 Block

TL/DD/11372-19

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC46064.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

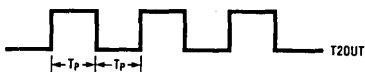
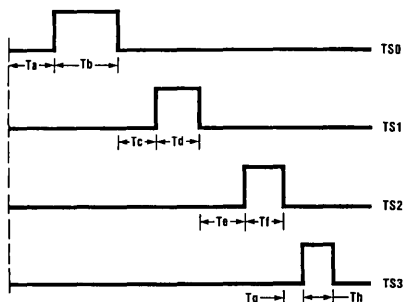


FIGURE 22. Square Wave Frequency Generation

TL/DD/11372-20

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 23* is an example of synchronous pulse train generation.



TL/DD/11372-21

FIGURE 23. Synchronous Pulse Generation

WATCHDOG Logic

The WATCHDOG Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the WATCHDOG logic are potentially infinite loops and illegal addresses. Should the WATCHDOG register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the WATCHDOG Output (\overline{WO}) pin low. The \overline{WO} pin is an open drain output and can be connected to the \overline{RESET} or NMI inputs or to the users external logic.

*Note: See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 24*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS (Continued)

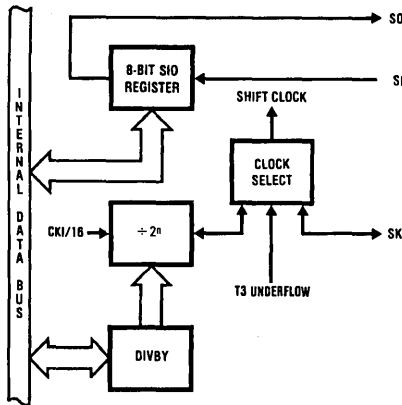


FIGURE 24. MICROWIRE/PLUS

MICROWIRE/PLUS Operation

The HPC46064 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC46064 is the master or slave. The shift clock is generated when the HPC46064 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC46064 is configured as a slave. When the HPC46064 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 14 se-

lectable binary steps or T3 underflow from 153 Hz to 1.25 MHz with CKI at 20.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 25 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC46064 microcontrollers interconnected to other MICROWIRE peripherals. HPC46064 #1 is set up as the master and initiates all data transfers. HPC46064 #2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of an LCD display controlled by the COP472 display driver. The data to be displayed is sent serially to the COP472 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC46064 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.

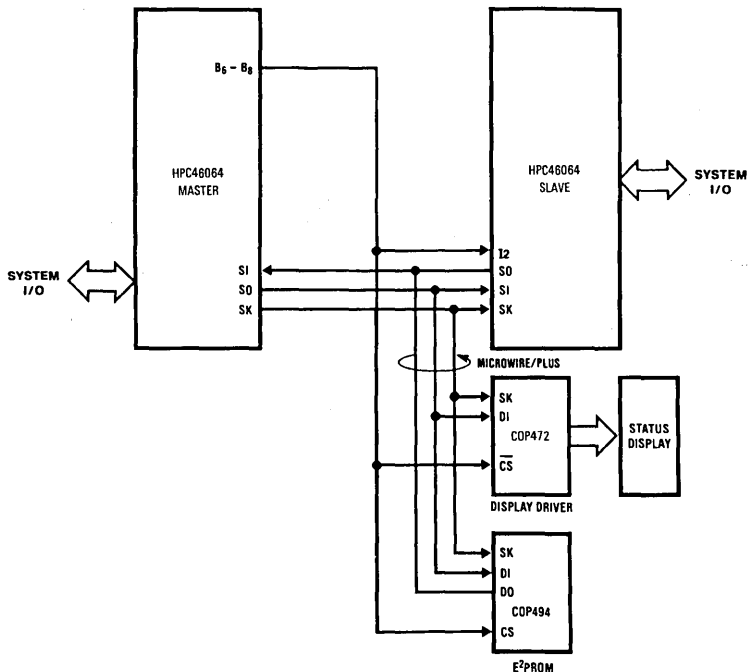


FIGURE 25. MICROWIRE/PLUS Application

TL/DD/11372-23

HPC46064 UART

The HPC46064 contains a software programmable UART. The UART (see Figure 26) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENUI register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC46064 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC46064 UART features a Wake-up Mode of operation. This mode of operation enables the HPC46064 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC46064 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

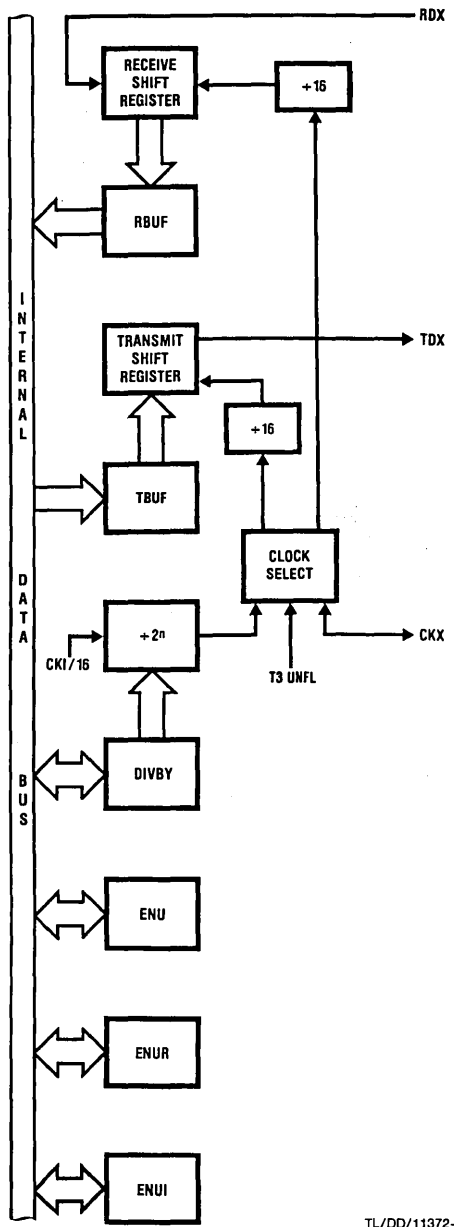


FIGURE 26. UART Block Diagram

TL/DD/11372-24

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC46064 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC46064's and set up systems with very high data exchange rates. Another area of application could be where an HPC46064 is programmed as an intelligent peripheral to a host system such as the Series 32000[®] microprocessor. *Figure 27* illustrates how an HPC46064 could be used as an intelligent peripheral for a Series 32000-based application. The interface consists of a Data Bus (port A), a Read Strobe (\overline{URD}), a Write Strobe (\overline{UWR}), a Read Ready Line (\overline{RDRDY}),

a Write Ready Line (\overline{WRRDY}) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The \overline{URD} and \overline{UWR} inputs may be used to interrupt the HPC46064. The \overline{RDRDY} and \overline{WRRDY} outputs may be used to interrupt the host processor.

The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC46064 is the data bus. UPI can only be used if the HPC46064 is in the Single-Chip mode.

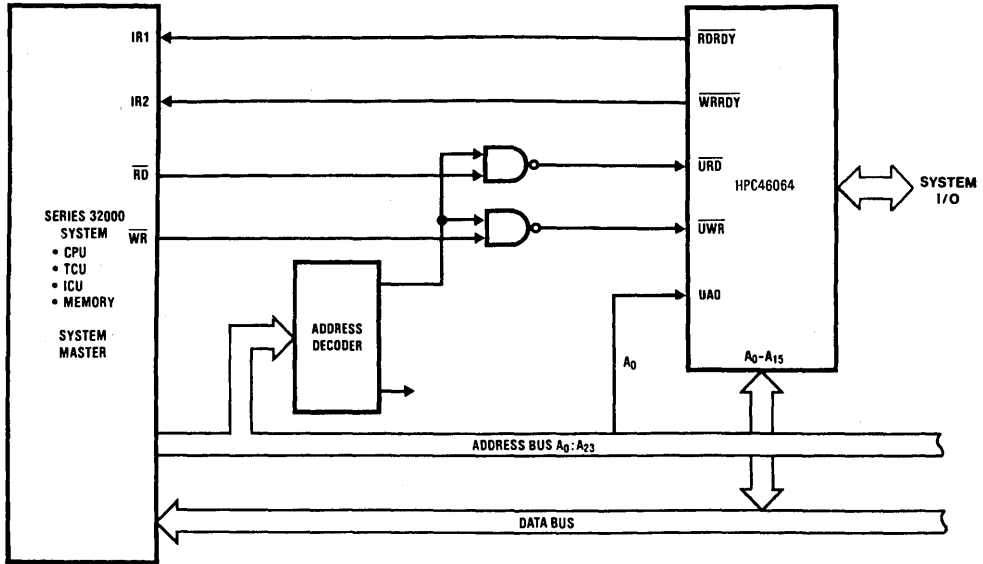


FIGURE 27. HPC46064 as a Peripheral: (UPI Interface to Series 32000 Application)

TL/DD/11372-25

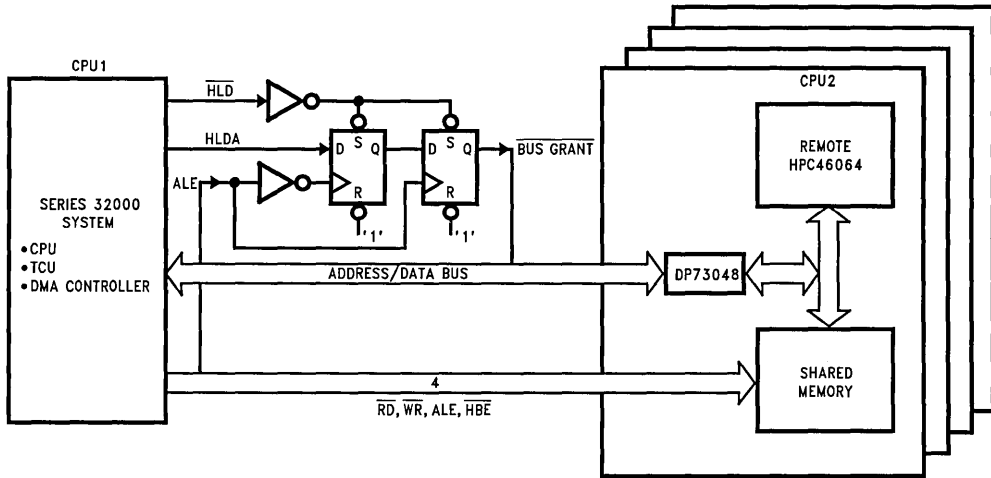
Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC46064 supports shared memory access with two pins. The pins are the RDY/HLD input pin and the HLD \bar{A} output pin. The user can software select either the Hold or Ready function by the state of a control bit. The HLD \bar{A} output is multiplexed onto port B.

The host uses DMA to interface with the HPC46064. The host initiates a data transfer by activating the HLD input of

the HPC46064. In response, the HPC46064 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal (HLD \bar{A}) from the HPC46064 indicating that the system bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC46064 resumes normal operations.

To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus. *Figure 28* illustrates an application of the shared memory interface between the HPC46064 and a Series 32000 system.



TL/DD/11372-26

FIGURE 28. Shared Memory Application: HPC46064 Interface to Series 32000 System

Memory

The HPC46064 has been designed to offer flexibility in memory usage. A total address space of 64 Kbytes can be addressed with 16 Kbytes of ROM and 512 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed

directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC46064 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC46064 memory address space extends to 64 Kbytes and registers and I/O are mapped as shown in Table IV.

TABLE IV. HPC46064 Memory Map

FFFF:FFF0 FFEF:FFD0 FFCF:FFCE : : E001:C000	Interrupt Vectors JSRP Vectors } On-Chip ROM*	USER MEMORY
BFFF:BFFE : : 0301:0300	} External Expansion Memory	
02FF:02FE : : 01C1:01C0	} On-Chip RAM	USER RAM
0195:0194	WATCHDOG Address	WATCHDOG Logic
0192 0191:0190 018F:018E 018D:018C 018B:018A 0189:0188 0187:0186 0185:0184 0183:0182 0181:0180	T0CON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/ R1 I3CR Register/ T1 I4CR Register	Timer Block T0:T3
015E:015F 015C 0153:0152 0151:0150 014F:014E 014D:014C 014B:014A 0149:0148 0147:0146 0145:0144 0143:0142 0141:0140	EICR EICON Port P Register PWMODE Register R7 Register T7 Timer R6 Register T6 Timer R5 Register T5 Timer R4 Register T4 Timer	Timer Block T4:T7

0128 0126 0124 0122 0120	ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART
0104	Port D Input Register	
00F5:00F4 00F3:00F2 00F1:00F0	BFUN Register DIR B Register DIR A Register / IBUF	PORTS A & B CONTROL
00E6	UPIC Register	UPI CONTROL
00E3:00E2 00E1:00E0	Port B Port A / OBUF	PORTS A & B
00DE 00DD:00DC 00D8 00D6 00D4 00D2 00D0	Reserved HALT Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00CF:00CE 00CD:00CC 00CB:00CA 00C9:00C8 00C7:00C6 00C5:00C4 00C3:00C2 00C0	X Register B Register K Register A Register PC Register SP Register Reserved PSW Register	HPC CORE REGISTERS
00BF:00BE : : 0001:0000	On-Chip RAM	USER RAM

*Note: The HPC46064 On-Chip ROM is on addresses C000:FFFF and the External Expansion Memory is 0300:BFFF. The HPC46004 have no On-Chip ROM, External Memory is 0300:FFFF.

Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused output should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least $1\ \mu\text{F}$ and bypass their outputs with a $10\ \mu\text{F}$ to $50\ \mu\text{F}$ tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a $10\ \mu\text{F}$ to $20\ \mu\text{F}$ tantalum electrolytic capacitor or a $50\ \mu\text{F}$ to $100\ \mu\text{F}$ aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately $10\ \text{nF}$ (spaced within $12\ \text{cm}$) per every two to five packages, and $100\ \text{nF}$ for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.

A recommended crystal oscillator circuit to be used with the HPC is shown in Figure 29. See Table V for recommended component values. The recommended values given in Table V have yielded consistent results and are made to match a crystal with a $20\ \text{pF}$ load capacitance, with some small allowance for layout capacitance.

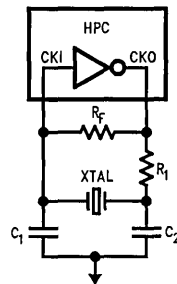
A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within "1" distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the

chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A $1.0\ \mu\text{F}$, a $0.1\ \mu\text{F}$, and a $0.001\ \mu\text{F}$ dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

TABLE V. HPC Oscillator Table

XTAL Freq (MHz)	R_1 (Ω)
≤ 2	1500
4	1200
6	910
8	750
10	600
12	470
14	390
16	300
18	220
20	180
22	150
24	120
26	100
28	75
30	62



$R_F = 3.3\ \text{M}\Omega$

$C_1 = 27\ \text{pF}$

$C_2 = 33\ \text{F}$

XTAL Specifications: The crystal used was an M-TRON Industries MP-1 Series XTAL. "AT" cut, parallel resonant

$C_L = 18\ \text{pF}$

Series Resistance is

$25\ \Omega @ 25\ \text{MHz}$

$40\ \Omega @ 10\ \text{MHz}$

$600\ \Omega @ 2\ \text{MHz}$

TL/DD/11372-31

FIGURE 29. Recommended Crystal Circuit

HPC46064 CPU

The HPC46064 CPU has a 16-bit ALU and six 16-bit registers:

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

HPC46064 CPU (Continued)

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC46064 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	$MA + Mem1 \rightarrow MA$ carry $\rightarrow C$
ADC	Add with carry	$MA + Mem1 + C \rightarrow MA$ carry $\rightarrow C$
ADDS	Add short imm8	$A + imm8 \rightarrow A$ carry $\rightarrow C$
DADC	Decimal add with carry	$MA + Mem1 + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
SUBC	Subtract with carry	$MA - Mem1 + C \rightarrow MA$ carry $\rightarrow C$
DSUBC	Decimal subtract w/carry	$MA - Mem1 + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
MULT	Multiply (unsigned)	$MA * Mem1 \rightarrow MA \& X, 0 \rightarrow K, 0 \rightarrow C$
DIV	Divide (unsigned)	$MA / Mem1 \rightarrow MA, rem. \rightarrow X, 0 \rightarrow K, 0 \rightarrow C$
DIVD	Divide Double Word (unsigned)	$X \& MA / Mem1 \rightarrow MA, rem \rightarrow X, 0 \rightarrow K, Carry \rightarrow C$
IFEQ	If equal	Compare MA & Mem1, Do next if equal
IFGT	If greater than	Compare MA & Mem1, Do next if $MA > Mem1$
AND	Logical and	$MA \text{ and } Mem1 \rightarrow MA$
OR	Logical or	$MA \text{ or } Mem1 \rightarrow MA$
XOR	Logical exclusive-or	$MA \text{ xor } Mem1 \rightarrow MA$
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	$Mem + 1 \rightarrow Mem$
DECSZ	Decrement, skip if 0	$Mem - 1 \rightarrow Mem$, Skip next if $Mem = 0$

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Mem1 → MA
ST	Load, incr/decr X	Mem(X) → A, X ± 1 (or 2) → X
X	Store to Memory	A → Mem
	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ± 1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP
POP	Pop Stack to Memory	SP - 2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ± 1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A, B ± 1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B, imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A - 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP), SP + 2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP), SP + 2 → SP, PC + # → PC (# is +1025 to -1023)
JSRL	Jump subroutine long	PC → W(SP), SP + 2 → SP, PC + # → PC
JP	Jump relative short	PC + # → PC (# is +32 to -31)
JMP	Jump relative	PC + # → PC (# is +257 to -255)
JMPL	Jump relative long	PC + # → PC
JID	Jump indirect at PC + A	PC + A + 1 → PC
JIDW		then Mem(PC) + PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP - 2 → SP, W(SP) → PC
RETSK	Return then skip next	SP - 2 → SP, W(SP) → PC, & skip
RETI	Return from interrupt	SP - 2 → SP, W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory
 MA is Accumulator A or direct memory (8- or 16-bit)
 Mem is 8-bit byte or 16-bit word of memory
 Mem1 is 8- or 16-bit memory or 8- or 16-bit immediate data
 imm is 8-bit or 16-bit immediate data
 imm8 is 8-bit immediate data only

Memory Usage

Number of Bytes for Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir.		Direct	Indir	Index	Immed.	Direct		Immed.	
	(B)	(X)					*	**	*	**
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
 **16-bit direct address

Instructions that Modify Memory Directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B-)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X-)
LDA,*	1	1
XA,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC46064 has been designed to be extremely code-efficient. The HPC46064 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC46064, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC46064 are single-byte. There are two especially code-saving instructions: JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of the 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into this table; the assembler can give this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC46064 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC46064 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC46064 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

HPC Microcontroller Development System

The HPC microcontroller development system is an in-system emulator (ISE) designed to support the entire family of HPC Microcontrollers. The complete package of hardware and software tools combined with a host system provides a powerful system for design, development and debug of HPC based designs. Software tools are available for IBM® PC-AT® (MS-DOS, PC-DOS) and for Unix based multi-user Sun® Sparcstation (SunOST™).

The stand alone units comes complete with a power supply and external emulation POD. This unit can be connected to various host systems through an RS-232 link. The software package includes an ANSI compatible C-Compiler, Linker, Assembler and librarian package. Source symbolic debug capability is provided through a user friendly MS-windows 3.0 interface for IBM PC-AT environment and through a line debugger under Sunview for Sun Sparcstations.

The ISE provides fully transparent in-system emulation at speeds up to 20 MHz 1 waitstate. A 2k word (48-bit wide) trace buffer gives trace trigger and non-intrusive monitoring of the system. External triggering is also available through an external logic interface socket on the POD. Direct EPROM programming can be done through the use of externally mounted EPROM socket. Form-Fit-Function emulator programming is supported by a programming board included with the system. Comprehensive on-line help and diagnostics features reduce user's design and debug time. 8 hardware breakpoints (Address/range), 64 Kbytes of user memory, and break on external events are some of the other features offered.

Hewlett Packard model HP64775 Emulator/Analyzer providing in-system emulation for up to 30 MHz 1 waitstate is also available. Contact your local sales office for technical details and support.

Development Support (Continued)**Development Tools Selection Table**

Product	Order Number	Description	Included	Manual Number
HPC16004/ 16064	HPC-DEV-ISE4 HPC-DEV-ISE4-E	HPC In-System Emulator HPC in-System Emulator for Europe and South East Asia	HPC MDS User's Manual MDS Comm User's Manual HPC Emulator Programmer User's Manual HPC16004/16064 Manual	420420184-001 424420188-001 420421313-001
	NPC-DEV-IBMA	Assembler/Linker/ Library Package for IBM PC-AT	HPC Assembler/Linker Librarian User's Manual	424410836-001
	HPC-DEV-IBMC	C Compiler/Assembler/ Linker/Library Package for IBM PC-AT	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-001 424410836-001
	HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC-AT C Compiler/Assembler/ Linker Library Package for IBM PC-AT	Source/Symbolic Debugger User's Manual	424420189-001
			HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-001 424410836-001
	HPC-DEV-SUNC	C-Compiler/Assembler/ Linker Library Package for Sun Sparcstation	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	
HPC-DEV-SUNDB	Source/Symbolic Debugger for Sun Sparcstation C Compiler/Assembler/Linker Library Package	Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual		
Complete System: HPC16004/ 16064	HPC-DEV-SYS4	HPC In-System Emulator with C Compiler/Assembler/ Linker/Library and Source Symbolic Debugger		
	HPC-DEV-SYS4-E	Same for Europe and South East Asia		

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION

(electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MDS-DIAL-A-HLP

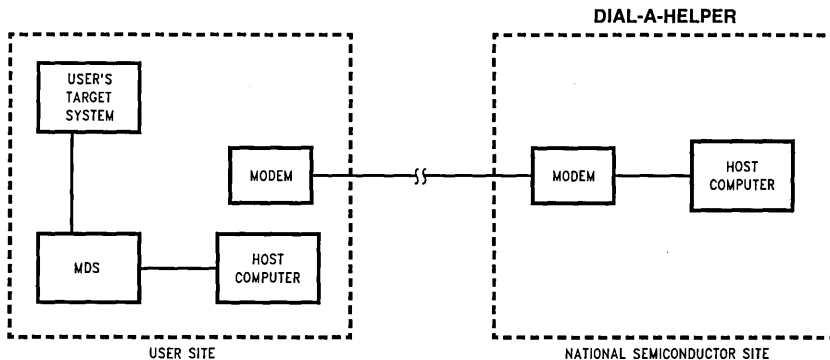
Information System Package Contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

Development Support (Continued)

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a development system, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582
 Modem: (408) 739-1162
 Baud: 300 or 1200 baud
 Set-Up: Length: 8-Bit
 Parity: None
 Stop Bit: 1
 Operation: 24 Hrs. 7 Days

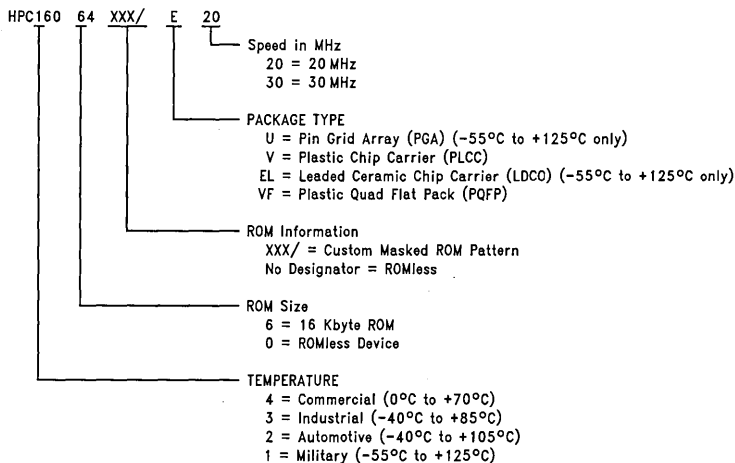


TL/DD/11372-35

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC46064 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/11372-36

FIGURE 8. HPC Family Part Numbering Scheme

Examples

- HPC46004V20 — ROMless, Commercial temperature (0°C to 70°C), PLCC
- HPC16064XXX/U20— 16k masked ROM, Military temperature (-55°C to 125°C), PGA
- HPC26004XXX/V20— ROMless, Automotive temperature (-40°C to +105°C), PLCC



HPC36400E/HPC46400E

High-Performance Communications microController

General Description

The HPC46400E is an upgraded HPC16400. Features have been added to support V.120, the 8-bit mode has been enhanced to support all instructions, and the UART has been changed to provide more flexibility and power. The HPC46400E is fully upward compatible with the HPC16400.

The HPC46400E has 4 functional blocks to support a wide range of communication application—2 HDLC channels, 4 channel DMA controller to facilitate data flow for the HDLC channels, programmable serial interface and UART.

The serial interface decoder allows the 2 HDLC channels to be used with devices using interchip serial link for point-to-point and multipoint data exchanges. The decoder generates enable signals for the HDLC channels allowing multiplexed D and B channel data to be accessed.

The HDLC channels manage the link by providing sequencing using the HDLC framing along with error control based upon a cyclic redundancy check (CRC). Multiple address recognition modes, and both bit and byte modes of operation are supported.

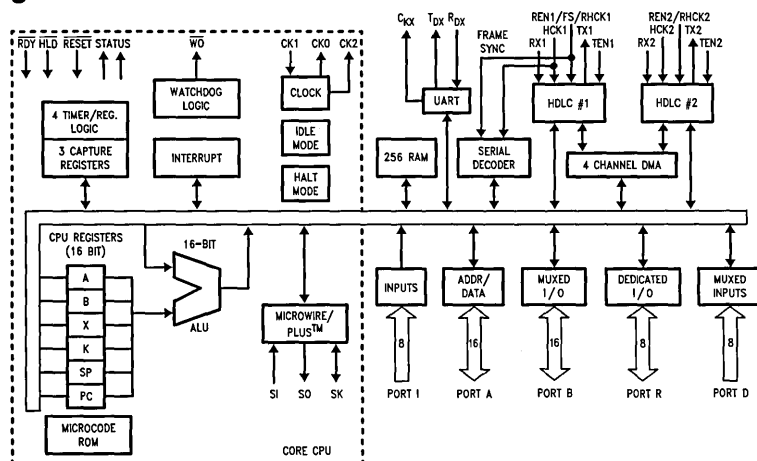
The HPC36400E and HPC46400E are available in 68-pin PLCC and LDCC packages.

Features

- HPCTM family—core features:
 - 16-bit data bus, ALU, and registers
 - 64 kbytes of external memory addressing
 - FAST!—20.0 MHz system clock
 - Four 16-bit timer/counters with WATCHDOG™ logic
 - MICROWIRE/PLUS™ serial I/O interface
 - CMOS—low power with two power save modes

- Two full duplex HDLC channels
 - Optimized for ISDN, X.25, V.120, and LAPD applications
 - Programmable frame address recognition
 - Up to 4.65 Mbps serial data rate
 - Built in diagnostics
 - Synchronous bypass mode
 - Optional CRC generation
 - Received CRC bytes can be read by the CPU
- Four channel DMA controller
- 8- or 16-bit external data bus
- UART
 - Full duplex
 - 7, 8, or 9 data bits
 - Even, odd, mark, space or no parity
 - 7/8, 1 or 2 stop bit generation
 - Accurate internal baud rate generation up to 625k baud without penalty of using expensive crystal
 - Synchronous and asynchronous modes of operation
- Serial Decoder
 - Supports 6 popular time division multiplexing protocols for inter-chip communications
 - Optional rate adaptation of 64 kbit/s data rate to 56 kbit/s
- Over 1/2 Mbyte of extended addressing
- Easy interface to National's DASL, 'U' and 'S' transceivers—TP3400, TP3410 and TP3420
- Commercial (0°C to +70°C) and industrial (-40°C to +85°C)

Block Diagram



TL/DD/10422-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

V_{CC} with Respect to GND -0.5V to 7.0V
 All Other Pins ($V_{CC} + 0.5$)V to (GND - 0.5)V
 Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC46400E, $-40^\circ C$ to $+85^\circ C$ for HPC36400E

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC1}	Supply Current	$V_{CC} = 5.5V, f_{in} = 20.0$ MHz (Note 1)		70	mA
		$V_{CC} = 5.5V, f_{in} = 2.0$ MHz (Note 1)		10	mA
I_{CC2}	IDLE Mode Current	$V_{CC} = 5.5V, f_{in} = 20.0$ MHz (Note 1)		10	mA
		$V_{CC} = 5.5V, f_{in} = 2.0$ MHz (Note 1)		2	mA
I_{CC3}	HALT Mode Current	$V_{CC} = 5.5V, f_{in} = 0$ kHz (Note 1)		500	μA
		$V_{CC} = 2.5V, f_{in} = 0$ kHz (Note 1)		150	μA
INPUT VOLTAGE LEVELS—SCHMITT TRIGGERED: RESET, CKI, WO, D0, I1, I2, I3					
V_{IH1}	Logic High		$0.9 V_{CC}$		V
V_{IL1}	Logic Low			$0.1 V_{CC}$	V
INPUT VOLTAGE LEVELS—PORT A					
V_{IH2}	Logic High		2.0		V
V_{IL2}	Logic Low			0.8	V
INPUT VOLTAGE LEVELS—ALL OTHERS					
V_{IH3}	Logic High		$0.7 V_{CC}$		V
V_{IL3}	Logic Low			$0.2 V_{CC}$	V
I_{LI}	Input Leakage Current	(Note 2)		± 1	μA
C_I	Input Capacitance	(Note 3)		10	pF
C_{IO}	I/O Capacitance	(Note 3)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH1}	Logic High (CMOS)	$I_{OH} = -10 \mu A$ (Note 3)	$V_{CC} - 0.1$		V
V_{OL1}	Logic Low (CMOS)	$I_{OH} = 10 \mu A$ (Note 3)		0.1	V
V_{OH2}	Port A/B Drive, CK2 (A ₀ -A ₁₅ , B ₁₀ , B ₁₁ , B ₁₂ , B ₁₅)	$I_{OH} = -1$ mA	2.4		V
		$I_{OL} = 3$ mA		0.4	V
V_{OH3}	Other Port Pin Drive, WO (open drain) (B ₀ -B ₉ , B ₁₃ , B ₁₄ , R ₀ -R ₇ , D ₅ , D ₇)	$I_{OH} = -1.6$ mA (except WO)	2.4		V
		$I_{OL} = 0.5$ mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	$I_{OH} = -6$ mA	2.4		V
		$I_{OL} = 1.6$ mA (Note 4)		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 5)	2.5		V
I_{OZ}	TRI-STATE Leakage Current			± 5	μA

Note 1: I_{CC1} , I_{CC2} , I_{CC3} measured with no external drive (I_{OH} and $I_{OL} = 0$, I_{IH} and $I_{IL} = 0$). I_{CC1} is measured with RESET = V_{SS} . I_{CC3} is measured with NMI = V_{CC} . CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: RDY/HLD and RDY/I4 pins have internal pullups and meet this spec only at $V_{IN} = V_{CC}$.

Note 3: These parameters are guaranteed by design and are not tested.

Note 4: ST2 drive will not meet this spec under condition of RESET pin = low.

Note 5: Test duration is 100 ms.

AC Electrical Characteristics

(see Notes 1 and 4 and Figures 1 thru 5), $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46400E, -40°C to $+85^\circ\text{C}$ for HPC36400E

	Symbol and Formula	Parameter and Notes	Min	Max	Units	Note
Clocks	f_C	CKI Operating Frequency	2	20	MHz	
	$t_{C1} = 1/f_C$	CKI Period	50	500	ns	
	t_{C1R}	CKI Rise Time		7	ns	(Note 1)
	t_{C1F}	CKI Fall Time		7	ns	(Note 1)
	$100 t_{C1H}/t_{C1}$	CKI Duty Cycle	45	55	%	(Note 1)
	$t_C = 2/t_C$	CPU or DMA Timing Cycle	100		ns	
	$t_{WAIT} = t_C$	CPU or DMA Wait State Period	100		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CK1 Falling Edge	0	55	ns	(Note 2)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CK1 Falling Edge	0	55	ns	(Note 2)
Timers	$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz	
	$f_{MW} = f_C/19$	External MICROWIRE/PLUS Clock Input Frequency		1.052	MHz	
	$t_{HCK} = 4t_{C1} + 14$	HDL Clock Input Period	214		ns	
MICROWIRE/ PLUS	$f_{XIN} = f_C/19$	External Timer Input Frequency		1052	kHz	
	$t_{XIN} = t_C$	Pulse Width for Timer Inputs	100		ns	
	$f_{XOUT} = f_C/16$	Timer Output Frequency		1.25	MHz	
External Hold	t_{UWS}	MICROWIRE Setup Time — Master — Slave	100 20		ns ns	
	t_{UWH}	MICROWIRE Hold Time — Master — Slave	20 50		ns ns	
	t_{UWV}	MICROWIRE Output Valid Time — Master — Slave		50 150	ns ns	
External Hold	$t_{SALE} = 3/4 t_C + 40$	\overline{HLD} Falling Edge before \overline{ALE} Rising Edge	115		ns	(Note 3)
	$t_{HWP} = 3/4 t_C + 35$	\overline{HLD} Pulse Width	110		ns	
	$t_{HAE} = 3/4 t_C + 100$	\overline{HLDA} Falling Edge after \overline{HLD} Falling Edge		175	ns	
	$t_{HAD} = 5/4 t_C + 85$	\overline{HLDA} Rising Edge after \overline{HLD} Rising Edge		210	ns	
	t_{BF}	Bus Float after \overline{HLDA} Falling Edge		66	ns	
	$t_{BE} = t_C - 66$	Bus Enable after \overline{HLDA} Rising Edge	34		ns	

Note 1: These AC characteristics are guaranteed with external clock drive on CKI having 50% duty cycle and with less than 15 pF load on CKO. Spec'd t_{C1R} , t_{C1F} , and CKI duty cycle limits are not tested but are guaranteed functional by design.

Note 2: Do not design with this parameter unless CKI is driven with an active signal meeting T_{C1R} and T_{C1F} specs. When using a passive crystal circuit, its stability is not guaranteed if either CKI or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{HLD} falling edge occurring at the latest time it can be accepted during the present CPU or DMA cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} as long as $(3 t_C + 4 WS + 72 t_C + 100)$ may occur depending on the following CPU instruction or DMA cycle, its wait states and ready input.

Note 4: $WS (t_{WAIT}) \times$ (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $f_C = 20$ MHz, with one wait state preprogrammed. These values are guaranteed with AC loading of 100 pF on Port A, 50 pF on CK2, 80 pF on other outputs, and DC loading of the pin's DC spec non CMOS I_{OL} or I_{OH} .

AC Electrical Characteristics (Continued)

CPU and DMA Timing (see Notes 1 and 4 and Figures 2, 4, 6, 7, 8, and 9), $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC46400E, -40°C to $+85^\circ\text{C}$ for HPC36400E

	Symbol	Formula	Cycle	Parameter	Min	Max	Units	Note
Address Cycles	t_{1ALR}		CPU DMA	Delay of ALE Rising Edge after CKI Rising Edge Delay of ALE Rising Edge after CKI Falling Edge	0 0	35 35	ns ns	(Note 2) (Note 2)
	t_{1ALF}		CPU DMA	Delay of ALE Falling Edge after CKI Rising Edge Delay of ALE Falling Edge after CKI Falling Edge	0 0	35 35	ns ns	(Note 2) (Note 2)
	t_{2ALR}	$\frac{1}{4} t_C + 20$	CPU	ALE Rising Edge after CK2 Rising Edge		45	ns	
	t_{2ALF}	$\frac{1}{4} t_C + 20$	CPU	ALE Falling Edge after CK2 Falling Edge		45	ns	
	t_{LL}	$\frac{1}{2} t_C - 9$		ALE Pulse Width	41		ns	
	t_{ST}	$\frac{1}{4} t_C - 20$		Setup of Address Valid before ALE Falling Edge	5		ns	(Note 3)
	t_{VP}	$\frac{1}{4} t_C - 10$ $\frac{1}{2} t_C - 10$	CPU DMA	Hold of Address Valid after ALE Falling Edge	15 40		ns ns	
Read Cycles	t_{ARR}	$\frac{1}{2} t_C - 20$		ALE Falling Edge to \overline{RD} Falling Edge	30		ns	
	t_{ACC}	$t_C + WS - 55$ $\frac{5}{4} t_C + WS - 75$	CPU DMA	Data Input Valid after Address Output Valid		145 150	ns ns	
	t_{RD}	$\frac{1}{4} t_C + WS - 35$ $\frac{1}{2} t_C + WS$	CPU DMA	Data Input Valid after \overline{RD} Falling Edge		90 115	ns ns	
	t_{RW}	$\frac{1}{4} t_C + WS - 15$ $\frac{1}{2} t_C + WS - 15$	CPU DMA	\overline{RD} Pulse Width	110 135		ns ns	
	t_{DR}	$t_C - 15$ $\frac{3}{4} t_C - 15$	CPU DMA	Hold of Data Input Valid after \overline{RD} Rising Edge	0 0	85 60	ns ns	(Note 5) (Note 5)
	t_{RDA}	$t_C - 5$ $\frac{3}{4} t_C - 10$	CPU DMA	Bus Enable after \overline{RD} Rising Edge	95 65		ns ns	(Note 5) (Note 5)
Write Cycles	t_{ARW}	$\frac{1}{2} t_C - 20$		ALE Falling Edge to \overline{WR} Falling Edge	30		ns	
	t_{WW}	$\frac{3}{4} t_C + WS - 15$ $\frac{1}{2} t_C + WS - 15$	CPU DMA	\overline{WR} Pulse Width	160 135		ns ns	
	t_V	$\frac{1}{2} t_C + WS - 40$ $\frac{1}{2} t_C + WS - 50$	CPU DMA	Data Output Valid before \overline{WR} Rising Edge	110 100		ns ns	
	t_{HW}	$\frac{1}{4} t_C - 10$		Hold of Data Output Valid after \overline{WR} Rising Edge	15		ns	
Ready Input	t_{RDYS}			\overline{RDY} Falling Edge before CK2 Rising Edge	45		ns	
	t_{RDYH}			\overline{RDY} Rising Edge after CK2 Rising Edge	0		ns	
	t_{RDYV}	$WS - \frac{1}{4} t_C - 47$ $t_C - 47$	CPU DMA	\overline{RDY} Falling Edge after \overline{RD} or \overline{WR} Falling Edge		28 53	ns ns	(Note 6)

Note 1: These AC characteristics are guaranteed with external clock drive on CKI having 50% duty cycle and with less than 15 pF load on CKO. Spec'd t_{C1R} , t_{C1F} , and CKI duty cycle limits are not tested but are guaranteed functional by design.

Note 2: Do not design with this parameter unless CKI is driven with an active signal meeting T_{C1R} and T_{C1F} specs. When using a passive crystal circuit, its stability is not guaranteed if either CKI or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: Setup of HBE valid before ALE falling edge is 0 ns minimum.

Note 4: WS (t_{WAIT}) \times (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $f_C = 20$ MHz, with one wait state preprogrammed. These values are guaranteed with AC loading of 100 pF on Port A, 50 pF on CK2, 80 pF on other outputs, and DC loading of the pin's DC spec non CMOS I_{OL} or I_{OH} .

Note 5: Formula has $\frac{1}{4} t_C$ for CPU read followed by DMA $\frac{1}{4} t_C$ for DMA read followed by CPU.

Note 6: In HPC in-circuit emulators the t_{RDYV} formulas are $WS - \frac{1}{4} t_C - 57$ and $t_C - 57$ yielding minimums of 18 ns and 43 ns for CPU and DMA cycles, respectively.

Timing Waveforms

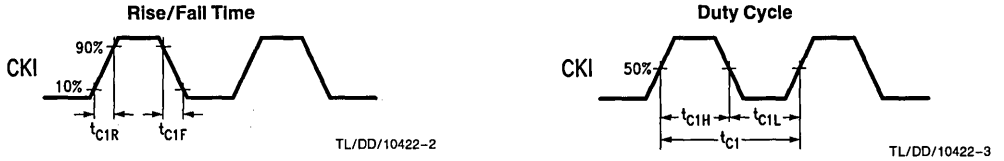


FIGURE 1. CKI Input Signal



Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at 2.0V for a logic "1" hold or rising edge and at 0.8V for a logic "0" hold or falling edge.

FIGURE 2. Input and Output for AC Tests

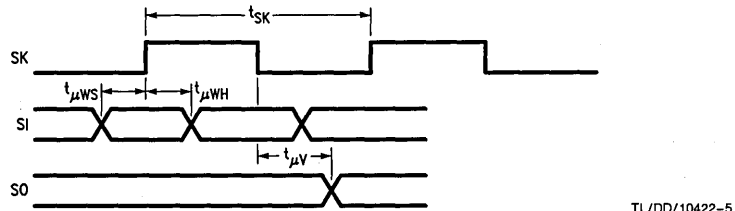


FIGURE 3. MICROWIRE Setup/Hold Timing

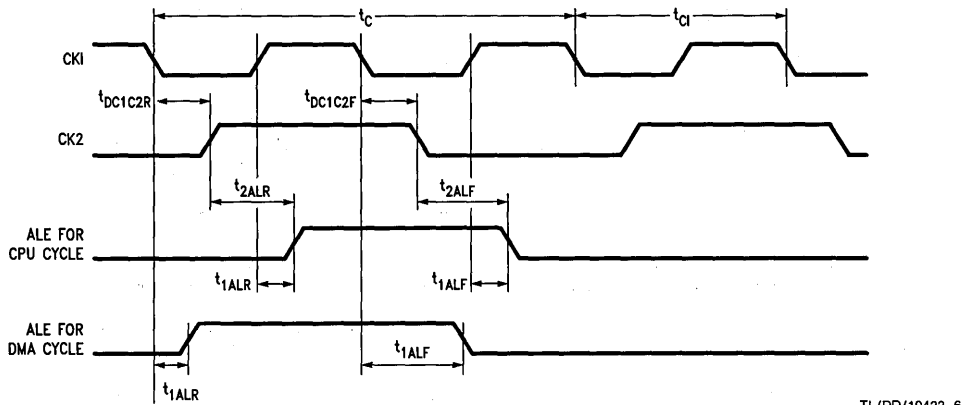


FIGURE 4. CK1, CK2 ALE Timing Diagram

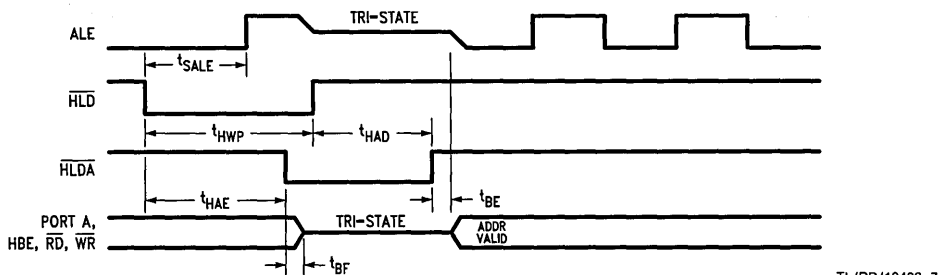


FIGURE 5. External Hold Timing

Timing Waveforms (Continued)

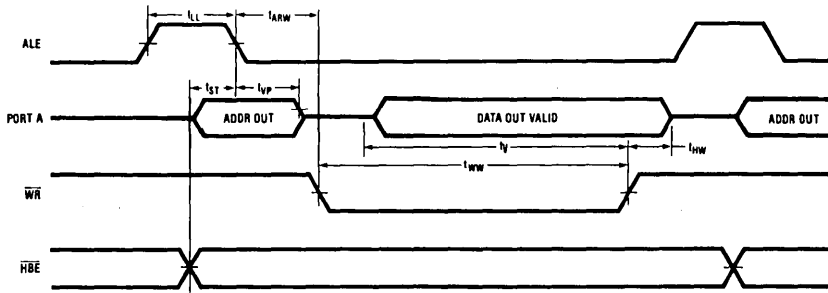


FIGURE 6. CPU and DMA Write Cycles

TL/DD/10422-8

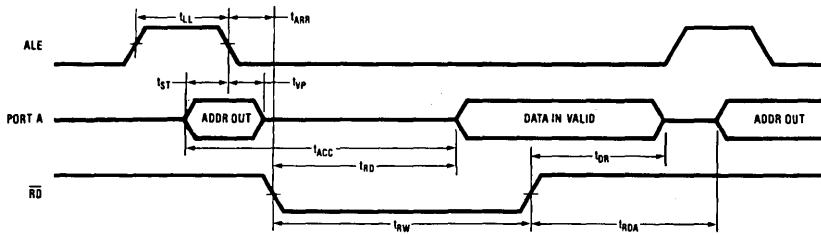


FIGURE 7. CPU and DMA Read Cycles

TL/DD/10422-9

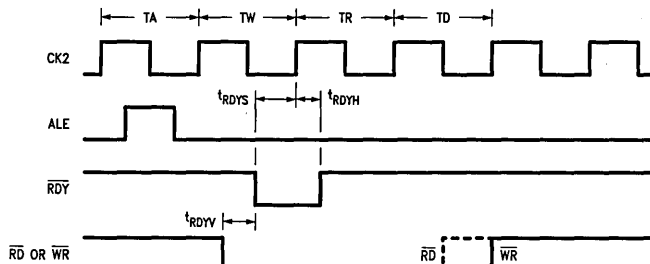


FIGURE 8. CPU Ready Mode with 1 Wait State and Ready Wait Extension

TL/DD/10422-10

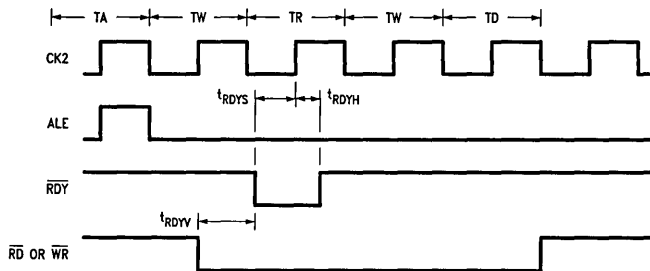
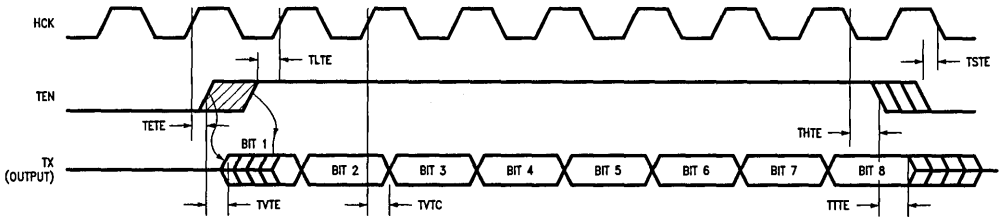


FIGURE 9. DMA Ready Mode with 2 Wait States and Ready Wait Extension

TL/DD/10422-11

Timing Waveforms (Continued)

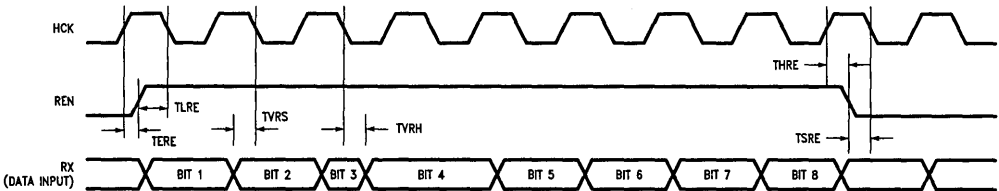
Timing Diagrams for TX Using External Enable



TL/DD/10422-12

Symbol	Parameter	Min	Max	Units
TETE	Hold of TEN Low after HCK Rising Edge	5		ns
TLTE	Setup of TEN Rising Edge before HCK Rising Edge	85		ns
TVTE	Delay of TX Output Valid after TEN Rising Edge		40	ns
TVTC	Delay of TX Output Valid after HCK Rising Edge		65	ns
THTE	Hold of TEN High after HCK Falling Edge	60		ns
TSTE	Setup of TEN Falling Edge before HCK Falling Edge	20		ns
TTTE	Delay of TX Output TRI-STATE® after TEN Falling Edge		40	ns
TVTR	TVTC in Rate Adaptation Mode		75	ns

Timing Diagrams for RX Using External Enable

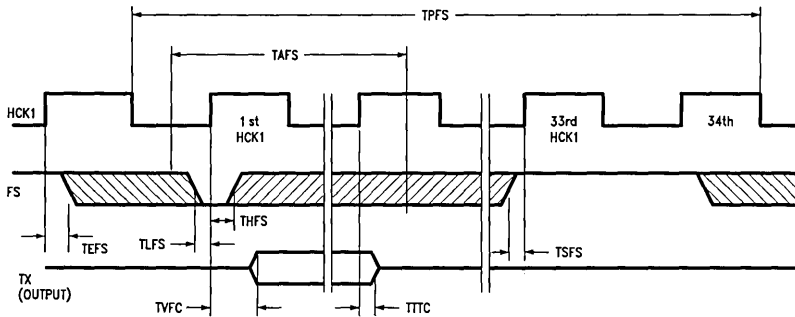


TL/DD/10422-13

Symbol	Parameter	Min	Max	Units
TERE	Hold of REN Low after HCK Rising Edge	5		ns
TLRE	Setup of REN Rising Edge before HCK Falling Edge	30		ns
TVRS	Setup of RX Data Input Valid before HCK Falling Edge	20		ns
TVRH	Hold of RX Data Input Valid after HCK Falling Edge	20		ns
THRE	Hold of REN High after HCK Rising Edge	5		ns
TSRE	Setup of REN Falling Edge before HCK Falling Edge	30		ns

Timing Waveforms (Continued)

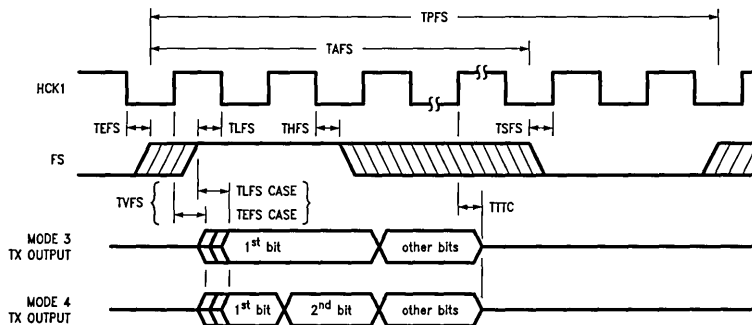
Serial Decoder Timing Diagram (Mode 2)



TL/DD/10422-14

Symbol	Parameter	Min	Max	Comments	Units
TPFS	Number of HCK1 Periods between FS Falling Edges	34			
TAFS	Number of HCK1 Rising Edges during FS Low	1	32		
TEFS	Hold of FS High after HCK1 Rising Edge	10		Early FS	ns
TLFS	Setup of FS Falling Edge before HCK1 Rising Edge	20		Late FS, (Note 8)	ns
TVFC	Delay of TX Output Valid after HCK1 Rising Edge		60	(Note 7)	ns
THFS	Hold of FS Low after HCK1 Rising Edge	20			ns
TSFS	Setup of FS Rising Edge before HCK1 Rising Edge	20			ns
TTTC	Delay of TX output TRI-STATE after HCK1 Rising Edge		40		ns
TVFR	TVFC in Rate Adaptation Mode		75		ns

Serial Decoder Timing Diagram (Modes 3, 4)



TL/DD/10422-15

Symbol	Parameter	Min	Max	Comments	Units
TPFS	Number of HCK1 Periods between FS Rising Edges	64		SD Mode 3	
TPFS	Number of HCK1 Periods between FS Rising Edges	32		SD Mode 4	
TAFS	Number of HCK1 Falling Edges during FS High	2	62	SD Mode 3	
TAFS	Number of HCK1 Falling Edges during FS High	2	30	SD Mode 4	
TEFS	Hold of FS Low after HCK1 Falling Edge	10		Early FS	ns
TLFS	Setup of FS Rising Edge before HCK1 Falling Edge	45		Late FS, (Note 8)	ns
TVFS	Delay of TX Output Valid after HCK1 and FS Rising Edges		70	(Note 9)	ns
THFS	Hold of FS High after HCK1 Falling Edge	20			ns
TSFS	Setup of FS Falling Edge before HCK1 Rising Edge	20			ns
TTTC	Delay of TX output TRI-STATE after HCK1 Rising Edge		40		ns

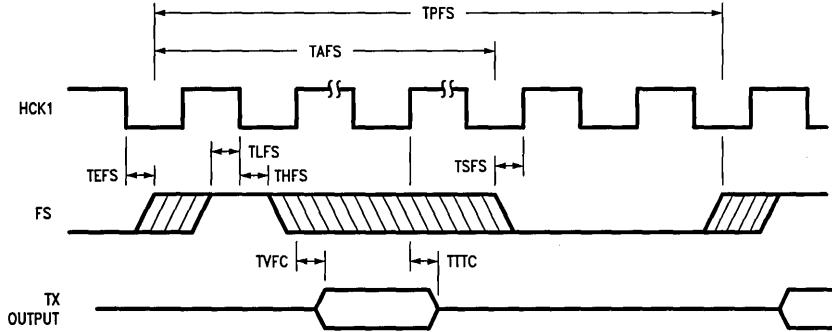
Note 7: This spec is for 1st bit only. Remaining bits are spec'd by transmitter TVTC spec.

Note 8: Receiver specs TVRS and TVRH are required along with TLFS for receiver operation using serial decoder.

Note 9: This spec is for 1st bit only and is measured from the later of either FS or HCK1 rising edge. Remaining bits are spec'd from HCK1 rising edges by transmitter TVTC spec.

Timing Waveforms (Continued)

Serial Decoder Timing Diagram (Modes 5, 6,7)



TL/DD/10422-16

Symbol	Parameter	Min	Max	Comments	Units
TPFS	Number of HCK1 Periods between FS Rising Edges	34			
TAFS	Number of HCK1 Falling Edges during FS High	1	32		
TEFS	Hold of FS Low after HCK1 Falling Edge	10		Early FS	ns
TLFS	Setup of FS Rising Edge before HCK1 Falling Edge	45		Late FS, (Note 8)	ns
TVFC	Delay of TX Output Valid after HCK1 Rising Edge		60	(Note 7)	ns
THFS	Hold of FS High after HCK1 Falling Edge	20			ns
TSFS	Setup of FS Falling Edge before HCK1 Rising Edge	20			ns
TTTC	Delay of TX output TRI-STATE after HCK1 Rising Edge		40		ns

Note 7: This spec is for 1st bit only. Remaining bits are spec'd by transmitter TVTC spec.

Note 8: Receiver specs TVRS and TVRH are required along with TLFS for receiver operation using serial decoder.

Pin Descriptions

I/O PORTS

Port A is a 16-bit multiplexed address/data bus used for accessing external program and data memory. Four associated bus control signals are available on port B. The Address Latch Enable (ALE) signal is used to provide timing to demultiplex the bus. Reading from and writing to external memory are signalled by \overline{RD} and \overline{WR} respectively. External memory can be addressed as either bytes or words with the decoding controlled by two lines, Bus High Byte enable (\overline{HBE}) and Address/Data Line 0 (A0).

Port B is a 16-bit port, with 12 bits of bidirectional I/O. Pins B10, B11, B12 and B15 are the control bus signals for the address/data bus. Port B may also be configured via a function register BFUN to individually allow each bidirectional I/O pin to have an alternate function.

B0:	TDX	UART Data Output
B1:	CFLG1	Closing Flag Output for HDLC #1 Transmitter
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	\overline{HLDA}	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	ALE	Address Latch Enable Output for Address/Data Bus
B11:	\overline{WR}	Address/Data Bus Write Output
B12:	\overline{HBE}	High Byte Enable Output for Address/Data Bus; also 8-Bit Mode Strap Input on Reset.
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	\overline{RD}	Address/Data Bus Read Output

When operating in the extended memory addressing mode, four bits of port B can be used as follows—

B8:	BS0	Memory bank switch output 0 (LSB)
B9:	BS1	Memory bank switch output 1

B13:	BS2	Memory bank switch output 2
B14:	BS3	Memory bank switch output 3 (MSB)

Port I is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

I0:	HCK2	HLDC #2 Clock Input
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture
I3:	INT3	Maskable Interrupt/Input Capture
I4:	INT4/RDY	Maskable Interrupt/Input Capture/Ready Input
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:	HCK1	HDLC #1 Clock and Serial Decoder Clock Input

Port D is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

D0:	REN1/FS/ RHCK1	Receiver #1 Enable/Serial Decoder Frame Sync Input/Receiver #1 Clock Input
D1:	TEN1	Transmitter #1 Enable Input
D2:	REN2/ RHCK2	Receiver #2 Enable Input/Receiver #2 Clock Input
D3:	TEN2	Transmitter #2 Enable Input
D4:	RX1	Receiver #1 Data Input
D5:	TX1	Transmitter #1 Data Output
D6:	RX2	Receiver #2 Data Input
D7:	TX2	Transmitter #2 Data Output

Note: Any of these pins can be read by software. Therefore, unused functions can be used as general purpose inputs, notably external enable lines when the internal serial decoder is used.

Port R is an 8-bit bidirectional I/O port available for general purpose I/O operations. Port R has a direction register to enable each separate pin to be individually defined as an input or output. It has a data register which contains the value to be output. In addition, the Port R pins can be read directly using the Port R pins address.

Pin Descriptions (Continued)

POWER SUPPLIES

- V_{CC1}, V_{CC2} Positive Power Supply (two pins)
- GND Ground for On-Chip Logic
- DGND Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

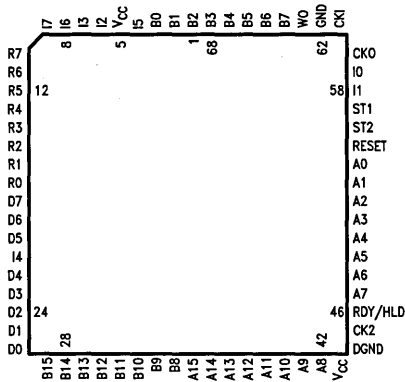
- CKI The System Clock Input
 - CKO The System Clock Output (Inversion of CKI)
- Pins CKI and CKO are usually connected across an external crystal.
- CK2 Clock Output (CKI divided by 2)

OTHER PINS

- W_O This is an active low open drain output which signals an illegal situation has been detected by the Watch Dog logic.
- ST1 Bus Cycle Status Output indicates first op-code fetch.
- ST2 Bus Cycle Status Output indicates machine states (skip and interrupt).
- RESET Active low input that forces the chip to restart and sets the ports in a TRI-STATE mode.
- RDY/HLD Has two uses, selected by a software bit. This pin is either a READY input to extend the bus cycle for slower memories or a HOLD-REQUEST input to put the bus in a high impedance state for external DMA purposes. In the second case the I4 pin can become the READY input.

Connection Diagram

Plastic and Leaded Chip Carriers



TL/DD/10422-18

Top View

See NS Package Number EL68A or V68A

Wait States

The HPC46400E provides software selectable Wait States for access to slower memories and for shared bus applications. The number of Wait States for the CPU are selected by two bits in the PSW register. The number of Wait States for DMA are selected by a bit in the Message System Configuration register. Additionally, the RDY input may be used to extend the RD or WR cycle, allowing the HPC to be used in shared memory applications and allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC46400E: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, on-board RAM, registers and I/O are unaffected (except the HDLC and UART which are reset).

HALT MODE

The HPC46400E is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC46400E are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the RESET or the NMI. The RESET input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC46400E is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. The HPC46400E resumes normal operation upon timer T0 overflow. As with the HALT mode, the processor is also returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization.

HPC46400E Interrupts

Complex interrupt handling is easily accomplished by the HPC46400E's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table I.

TABLE I. Interrupts

Vector/ Address	Interrupt Source	Arbitration Ranking
FFFF FFFE	Reset	0
FFFD FFFC	Nonmaskable Ext (NMI)	1
FFFB FFFA	External on I2	2
FFF9 FFF8	External on I3	3
FFF7 FFF6	External on I4	4
FFF5 FFF4	Internal on Timers	5
FFF3 FFF2	Internal on UART	6
FFF1 FFF0	End of Message (EOM)	7

The 46400E contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. Interrupts are serviced after the current instruction is completed except for the RESET which is serviced immediately.

The NMI interrupt will immediately stop DMA activity. Byte transfers in progress will finish thereby allowing an orderly transition to the interrupt service vector (see DMA description). The HDLC channels continue to operate, and the user must service data errors that might have occurred during the NMI service routine.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately. RESET holds on-chip logic in a reset state while low, and triggers the RESET interrupt on its rising edge. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3, and I4 can be software selected to be rising or falling edge sensitive.

Interrupt Control Registers

The HPC46400E allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the external interrupts are normally cleared by the HPC46400E upon entering the interrupt servicing routine.

For the interrupts from the on-board peripherals, the user has the responsibility of acknowledging the interrupt through software.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable (GIE) bit is reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack, set the GIE bit and return to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. Figure 10 shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets all pins at TRI-STATE except CK0, CK2, and WO. HBE and ST2 have pull-downs designed to withstand override. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFF and resumes operation at the address contained at that memory location.

The RESET pin must be asserted low for at least 16 cycles of the CK2 clock. In applications using the Watchdog feature, RESET should be asserted for at least 64 cycles of the CK2 clock.

On application of power, RESET must be held low for at least five times the power supply rise time to ensure that the on-chip oscillator circuit has time to stabilize.

Timer Overview

The HPC46400E contains a powerful set of flexible timers enabling the HPC46400E to perform extensive timer functions; not usually associated with microcontrollers.

The HPC46400E contains four 16-bit timers. Three of the timers have an associated 16-bit register. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for WATCHDOG logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TOCON configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see Figure 11).

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see Figure 12).

Timer Overview (Continued)

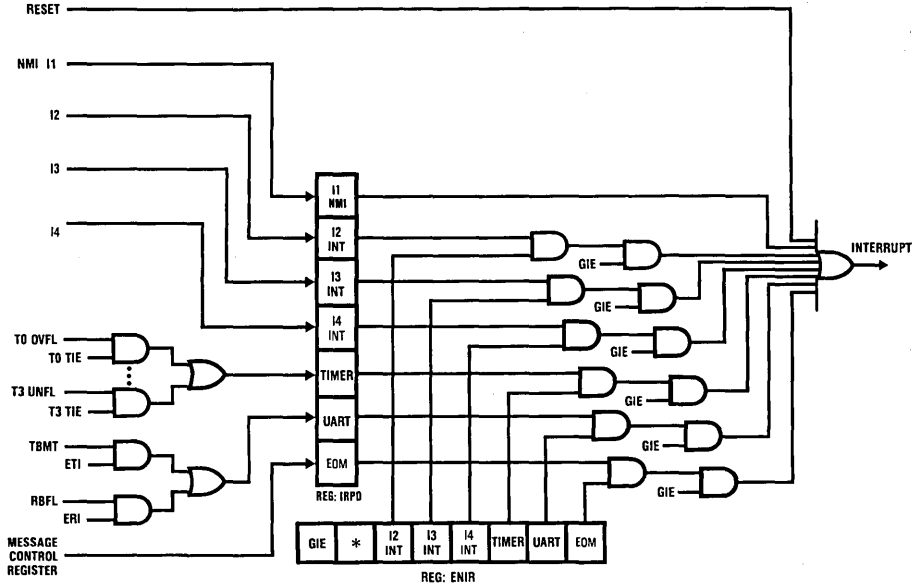


FIGURE 10. Interrupt Enable Logic

TL/DD/10422-19

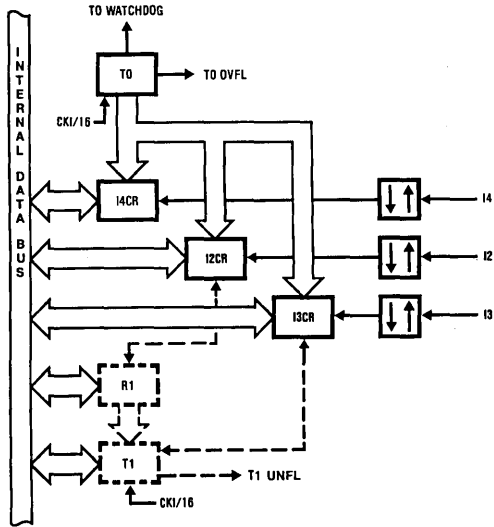


FIGURE 11. Timers T0-T1 Block

TL/DD/10422-21

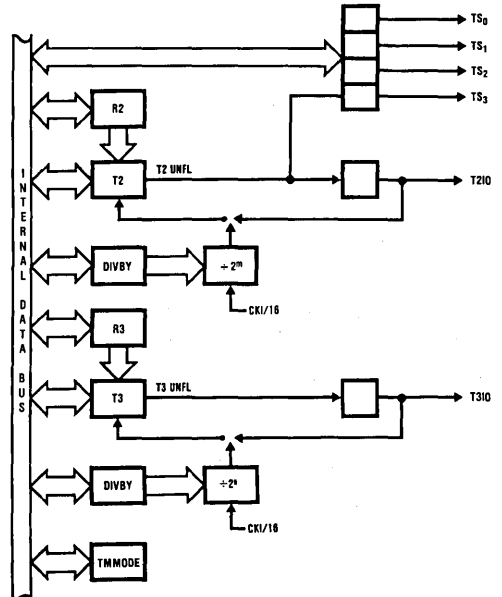


FIGURE 12. Timers T2-T3 Block

TL/DD/10422-20

Timer Overview (Continued)

The timers T1 through T3 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC46400E simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 12*). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to 1/2 the frequency of the source used for clocking the timer.

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC46400E.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

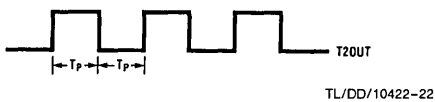


FIGURE 13. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 14* is an example of synchronous pulse train generation.

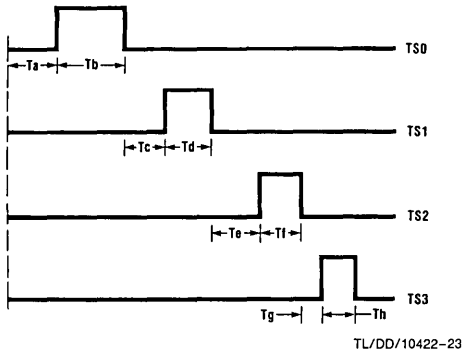


FIGURE 14. Synchronous Pulse Generation

WATCHDOG Logic

The WATCHDOG Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watch Dog logic are potentially infinite loops. Should the Watch Dog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. The illegal condition forces the Watch Out (WO) pin low. The WO pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 15*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., ISDN Transceivers, A/D converters, display drivers, EEPROMs).

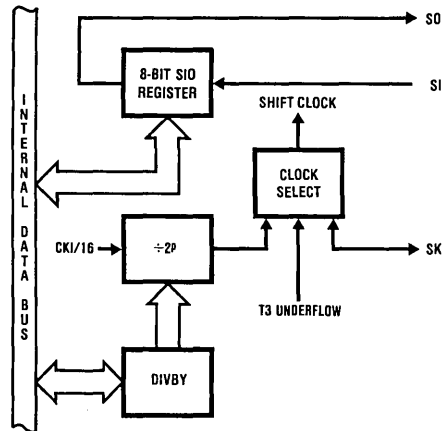


FIGURE 15. MICROWIRE/PLUS

TL/DD/10422-24

MICROWIRE/PLUS Operation

The HPC46400E can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC46400E is the master or slave. The shift clock is generated when the HPC46400E is configured as a master. An externally generated shift clock on the SK pin is used when the HPC46400E is configured as a slave. When the HPC46400E is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz with CKI at 16 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is shifted out on the falling edge of the SK clock. Serial data on the SI pin is latched in on the rising edge of the SK clock.

HPC46400E UART

The HPC46400E contains a software programmable UART. The UART (see Figure 16) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing, parity, and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame, reporting receiving and transmitting status,

and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits ($7/8$, 1, $1\frac{1}{8}$, 2), selecting between the synchronous or asynchronous mode and enabling or disabling transmit and receive interrupts.

The clock inputs to the Transmitter and Receiver sections of the UART can be individually selected to come from either an off-chip source on the CKX pin or one of the three on-chip sources. Presently, two of the on-chip sources, the Divide-By (DIVBY) Register and the Precision UART Timer (PUT), are primarily for reasons of upward compatibility from earlier HPC family members. The most flexible and accurate on-chip clocking is provided by the third source: the Baud Rate Generator (BRG).

The Baud Rate Generator is controlled by the register pair PSR and BAUD, shown below.

The Prescaler factor is selected by the upper 5 bits of the PSR register (the PRESCALE field), in units of the CK2 clock from 1 to 16 in $1/2$ step increments. The lower 3 bits of the PSR register, in conjunction with the 8 bits of the baud register, form the 11-bit BAUDRATE field, which defines a baud rate divisor ranging from 1 to 2048, in units of the prescaled clock selected by the PRESCALE field.

In Asynchronous Mode, the resulting baud rate is $1/16$ of the clocking rate selected through the BRG circuit. The maximum baud rate generated using the BRG is 625 kbaud.

In the Synchronous Mode data is transmitted on the rising edge and received on the falling edge of the external clock. Although the data is transmitted and received synchronously, it is still contained within an asynchronous frame; i.e., a start bit, parity bit (if selected) and stop bit(s) are still present.

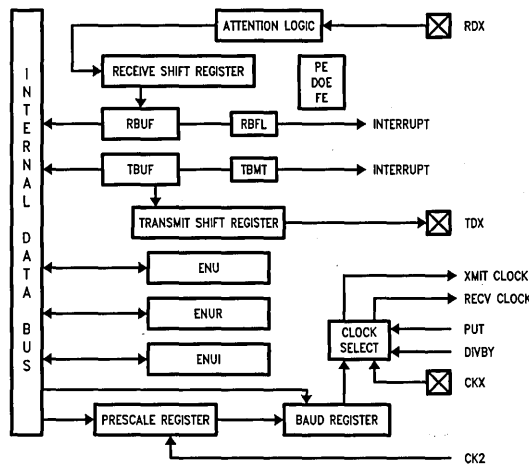
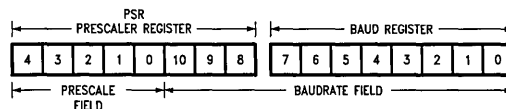


FIGURE 16. UART Block Diagram

TL/DD/10422-25



UART Baud Rate Generator (BRG) Registers PSR and BAUD

TL/DD/10422-26

UART Attention Mode

The HPC46400E UART features an Attention Mode of operation. This mode of operation enables the HPC46400E to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC46400E with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

Programmable Serial Decoder Interface

The programmable serial decoder interface allows the two HDLC channels to be used with devices employing several popular Time Division Multiplexing (TDM) serial protocols for point-to-point and multipoint data exchanges. These protocols combine the 'B' and 'D' channels onto common pins—received data, transmit data, clock and Sync, which normally occurs at an 8 KHz rate and provides framing for the particular protocol.

The decoder uses the serial link clock and Sync signals to generate internal enables for the 'D' and 'B' channels, thereby allowing the HDLC channels to access the appropriate channel data from the multiplexed link.

Additionally, 64 kbit/s to 56 kbits/s rate adaptation can be done using the Serial Decoder generated enable signals B1 or B2. The rate adaption to 56 kbits/s is accomplished by using only the first 7 bits of each B channel time slot for each TDM frame. The transmitter will insert a "1" in the eighth bit of each frame. The receiver will only receive the first seven data bits and skip the eighth bit. See *Figure 17* 65 kbit/56 kbit Rate Adaption Timing Diagram.

HDLC Channel Description

HDLC/DMA Structure

HDLC 1		HDLC 2	
HDLC1 Receive	HDLC1 Transmit	HDLC2 Receive	HDLC2 Transmit
DMAR1	DMAT1	DMAR2	DMAT2

GENERAL INFORMATION

Both HDLC channels on the HPC46400E are identical and operate up to 4.65 Mbps. When used in an ISDN Basic Rate access application, HDLC channel # 1 has been designated for use with the 16 kbps D-channel or either B channel and HDLC #2 can be used with either of the 64 kbps B-channels. If the 'D' and 'B' channels are present on a common serial link, the programmable serial decoder interface generates the necessary enable signals needed to access the D and B channel data.

There are two sources for the receive and transmit channel enable signals. They can be internally generated from the serial decoder interface or they can be externally enabled.

LAPD, the Link Access Protocol for the D channel is derived from the X.25 packet switching LAPB protocol. LAPD specifies the procedure for a terminal to use the D channel for the transfer of call control or user-data information. The pro-

cedure is used in both point-to-point and point-to-multipoint configurations. On the 46400E, the HDLC controller contains user programmable features that allow for the efficient processing of LAPD Information.

HDLC Channel Pin Description

Each HDLC channel has the following pins associated with it.

- HCK — HDLC Channel Clock Input Signal.
- RX — Receive Serial Data Input. Data latched on the negative HCK edge.
- REN/RHCK — HDLC Channel Receiver Enable Input/Receiver Clock Input.
- TEN — HDLC Channel Transmitter Enable Input.
- TX — Transmit Serial Data Output. Data clocked out on the positive HCK edge. Data (not including CRC) is sent LSB first. TRI-STATE when transmitter not enabled.
- CFLG1 — Closing Flag output for Channel 1.

HDLC Functional Description

TRANSMITTER DESCRIPTION

Data is transferred from external memory through the DMA controller into the transmit buffer register, from which it is loaded into a 8-bit serial shift register. The CRC is computed and appended to the frame prior to the closing flag being transmitted. Data is output at the TX output pin. If no further transmit commands are given the transmitter sends out continuous flags, aborts, or the idle pattern as selected by the control register.

An interrupt is generated when the DMA has transferred the last byte from RAM to the HDLC channel for a particular message or on a transmit error condition. An associated transmit status register will contain the status information indicating the specific interrupt source.

To support transmitting data packets at an "R" interface for V.120 in synchronous UI mode, to support the use of the HPC in test equipment, or to support proprietary CRC algorithms the transmitter has the option of preventing the transmitting of the hardware generated CRC bytes.

TRANSMITTER FEATURES

Interframe fill: the transmitter can send either continuous '1's or repeated flags or aborts between the closing flag of one packet and the opening flag of the next. When the CPU commands the transmitter to open a new frame, the interframe fill is terminated immediately.

Abort: the abort sequence, a zero followed by seven ones, will be immediately sent on command from the CPU or on an underrun condition in the DMA.

Bit/Byte boundaries: The message length between packet headers may have any number of bits and is not confined to an integral number of bytes. Three bits in the control register are used to indicate the number of valid bits in the last byte. These bits are loaded by the users software.

RECEIVER DESCRIPTION

Data is input to the receiver on the RX pin. The receive clock can be externally input at either the HCK pin or the REN/RHCK pin.

Incoming data is routed through one of several paths depending on whether it is the flag, data, or CRC.

Once the receiver is enabled it waits for the opening flag of the incoming frame, then starts the zero bit deletion, ad-

HDLC Functional Description (Continued)

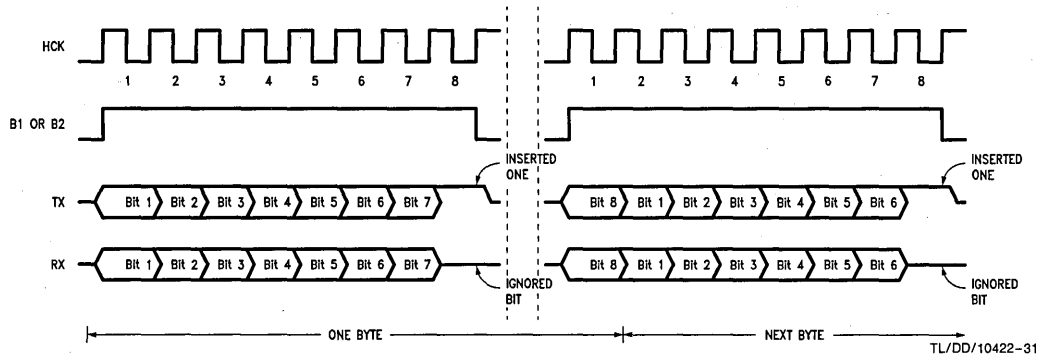


FIGURE 17. 64 kbit/56 kbit Rate Adaption Timing Diagram

addressing handling and CRC checking. All data between the flags is shifted through two 8-bit serial shift registers before being loaded into the buffer register. The user programmable address register values are compared to the incoming data while it resides in the shift registers. If an address match occurs or if operating in the transparent address recognition mode, the DMA channel is signaled that attention is required and the data is transferred by it to external memory. Appropriate interrupts are generated to the CPU on the reception of a complete frame, or on the occurrence of a frame error.

The receive interrupt, in conjunction with status data in the control registers allows interrupts to be generated on the following conditions—frame length error, CRC error, receive error, abort and receive complete.

To support V.120 UI data packets at the "R" interface, proprietary CRC algorithms, and test equipment the two bytes preceding the closing flag (usually the CRC bytes) will be loaded into registers. The two bytes can then be read by the CPU and placed into memory. The DMA address pointers used for that particular message will already contain the address that the first byte should be placed into.

RECEIVER FEATURES

Flag sharing: the closing flag of one packet may be shared as the opening flag of the next. Receiver will also be able to share a zero between flags—0111111011111110 is a valid two flag sequence for receive (not transmit).

Interframe fill: the receiver automatically accepts either repeated flags, repeated aborts, or all '1's as the interframe fill.

Idle: Reception of successive flags as the interframe fill sequence to be signaled to the user by setting the Flag bit in the Receiver Status register.

Short Frame Rejection: Reception of greater than 2 bytes but less than 4 bytes between flags will generate a frame error, terminating reception of the current frame and setting the Frame Error (FER) status bit in the Receive Control and Status register. Reception of less than 2 bytes will be ignored.

Abort: the 7 '1's abort sequence will be immediately recognized and will cause the receiver to reinitialize and return to searching the incoming data for an opening flag. Reception of the abort will cause the abort status bit in the Interrupt Error Status register to be set and will signal an End of Message (EOMR).

Bit/Byte boundaries: The message length between packet headers may have any number of bits and it is not confined to an integral number of bytes. Three bits in the status register are used to indicate the number of valid bits in the last byte.

Address Recognition: Two user programmable bytes are available to allow frame address recognition on the two bytes immediately following the opening flag. When the received address matches the programmed value(s), the frame is passed through to the DMA channel. If no match occurs, the received frame address information is disregarded and the receiver returns to searching for the next opening flag and the address recognition process starts anew.

Support is provided to allow recognition of the Broadcast address. Additionally, a transparent mode of operation is available where no address decoding is done.

HDLC INTERRUPT CONDITIONS

The end of message interrupt (EOM) indicates that a complete frame has been received or transmitted by the HDLC controller. Thus, there are four separate sources for this interrupt, two each from each HDLC channel. The Message Control Register contains the pending bits for each source.

HDLC ERROR DETECTION

The HDLC/DMA detects several error conditions and reports them in the two Error Status Registers. These conditions are a DMA transmitter underrun, a DMA receiver overrun, a CRC error, a frame too long, a frame too short, and an aborted message.

HDLC CHANNEL CLOCK

Each HDLC channel uses the falling edge of the clock to sample the receive data. Outgoing transmit data is shifted out on the rising edge of the external clock. The maximum data rate when using the externally provided clocks is 4.65 Mb/s.

The receiver/transmitter pair can share a single clock input to save I/O pins, or the inputs can be separated to allow different receive and transmit clocks. This feature allows the receiver and transmitter to operate at different frequencies or enables them to each be synchronized to different parts of the user's system.

CYCLIC REDUNDACY CHECK

There are two standard CRC codes used in generating the 16-bit Frame Check Sequence (FCS) that is appended to the end of the data frame. Both codes are supported and

HDLC Functional Description (Continued)

the user selects the error checking code to be used through software control (HDLC control reg). The two error checking polynomials available are:

- (1) CRC-16 ($x^{16} + x^{15} + x^2 + 1$)
- (2) CCITT CRC ($x^{16} + x^{12} + x^5 + 1$)

SYNCHRONOUS BYPASS MODE

When the BYPAS bit is set in the HDLC control register, all HDLC framing/formatting functions for the specified HDLC channel are disabled.

This allows byte-oriented data to be transmitted and received synchronously thus "bypassing" the HDLC functions.

LOOP BACK OPERATIONAL MODE

The user has the ability, by setting the appropriate bit in the register to internally route the transmitter output to the receiver input, and to internally route the RX pin to the TX pin.

DMA Controller

GENERAL INFORMATION

The HPC46400E uses Direct Memory Access (DMA) logic to facilitate data transfer between the 2 full Duplex HDLC channels and external packet RAM. There are four DMA channels to support the four individual HDLC channels. Control of the DMA channels is accomplished through registers which are configured by the CPU. These control registers define specific operation of each channel and changes are immediately reflected in DMA operation. In addition to individual control registers, global control bits (MSS and MSSC in Message Control Register) are available so that the HDLC channels may be globally controlled.

The DMA issues a bus request to the CPU when one or more of the individual HDLC channels request service. Upon receiving a bus acknowledge from the CPU, the DMA completes all requests pending and any requests that may have occurred during DMA operation before returning control to the CPU. If no further DMA transfers are pending, the DMA relinquishes the bus and the CPU can again initiate a bus cycle.

Four memory expansion bits have been added for each of the four channels to support data transfers into the expanded memory bank areas.

The DMA has priority logic for servicing DMA requests. The priorities are:

- 1st priorityReceiver channel 1
- 2nd priorityTransmit channel 1
- 3rd priorityReceive channel 2
- 4th priorityTransmit channel 2

RECEIVER DMA OPERATION

The receiver DMA consists of a shift register and two buffers. A receiver DMA operation is initiated by the buffer registers. Once a byte has been placed in a buffer register from the HDLC, it generates a request and upon obtaining control of the bus, the DMA places the byte in external memory.

RECEIVER REGISTERS

All the following registers are Read/Write

A. Frame Length Register

This user programmable 16-bit register contains the maximum number of bytes to be placed in a data "block". If

this number is exceeded, a Frame Too Long error is generated. DMA is stopped to prevent memory from being overwritten, however the receiver continues until the closing flag is received in order to check the CRC.

B. CNTRL ADDR 1 For split frame operation, the CNTRL ADDR register contains the external memory address where the Frame Header (Control & Address fields) are to be stored and the DATA ADDR register contains an equivalent address for the Information field.

For non-split frame operation, the CNTRL and DATA ADDR registers each contain the external memory address for entire frames.

TRANSMITTER DMA OPERATION

The transmitter DMA consists of a shift register and two buffers. A transmitter DMA cycle is initiated by the TX data buffers. The TX data buffers generate a request when either one is empty and the DMA responds by placing a byte in the buffer. The HDLC transmitter can then accept the byte to send when needed, upon which the DMA will issue another request, resulting in a subsequent DMA cycle.

TRANSMITTER REGISTERS

The following registers are Read/Write:

- FIELD ADDRESS 1 Field Address 1 and Field Address 2 are starting addresses of blocks of information to be transmitted.
- BYTE COUNT 1 Byte Count 1 and Byte Count 2 are the number of bytes in the block to be transmitted.
- FIELD ADDRESS 2
- BYTE COUNT 2

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC46400E supports shared memory access with two pins. The pins are the RDY/HLD input pin and the HLD \bar{A} output pin. The user can software select either the Hold or Ready function on the RDY/HLD pin by the state of a control bit. The HLD \bar{A} output must be selected as the HLD \bar{A} output on pin B7 by software.

The host uses DMA to interface with the HPC46400E. The host initiates a data transfer by activating the HLD input of the HPC46400E. In response, the HPC46400E places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal (HLD \bar{A}) from the HPC46400E indicating that the system bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC46400E resumes normal operations. See *Figure 18* (HPC46400E shared Memory Using HOLD).

An alternate approach is to use the Ready function available on either the RDY/HLD pin or the INT4/RDY pin. See *Figure 19* (HPC46400E Shared Memory Using READY). This technique is often required when the HPC is sharing memory over a system backplane bus.

Shared Memory Support (Continued)

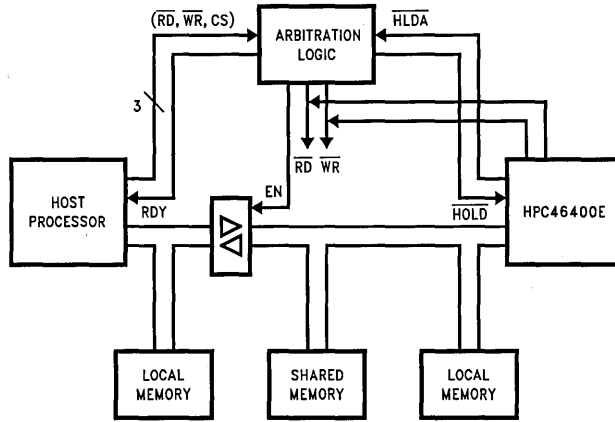


FIGURE 18. HPC46400E Shared Memory Using HOLD

TL/DD/10422-27

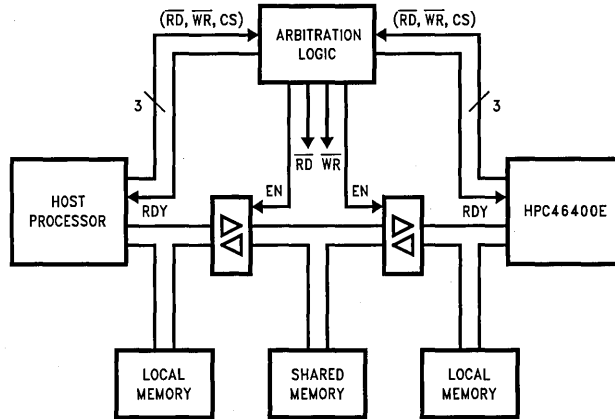


FIGURE 19. HPC46400E Shared Memory Using READY

TL/DD/10422-28

Memory

The HPC46400E has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 256 bytes of RAM available on the chip itself.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always accessed on even-byte boundaries. The HPC46400E uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC46400E memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table II.

Extended Memory Addressing

If more than 64k of addressing is desired in a HPC46400E system, on board bank select circuitry is available that al-

lows four I/O lines of Port B (B8, B9, B13, B14) to be used in extending the address range. This gives the user a main routine area of 32k and 16 banks of 32k each for subroutine and data, thus getting a total of 536.5k of memory.

Note: If all four lines are not needed for memory expansion, the unused lines can be used as general purpose inputs.

The Extended Memory Addressing mode is entered by setting the EMA control bit in the Message Control Register. If this bit is not set, the port B lines (B8, B9, B13, B14) are available as general purpose I/O or synchronous outputs as selected by the BFUN register.

The main memory area contains the interrupt vectors & service routines, stack memory, and common memory for the bank subroutines to use. The 16 banks of memory can contain program or data memory (note: since the on chip resources are mapped into addresses 0000-01FF, the first 512 bytes of each bank are not usable, actual available memory is 536.5k).

TABLE II. Memory Map

FFFF-FFF0 FFEF-FFD0	Interrupt Vectors JSRP Vectors		0159-0158 0157-0156 0155-0154 0153-0152 0151-0150	# Bytes 2 Field Addr 2 # Bytes 1 Field Addr 1 Xmit Cntrl & Status	DMAT # 1 (Xmit)
FFCF-FFCE : : 0201-0200	External Expansion	USER MEMORY	014B-014A 0149-0148 0147-0146 0145-0144 0143-0142 0141-0140	Frame Length Data Addr 2 Cntrl Addr 2 Data Addr 1 Cntrl Addr 1 Recv Cntrl & Status	DMAR # 1 (Recv)
01FF-01FE : : 01C1-01C0	On Chip RAM	USER RAM	012C 012A 0128 0126 0124 0122 0120	Baud PSR - Prescaler ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART
01BC 01BA 01B8 01B6 01B4 01B2 01B0	CRC Byte 2 CRC Byte 1 Error Status Receiver Status Cntrl Recv Addr Comp Reg 2 Recv Addr Comp Reg 1	HDLC # 2	010E 010C 010A 0108 0106 0104 0102 0100	Port R Pins DIR R Register Port R Data Register Message System Configuration Serial Decoder/Enable Configuration Reg Message Pending Message System Control Port D Input	PORTS R & D
01AC 01AA 01A8 01A6 01A4 01A2 01A0	CRC Byte 2 CRC Byte 1 Error Status Receiver Status Cntrl Recv Addr Comp Reg 2 Recv Addr Comp Reg 1	HDLC # 1	00F5-00F4 00F3-00F2 00E6 00E3-00E2	BFUN Register DIR B Register Chip Revision Register Port B	PORT B
0195-0194	Watch Dog Register	Watch Dog Logic	00DD-00DC 00D8 00D6 00D4 00D2 00D0	Halt Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	PORT CONTROL & INTERRUPT CONTROL REGISTERS
0193-0192 0191-0190 018F-018E 018D-018C 018B-018A 0189-0188 0187-0186 0185-0184 0183-0182 0181-0180	TOCON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/ R1 I3CR Register/ T1 I4CR Register	Timer Block T0-T3	00CF-00CE 00CD-00CC 00CB-00CA 00C9-00C8 00C7-00C6 00C5-00C4 00C3-00C2 00C0	X Register B Register K Register A Register PC Register SP Register (Reserved) PSW Register	HPC CORE REGISTERS
017F-017E 017D-017C	Baud Counter Baud Register	UART Timer	00BF-00BE : : 0001-0000	On Chip RAM	USER RAM
0179-0178 0177-0176 0175-0174 0173-0172 0171-0170	Byte Count 2 Field Addr 2 Byte Count 1 Field Addr 1 Xmit Cntrl & Status	DMAT # 2 (Xmit)	016B-016A 0169-0168 0167-0166 0165-0164 0163-0162 0161-0160	Frame Length Data Addr 2 Cntrl Addr 2 Data Addr 1 Cntrl Addr 1 Recv Cntrl & Status	DMAR # 2 (Recv)

Note: All unused addresses are reserved by National Semiconductor

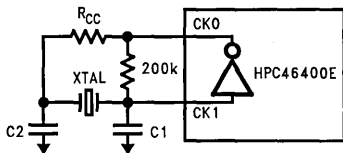
Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage possibly causing internal devices to go into active mode and draw DC current. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused outputs should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- When using local regulators, bypass their inputs with a tantalum capacitor of at least $1 \mu\text{F}$ and bypass their outputs with a $10 \mu\text{F}$ to $50 \mu\text{F}$ tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a $10 \mu\text{F}$ to $20 \mu\text{F}$ tantalum electrolytic capacitor or a $50 \mu\text{F}$ to $100 \mu\text{F}$ aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.



TL/DD/10422-29

A recommended crystal oscillator circuit to be used with the HPC is shown below. See table for recommended component values. The recommended values given in the table below have yielded consistent results and are made to match a crystal with a 20 pF load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within $1''$ distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout should contain a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A $1.0 \mu\text{F}$, a $0.1 \mu\text{F}$, and a $0.001 \mu\text{F}$ dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

HPC Oscillator Table

f_c (MHz)	R_{cc} (Ω)	C_1 (pF)	C_2 (pF)
2	50	82	100
4	50	62	75
6	50	50	56
8	50	47	50
10	50	39	50
12	0	39	39
14	0	33	39
16	0	33	39
18	0	33	33
20	0	33	33

Crystal Specifications:
 "AT" cut, parallel resonant crystals tuned to the desired frequency with the following specifications are recommended:

Series Resistance < 65Ω

Loading Capacitance: $C_L = 20 \text{ pF}$

HPC46400E CPU

The HPC46400E CPU has a 16-bit ALU and six 16-bit registers.

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the stack pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC46400E (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	$MA + Meml \rightarrow MA$ carry $\rightarrow C$
ADDS	Add short imm8	$MA + imm8 \rightarrow MA$ carry $\rightarrow C$
ADC	Add with carry	$MA + Meml + C \rightarrow MA$ carry $\rightarrow C$
DADC	Decimal add with carry	$MA + Meml + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
SUBC	Subtract with carry	$MA - Meml + C \rightarrow MA$ carry $\rightarrow C$
DSUBC	Decimal subtract w/carry	$MA - Meml + C \rightarrow MA$ (Decimal) carry $\rightarrow C$
MULT	Multiply (unsigned)	$MA * Meml \rightarrow MA \& X, 0 \rightarrow K, 0 \rightarrow C$
DIV	Divide (unsigned)	$MA / Meml \rightarrow MA, rem. \rightarrow X, 0 \rightarrow K, 0 \rightarrow C$
DIVD	Divide Double Word (unsigned)	$(x8 MA) / Meml \rightarrow MA, rem \rightarrow X, 0 \rightarrow K$ carry $\rightarrow C$
IFEQ	If equal	Compare MA & Meml, Do next if equal
IFGT	If greater than	Compare MA & Meml, Do next if MA \rightarrow Meml
AND	Logical and	$MA \text{ and } Meml \rightarrow MA$
OR	Logical or	$MA \text{ or } Meml \rightarrow MA$
XOR	Logical exclusive-or	$MA \text{ xor } Meml \rightarrow MA$
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	$Mem + 1 \rightarrow Mem$
DECSZ	Decrement, skip if 0	$Mem - 1 \rightarrow Mem$, Skip next if Mem = 0
BIT INSTRUCTIONS		
SBIT	Set bit	$1 \rightarrow Mem.bit$ (bit is 0 to 7 immediate)
RBIT	Reset bit	$0 \rightarrow Mem.bit$
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	$Meml \rightarrow MA$
ST	Load, incr/decr X	$Mem(X) \rightarrow A, X \pm 1 \text{ (or 2)} \rightarrow X$
X	Store to Memory	$MA \rightarrow Mem$
	Exchange	$A \leftrightarrow Mem; Mem \leftrightarrow Mem$
	Exchange, incr/decr X	$A \leftrightarrow Mem(X), X \pm 1 \text{ (or 2)} \rightarrow X$
PUSH	Push Memory to Stack	$W \rightarrow W(SP), SP + 2 \rightarrow SP$
POP	Pop Stack to Memory	$SP - 2 \rightarrow SP, W(SP) \rightarrow W$
LDS	Load A, incr/decr B, Skip on condition	$Mem(B) \rightarrow A, B \pm 1 \text{ (or 2)} \rightarrow B$, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	$Mem(B) \leftrightarrow A, B \pm 1 \text{ (or 2)} \rightarrow B$, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD A	Load A immediate	$imm \rightarrow A$
LD B	Load B immediate	$imm \rightarrow B$
LD K	Load K immediate	$imm \rightarrow K$
LD X	Load X immediate	$imm \rightarrow X$
LD BK	Load B and K immediate	$imm \rightarrow B, imm \rightarrow K$
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	$0 \rightarrow A$
INC A	Increment A	$A + 1 \rightarrow A$
DEC A	Decrement A	$A - 1 \rightarrow A$
COMP A	Complement A	1's complement of A $\rightarrow A$
SWAP A	Swap nibbles of A	$A15:12 \leftarrow A11:8 \leftarrow A7:4 \leftrightarrow A3:0$
RRC A	Rotate A right thru C	$C \rightarrow A15 \rightarrow \dots \rightarrow A0 \rightarrow C$
RLC A	Rotate A left thru C	$C \leftarrow A15 \leftarrow \dots \leftarrow A0 \leftarrow C$
SHR A	Shift A right	$0 \rightarrow A15 \rightarrow \dots \rightarrow A0 \rightarrow C$
SHL A	Shift A left	$C \leftarrow A15 \leftarrow \dots \leftarrow A0 \leftarrow 0$
SC	Set C	$1 \rightarrow C$
RC	Reset C	$0 \rightarrow C$
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP), SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP), SP+2 → SP, PC+ # → PC (# is +1024 to -1023)
JSRL	Jump subroutine long	PC → W(SP), SP+2 → SP, PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to -31)
JMP	Jump relative	PC+ # → PC(# is +256 to -255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC then Mem(PC)+PC → PC
JIDW		
NOP	No Operation	PC ← PC + 1
RET	Return	SP-2 → SP, W(SP) → PC
RETS	Return then skip next	SP-2 → SP, W(SP) → PC, & skip
RETI	Return from interrupt	SP-2 → SP, W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8-bit or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
MemI is 8-bit or 16-bit memory or 8-bit or 16-bit immediate data
imm is 8-bit or 16-bit immediate data

Memory Usage

For information on memory usage and instruction timing please refer to the HPC46400E User's Manual.

Code Efficiency

The HPC46400E has been designed to be extremely code-efficient. The HPC46400E looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC46400E, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC46400E are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC46400E has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register

3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC46400E supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC46400E has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC46400E has been generally used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.

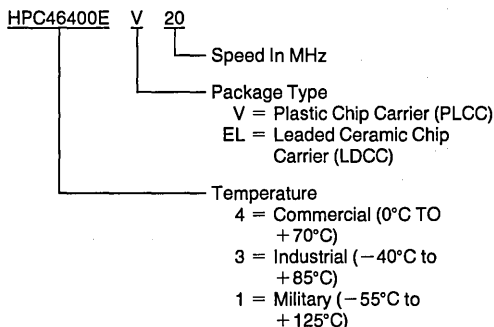


FIGURE 15. HPC Family Part Numbering Scheme

EXAMPLES

HPC46400EV20—Commercial temp (0° to +70°C), PLCC
 HPC36400EV20—Industrial temp (-40°C to +85°C), PLCC

Development Support

HPC MICROCONTROLLER DEVELOPMENT SYSTEM

The HPC microcontroller development system is an in-system emulator (ISE) designed to support the entire family of HPC Microcontrollers. The complete package of hardware and software tools combined with a host system provides a powerful system for design, development and debug of HPC based designs. Software tools are available for IBM®, PC-AT® (MS-DOS, PC-DOS) and for UNIX® based multi-user Sun SPARCstation (SunOS™).

The stand alone unit comes complete with a power supply and external emulation POD. This unit can be connected to various host systems through an RS-232 link. The software package includes an ANSI compatible C-Compiler, Linker, Assembler and librarian package. Source symbolic debug capability is provided through a user friendly MS-windows 3.0 interface for IBM PC-AT environment and through a line debugger under Sunview for Sun SPARCstations.

The ISE provides fully transparent in-system emulation at speeds up to 20 MHz 1 waitstate. A 2K word (48-bit wide) trace buffer gives trace trigger and non intrusive monitoring of the system. External triggering is also available through an external logic interface socket on the POD. Comprehensive on-line help and diagnostics features reduce user's design and debug time. 8 hardware breakpoints (Address/range), 64 kbytes of user memory, and break on external events are some of the other features offered.

Hewlett Packard model HP64775 Emulator/Analyzer providing in-system emulation for up to 30 MHz 1 waitstate is also available. Contact your local sales office for technical details and support.

Development Tools Selection Table

Product	Order Part Number	Description	Included	Manual Number
HPC46400E	HPC-DEV-ISE2	HPC In-System Emulator	HPC MDS User's Manual	420420184-001
	HPC-DEV-ISE2-E	HPC In-System Emulator for Europe and South East Asia	MDS Comm User's Manual HPC46400E User's Manual	424420188-001 420420213-001
	HPC-DEV-IBMA	Assembler/Linker/Library Package	HPC Assembler/Linker Librarian User's Manual for IBM PC-AT	424410836-001
	HPC-DEV-IBMC	C Compiler/Assembler/Linker/Library Package for IBM PC-AT	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-0 424410836-001
	HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC-AT C Compiler/Assembler/Linker Library Package for IBM PC-AT	Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424420189-001 424410883-001 424410836-001
	HPC-DEV-SUNC	C Compiler/Assembler/Linker Library Package for Sun SPARCstation	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	
HPC-DEV-SUNDB	Source/Symbolic Debugger for Sun SPARCstation C Compiler/Assembler/Linker Library Package	Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual		

Development Support (Continued)

Development Tools Selection Table (Continued)

Product	Order Part Number	Description	Included	Manual Number
Complete System				
HPC46400E	HPC-DEV-SYS2	HPC In-System Emulator with C Compiler/Assembler/Linker/Library and Source Symbolic Debugger	HPC Microcontroller Development System User's Manual	420420184-001
	HPC-DEV-SYS2-E	Same for Europe and South East Asia	C-Compiler Manual	424410883-001
			Assembler Manual	424410836-001
			Debugger User's Manual	424420189-001

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. Dial-A-Helper is an electronic bulletin board information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a development system, he can leave messages on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Set-Up: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs, 7 Days



PRELIMINARY

HPC167064/467064 High-Performance microController with a 16k UV Erasable CMOS EPROM

General Description

The HPC167064 is a member of the HPC family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC167064 has a 16 kbyte, high-speed, UV-erasable, electrically programmable CMOS EPROM. This is ideally suited for applications where fast turnaround, pattern experimentation, and code confidentiality are important requirements. The HPC167064 can serve as a stand-alone emulator for either the HPC16064 or the HPC16083. Two configuration registers have been added for emulation of the different chips. The on-chip EPROM replaces the presently available user ROM space. The on-chip EPROM can be programmed via DATA I/O UNISITE, and HPC-MDS. There are security features added to the chip to implement READ, ENCRYPTED READ, and WRITE privileges for the on-chip EPROM. These defined privileges are intended to deter theft, alteration, or unintentional destruction of user code. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, EPROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips.

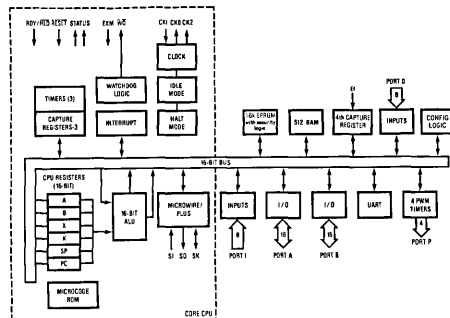
The microCMOS process results in very low current drain and enables the user to select the optimum speed/power

product for his system. The IDLE and HALT modes provide further current savings. The HPC167064 is available only in 68-pin LDCC package.

Features

- HPC family—core features:
 - 16-bit architecture, both byte and word operations
 - 16-bit data bus, ALU, and registers
 - 64 kbytes of direct memory addressing
 - FAST—200 ns for fastest instruction when using 20.0 MHz clock
 - High code efficiency—most instructions are single byte
 - 16 x 16 multiply and 32 x 16 divide
 - Eight vectored interrupt sources
 - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
 - MICROWIRE/PLUS serial I/O interface
 - CMOS—very low power with two power save modes: IDLE and HALT
- 16 kbytes high speed UV erasable: electrically programmable CMOS EPROM
- Stand-alone emulation of HPC16083 and HPC16064 family
- EPROM and configuration bytes programmable by DATA I/O UNISITE with Pinsite Module, MDS
- Four selectable levels of security to protect on-chip EPROM contents
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- Commercial (0°C to +70°C), and military (-55°C to +125°C) temperature ranges

Block Diagram (HPC167064 with 16k EPROM shown)



TL/DD/111046-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

V_{CC} with Respect to GND -0.5V to 7.0V
All Other Pins ($V_{CC} + 0.5V$) to (GND - 0.5V)

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

$V_{CC} = 5.0V \pm 5\%$ unless otherwise specified, $T_A = -55^\circ C$ to $+125^\circ C$ for HPC167064 and $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 0^\circ C$ to $70^\circ C$ for HPC467064

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC1}	Supply Current	$V_{CC} = \max, f_{IN} = 20.0 \text{ MHz}$ (Note 1) $V_{CC} = \max, f_{IN} = 2.0 \text{ MHz}$ (Note 1)		70 40	mA mA
I_{CC2}	IDLE Mode Current	$V_{CC} = \max, f_{IN} = 20.0 \text{ MHz}$, (Note 1) $V_{CC} = \max, f_{IN} = 2.0 \text{ MHz}$, (Note 1)		4.5 1	mA mA
I_{CC3}	HALT Mode Current	$V_{CC} = \max, f_{IN} = 0 \text{ kHz}$, (Note 1) $V_{CC} = 2.5V, f_{IN} = 0 \text{ kHz}$, (Note 1)		400 100	μA μA

INPUT VOLTAGE LEVELS FOR SCHMITT TRIGGERED INPUTS RESET, NMI, AND \overline{WO} ; AND ALSO CK1

V_{IH1}	Logic High		$0.9 V_{CC}$		V
V_{IL1}	Logic Low			$0.1 V_{CC}$	V

ALL OTHER INPUTS

V_{IH2}	Logic High		$0.7 V_{CC}$	*	V
V_{IL2}	Logic Low		*	$0.2 V_{CC}$	V
I_{L11}	Input Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$ (Note 4)		± 2	μA
I_{L12}	Input Leakage Current RDY/HLD, EXUI	$V_{IN} = 0$	-3	-50	μA
I_{L13}	Input Leakage Current B12	RESET = 0, $V_{IN} = V_{CC}$	0.5	7	μA
I_{L14}	Input Leakage Current EXM	$V_{IN} = 0$ and $V_{IN} = V_{CC}$ (Note 4)	± 10		μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF

OUTPUT VOLTAGE LEVELS

V_{OH1} V_{OL1}	Logic High (CMOS) Logic Low (CMOS)	$I_{OH} = -10 \mu A$ (Note 2) $I_{OH} = 10 \mu A$ (Note 2)	$V_{CC} - 0.1$	0.1	V
V_{OH2} V_{OL2}	Port A/B Drive, CK2 (A0-A15, B10, B11, B12, B15)	$I_{OH} = -7 \text{ mA}$ $I_{OL} = 3 \text{ mA}$	2.4	0.4	V
V_{OH3} V_{OL3}	Other Port Pin Drive, \overline{WO} (open drain) (B0-B9, B13, B14, P0-P3)	$I_{OH} = -1.6 \text{ mA}$ (except \overline{WO}) $I_{OL} = 0.5 \text{ mA}$	2.4	0.4	V
V_{OH4} V_{OL4}	ST1 and ST2 Drive	$I_{OH} = -6 \text{ mA}$ $I_{OL} = 1.6 \text{ mA}$	2.4	0.4	V
V_{OH5} V_{OL5}	Port A/B Drive (A0-15, B10, B11, B12, B15) when used as External Address/Data Bus	$I_{OH} = -1 \text{ mA}$ $I_{OL} = 3 \text{ mA}$	2.4	0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE® Leakage Current	$V_{IN} = 0$ and $V_{IN} = V_{CC}$		± 5	μA

Note 1: I_{CC1} , I_{CC2} , I_{CC3} measured with no external drive (I_{OH} and $I_{OL} = 0$, I_{IH} and $I_{IL} = 0$). I_{CC1} is measured with RESET = GND. I_{CC3} is measured with NMI = V_{CC} . CK1 driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

Note 4: The EPROM mode of operation for this device requires high voltage input on pins EXM/ V_{PP} , I3, I4, I5, I6 and I7. This will increase the input leakage current above the normal specification when driven to voltages greater than $V_{CC} + 0.3V$.

*See NORMAL RUNNING MODE.

20 MHz

AC Electrical Characteristics

(See Notes 1 and 4 and Figures 1 thru 5). $V_{CC} = 5V \pm 5\%$, $T_A = -55^\circ\text{C}$ to $+125^\circ\text{C}$ for HPC167064 and $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ for HPC467064

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Clocks	f_C	CKI Operating Frequency	2	20	MHz	
	$t_{C1} = 1/f_C$	CKI Clock Period	50	500	ns	
	t_{CKIH}	CKI High Time	22.5		ns	
	t_{CKIL}	CKI Low Time	22.5		ns	
	$t_C = 2/f_C$	CPU Timing Cycle	100		ns	
	$t_{WAIT} = t_C$	CPU Wait State Period	100		ns	
	t_{DC1C2R}	Delay of CK2 Rising Edge after CK1 Falling Edge	0	55	ns	(Note 2)
	t_{DC1C2F}	Delay of CK2 Falling Edge after CK1 Falling Edge	0	55	ns	(Note 2)
	$f_U = f_C/8$	External UART Clock Input Frequency		2.5**	MHz	
	f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz	
Timers	$f_{XIN} = f_C/22$	External Timer Input Frequency	100	0.91	MHz	
	$t_{XIN} = t_C$	Pulse Width for Timer Inputs			ns	
Microwire/Plus	t_{UWS}	MICROWIRE Setup Time—Master MICROWIRE Setup Time—Slave	100 20		ns	
	t_{UWH}	MICROWIRE Hold Time—Master MICROWIRE Hold Time—Slave	20 50		ns	
	t_{UWV}	MICROWIRE Output Valid Time—Master MICROWIRE Output Valid Time—Slave		50 150	ns	
External Hold	$t_{SALE} = \frac{3}{4} t_C + 40$	\overline{HLD} Falling Edge before \overline{ALE} Rising Edge	115		ns	
	$t_{HWP} = t_C + 10$	\overline{HLD} Pulse Width	110		ns	
	$t_{HAE} = t_C + 100$	\overline{HLDA} Falling Edge after \overline{HLD} Falling Edge		200*	ns	(Note 3)
	$t_{HAD} = \frac{3}{4} t_C + 85$	\overline{HLDA} Rising Edge after \overline{HLD} Rising Edge		160	ns	
	$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after \overline{HLDA} Falling Edge		116	ns	(Note 5)
	$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable after \overline{HLDA} Rising Edge	116		ns	(Note 5)
UPI Timing	t_{UAS}	Address Setup Time to Falling Edge of \overline{URD}	10		ns	
	t_{UAH}	Address Hold Time from Rising Edge of \overline{URD}	10		ns	
	t_{RPW}	\overline{URD} Pulse Width	100		ns	
	t_{OE}	\overline{URD} Falling Edge to Output Data Valid	0	60	ns	
	t_{OD}	Rising Edge of \overline{URD} to Output Data Invalid	5	35	ns	
	t_{DRDY}	\overline{RDRDY} Delay from Rising Edge of \overline{URD}		70	ns	
	t_{WDW}	\overline{UWR} Pulse Width	40		ns	
	t_{UDS}	Input Data Valid before Rising Edge of \overline{UWR}	10		ns	
	t_{UDH} (HPC467064)	Input Data Hold after Rising Edge of \overline{UWR}	20		ns	
	t_{UDH} (HPC167064)		25*		ns	
	\overline{WRRDY} Delay from Rising Edge of \overline{UWR}		70	ns		

*See NORMAL RUNNING MODE.

**This maximum frequency is attainable provided that this external baud clock has a duty cycle such that the high period includes two (2) falling edges of the CK2 clock.

Note: $C_L = 40$ pF.

Note 1: These AC Characteristics are guaranteed with external clock drive on CKI having 50% duty cycle and with less than 15 pF load on CKO with rise and fall times (t_{CKIR} and t_{CKIL}) on CKI input less than 2.5 ns.

Note 2: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CKI or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{HLD} falling edge occurring at the latest time can be accepted during the present CPU cycle being executed. If \overline{HLD} falling edge occurs later, t_{HAE} may be as long as $(3t_C + 4 WS + 72t_C + 100)$ depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: $WS = t_{WAIT} \times$ (number of pre-programmed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 20.00$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

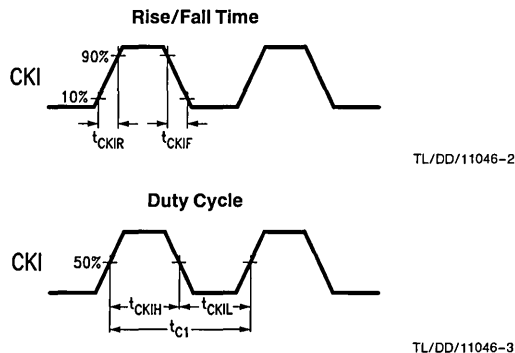
20 MHz

AC Electrical Characteristics (Continued)

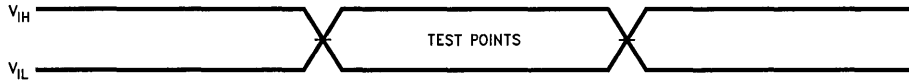
(See Notes 1 and 4 and Figures 1 thru 5.) $V_{CC} = 5V \pm 5\%$, $T_A = -55^\circ C$ to $+125^\circ C$ for HPC167064 and $V_{CC} = 5V \pm 10\%$, $T_A = 0^\circ C$ to $+70^\circ C$ for HPC467064 (Continued)

	Symbol and Formula	Parameter	Min	Max	Units	Notes
Address Cycles	$t_{DC1ALER}$	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns	(Note 2)
	$t_{DC1ALEF}$	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns	(Note 2)
	$t_{DC2ALER} = \frac{1}{4} t_C + 20$	Delay from CK2 Rising Edge to ALE Rising Edge		45	ns	
	$t_{DC2ALEF} = \frac{1}{4} t_C + 20$	Delay from CK2 Falling Edge to ALE Falling Edge		45	ns	
	$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns	
	$t_{ST} = \frac{1}{4} t_C - 7$	Setup of Address Valid before ALE Falling Edge	18		ns	
	$t_{VP} = \frac{1}{4} t_C - 5$	Hold of Address Valid after ALE Falling Edge	20		ns	
Read Cycles	$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to \overline{RD} Falling Edge	20		ns	
	$t_{ACC} = t_C + WS - 55$	Data Input Valid after Address Output Valid		145	ns	
	$t_{RD} = \frac{1}{2} t_C + WS - 65$	Data Input Valid after \overline{RD} Falling Edge		85	ns	
	$t_{RW} = \frac{1}{2} t_C + WS - 10$	\overline{RD} Pulse Width	140		ns	
	$t_{DR} = \frac{3}{4} t_C - 15$	Hold of Data Input Valid after \overline{RD} Rising Edge	0	60	ns	
	$t_{RDA} = t_C - 15$	Bus Enable after \overline{RD} Rising Edge	85		ns	
Write Cycles	$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to \overline{WR} Falling Edge	45		ns	
	$t_{WW} = \frac{3}{4} t_C + WS - 15$	\overline{WR} Pulse Width	160		ns	
	$t_V = \frac{1}{2} t_C + WS - 5$	Data Output Valid before \overline{WR} Rising Edge	145		ns	
	$t_{HW} = \frac{1}{4} t_C - 5$	Hold of Data Valid after \overline{WR} Rising Edge	20		ns	
Ready Input	$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of \overline{RDY}		75	ns	
	$t_{RWR} = t_C$	\overline{RDY} Pulse Width	100		ns	

CKI Input Signal Characteristics



CKI Input Signal Characteristics

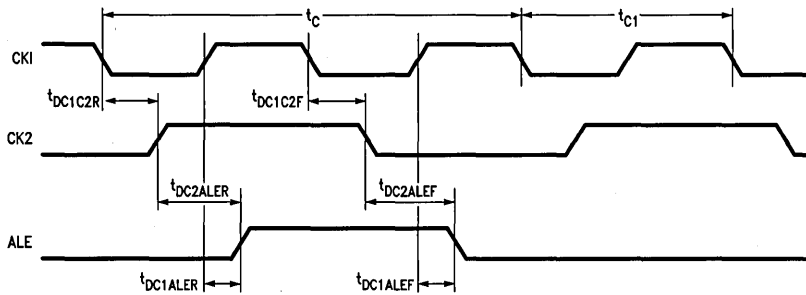


TL/DD/11046-4

Note: AC testing inputs are driven at V_{IH} for logic "1" and V_{IL} for a logic "0". Output timing measurements are made at $V_{CC}/2$ for both logic "1" and logic "0".

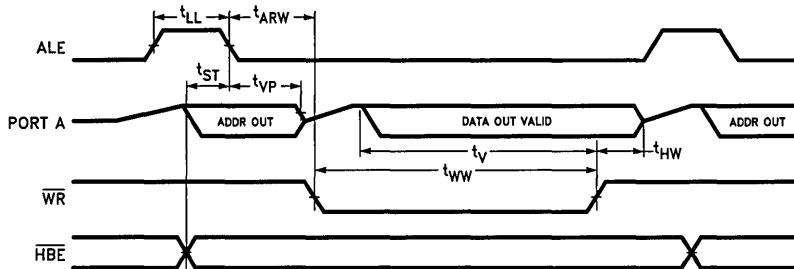
FIGURE 2. Input and Output for AC Tests

Timing Waveforms



TL/DD/11046-5

FIGURE 3. CK1, CK2, ALE Timing Diagram



TL/DD/11046-6

FIGURE 4. Write Cycle

Timing Waveforms (Continued)

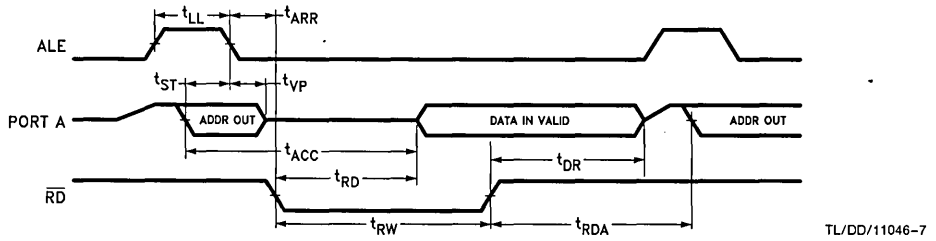


FIGURE 5. Read Cycle

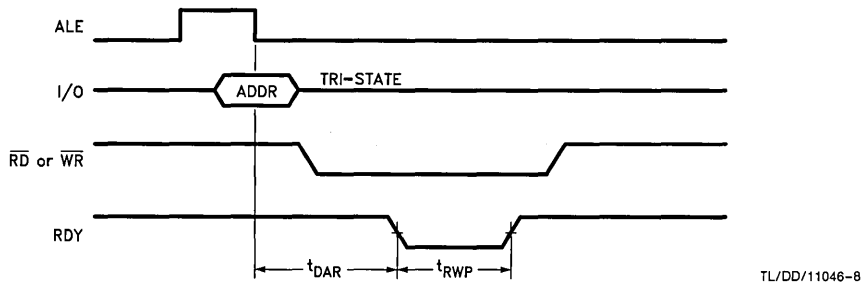


FIGURE 6. Ready Mode Timing

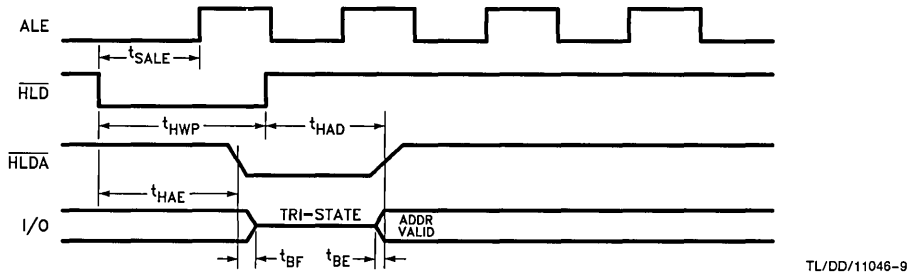
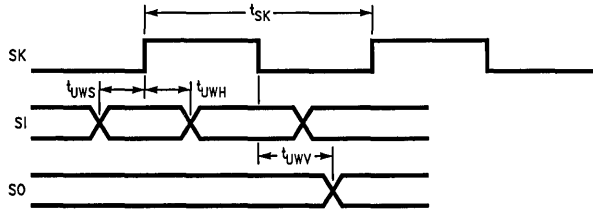


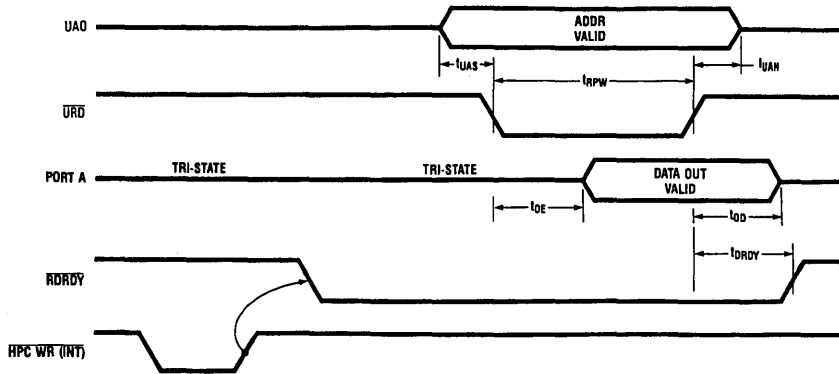
FIGURE 7. Hold Mode Timing

Timing Waveforms (Continued)



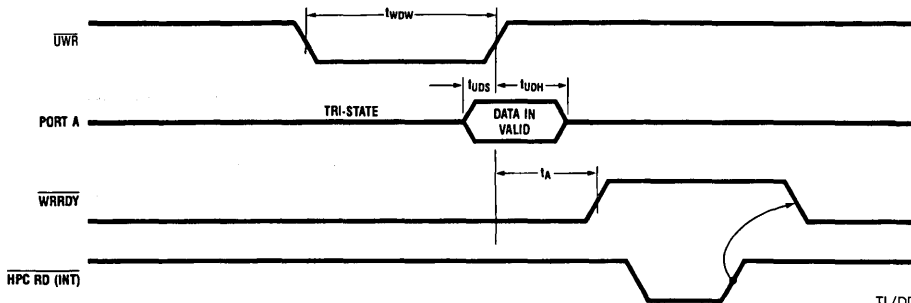
TL/DD/11046-10

FIGURE 8. MICROWIRE Setup/Hold Timing



TL/DD/11046-11

FIGURE 9. UPI Read Timing



TL/DD/11046-12

FIGURE 10. UPI Write Timing

Functional Modes of Operation

There are two primary functional modes of operation for the HPC167064.

- EPROM Mode
- Normal Running Mode

EPROM MODE

In the EPROM mode, the HPC167064 is configured to "approximately emulate" a standard NMC27C256 EPROM. Some dissimilarities do exist. The most significant one is that HPC167064 contains only 16 kbytes of programmable memory, rather than the 32 kbytes in 27C256. An HPC167064 in the EPROM mode can be programmed with a Data I/O machine or an HPC-MDS using a programming adapter board.

Given below is the list of functions that can be performed by the user in the EPROM mode.

- Programming

CAUTION: Exceeding 14V on pin 1 (V_{PP}) will damage the HPC167064.

Initially, and after each erasure, all bits of the HPC EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be presented in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

- Program/verify EPROM registers

To read data (verify) during the programming process, V_{PP} must be at 13V. When reading data after the programming process, V_{PP} can be either 13V or at V_{CC} .

- Program/verify ECON registers

There are two configuration registers ECON6 and ECON7 to emulate different family members and also to enable/disable different features in the chip. These registers are not mapped in the EPROM user space. These bytes must be programmed through a pointer register ECONA.

To prevent unintentional programming, the ECON6, 7 registers must be programmed with the assistance of this pointer register. ECONA, and externally presented address, both identify the same ECON register may be programmed.

NORMAL RUNNING MODE

In this mode, the HPC167064 executes user software in the normal manner. By default, its architecture imitates that of the HPC16064. It may be configured to emulate the HPC16083. The addressable memory map will be exactly as for the HPC16083. The WATCHDOG function monitors addresses accordingly. Thus, the HPC167064 can be used as a stand-alone emulator for both HPC16064 and HPC16083.

Within this mode, the on-chip EPROM cell acts as read only memory. Each memory fetch is 16-bits wide. The HPC167064 operates to 20 MHz with 1 wait state for the on-chip memory.

The HPC167064 emulates all functional modes of operation for the HPC16064 and HPC16083, except as described here.

- The value of EXM is latched on the rising edge of RESET. Thus, the user may not switch from ROMed to ROMless operation or vice-versa, without another RESET pulse.
- The security logic can be used to control access to the on-chip EPROM. This feature is unique to the HPC167064. There is no corresponding mode of operation on the HPC16064 or the HPC16083.
- Specific inputs are allowed to be driven at high voltage (13V) to configure the device for programming. These high voltage inputs are unique to the HPC167064. The same inputs cannot be driven to high voltage on the HPC16064 and HPC16083 without damage to the part.
- The Port D input structure on this device is slightly different from the masked ROM HPC16083 and HPC16064. V_{IH2} min and V_{IL2} max are the same as for the masked ROM HPC16083 and HPC16064. There is a V_{IH2} max requirement for this device equal to $V_{CC} + 0.05V$. There is also a V_{IL2} min requirement for this device equal to $GND-0.05V$. The V_{IH2} max and V_{IL2} min requirement for the masked ROM devices is the Absolute Maximum Ratings of $V_{CC} + 0.5V$ and $GND-0.5V$ respectively.
- The D.C. Electrical Characteristics and A.C. Electrical Characteristics for the HPC167064, where $T_A = -55^{\circ}C$ to $+125^{\circ}C$, are guaranteed over a reduced operating voltage range of $V_{CC} \pm 5\%$. This is different from the masked ROM devices that it simulates which is $V_{CC} \pm 10\%$. These characteristics for the HPC467064, where $T_A = -0^{\circ}C$ to $+70^{\circ}C$, are guaranteed over the masked ROM operating voltage range which is $V_{CC} \pm 10\%$.
- In addition to the reduced operating voltage range for the HPC167064, the A.C. timing parameter t_{UDH} is required to be a minimum value of 25 ns. The masked ROM devices require a minimum t_{UDH} of 20 ns. This A.C. timing parameter for the HPC467064 is required to be the same as the masked ROM devices.

HPC167064 EPROM SECURITY

The HPC167064 includes security logic to provide READ and WRITE protection of the on-chip EPROM. These defined privileges are intended to deter theft, alteration, or unintentional destruction of user code. Two bits are used to define four levels of security on the HPC167064 to control access to on-chip EPROM.

Security Level 3

This is the default configuration of an erased HPC167064. READ and WRITE accesses to the on-chip EPROM or ECON registers may be accomplished without constraint in EPROM mode. READ accesses to the on-chip EPROM may be accomplished without constraint in NORMAL RUNNING mode.

Functional Modes of Operation (Continued)

Security Level 2

This security level prevents programming of the on-chip EPROM or the ECON registers thereby providing WRITE protection. Read accesses to the on-chip EPROM or ECON registers may be accomplished without constraint in EPROM mode. Read accesses to the on-chip EPROM may be accomplished without constraint in NORMAL RUNNING mode.

Security Level 1

This security level prevents programming of the on-chip EPROM or ECON registers—thereby providing registers write protection. Read accesses to the on-chip ECON-registers may be accomplished without constraint in EPROM mode. Read accesses to the on-chip EPROM will produce ENCRYPTED data in EPROM. READ accesses to the on-chip EPROM, during NORMAL RUNNING mode, are subject to **Runtime Memory Protection**. Under Runtime Memory Protection, only instruction opcodes stored within the on-chip EPROM are allowed to access the EPROM as operand. If any other instruction opcode attempts to use the contents of EPROM as an operand, it will receive the hex value "FF". The Runtime Memory Protection feature is designed to prevent hostile software, running from external memory or on-chip RAM, from reading secured EPROM data. Transfers of control into, or out of the on-chip EPROM (such as jump or branch) are not affected by Runtime Memory Protection. Interrupt vector fetches from EPROM proceed normally, and are not affected by Runtime Memory Protection.

Security Level 0

This security level prevents programming of the on-chip EPROM or ECON registers, thereby providing write protection. Read accesses to the on-chip ECON registers may be accomplished without constraint in EPROM mode. READ accesses to the on-chip EPROM are NOT ALLOWED in EPROM mode. Such accesses will return data value "FF" hex. Runtime Memory Protection is enforced as in security level 1.

These four levels of security help ensure that the user EPROM code is not tampered with in a test fixture and that code executing from RAM or external memory does not dump the user algorithm.

Erasure Characteristics

The erasure characteristics of the HPC167064 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range.

After programming, opaque labels should be placed over the HPC167064's window to prevent unintentional erasure. Covering the window will also prevent temporary functional failure due to the generation of photo currents.

The recommended erasure procedure for the HPC167064 is exposure to short wave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 30W-sec/cm².

The HPC167064 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. The erasure time table shows the minimum HPC167064 erasure time for various light intensities.

An erasure system should be calibrated periodically. The distance from lamp to unit should be maintained at one inch. The erasure time increases as the square of the distance. (If distance is doubled the erasure time increases by a factor of 4.) Lamps lose intensity as they age. When a lamp is changed, the distance has changed or the lamp has aged, the system should be checked to make certain full erasure is occurring.

Incomplete erasure will cause symptoms that can be misleading. Programmers, components, and even system designs have been erroneously suspected when incomplete erasure was the problem.

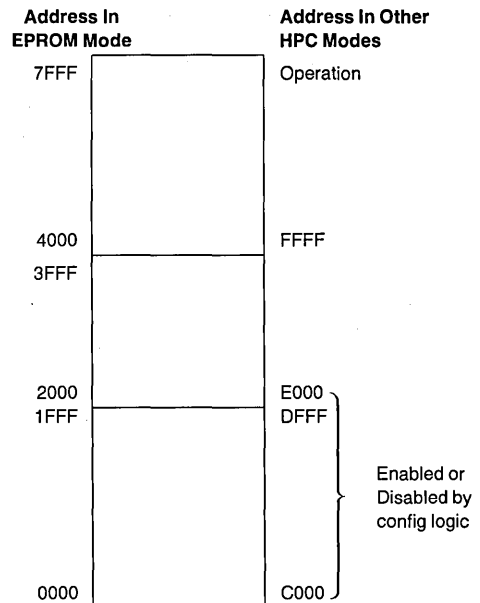
Minimum HPC167064 Erasure Time

Light Intensity (Micro-Watts/cm ²)	Erasure Time (Minutes)
15,000	36
10,000	50

Memory Map of the HPC167064

The HPC167064 has 256 bytes of on-chip user RAM and chip registers located at address 0000–01FF that is always enabled, and 256 bytes of on-chip RAM located at 0200–02FF that can be enabled or disabled. It has 8 kbytes of on-chip EPROM located at address 0E000–0FFFF that is always enabled and 8 kbytes of EPROM located at address 0C000–0DFFF that can be enabled or disabled.

The ECON6 contains two bits ROM0 and RAM0. When these bits are "1" (erased default), full 16 kbytes of ROM and 512 bytes of RAM are enabled. Programming a "0" to these bits disables the lower 8k for the EPROM and upper 256 bytes for the RAM. The ECON registers are only accessible to the user during EPROM mode.



Pin Descriptions

The HPC167064 is available only in 68-pin LDCC package.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{FLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	$\overline{\text{WRRDY}}$	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RDRDY}}$	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	$\overline{\text{WR}}$	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output/Input (sampled at reset)
B15:	$\overline{\text{RD}}$	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/ $\overline{\text{URD}}$
I3:	INT3	Maskable Interrupt/Input Capture/ $\overline{\text{UWR}}$
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

VCC1 and VCC2	Positive Power Supply
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: There are two electrically connected VCC pins on the chip, GND and DGND are electrically isolated. Both VCC pins and both ground pins must be used.

CLOCK PINS

CKI	The Chip System Clock Input
CKO	The Chip System Clock Output (inversion of CKI)

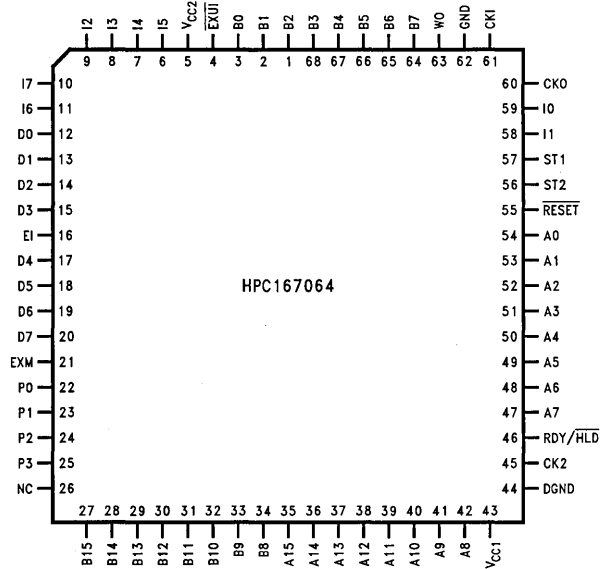
Pins CKI and CKO are usually connected across an external crystal.

CK2	Clock Output (CKI divided by 2)
-----	---------------------------------

OTHER PINS

$\overline{\text{WO}}$	This is an active low open drain output that signals an illegal situation has been detected by the WATCHDOG logic.
ST1	Bus Cycle Status Output: indicates first opcode fetch.
ST2	Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
$\overline{\text{RESET}}$	is an active low input that forces the chip to re-start and sets the ports in a TRI-STATE mode.
RDY/ $\overline{\text{HLD}}$	has two uses, selected by a software bit. It's either an input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.
NC	(no connection) do not connect anything to this pin.
EXM	Has two uses. External memory enable (active high) which disables internal EPROM and maps it to external memory, and is V _{PP} during EPROM mode.
EI	External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.
$\overline{\text{EXUI}}$	External interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 (Active Low).

Connection Diagram



TL/DD/11046-17

Top View
Order Number HPC167064, EL
See NS Package Number EL68C

Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 10*), consists of a data register and a direction register. Port B (see *Figures 11* thru *Figure 13*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.

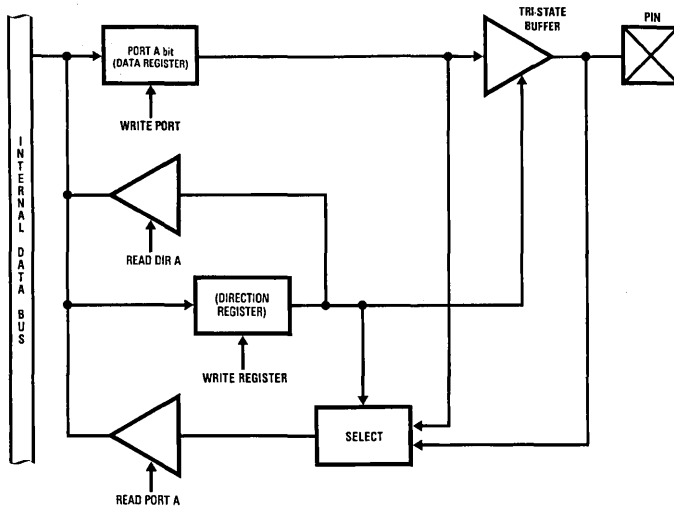
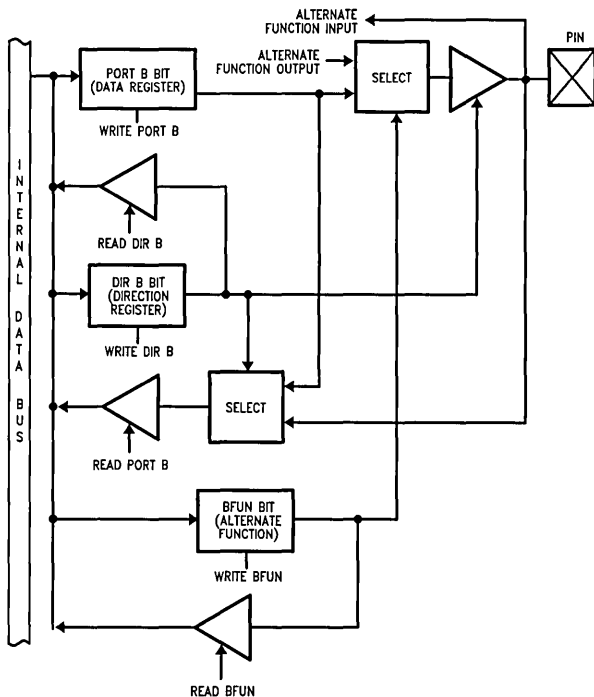


FIGURE 11. Port A: I/O Structure

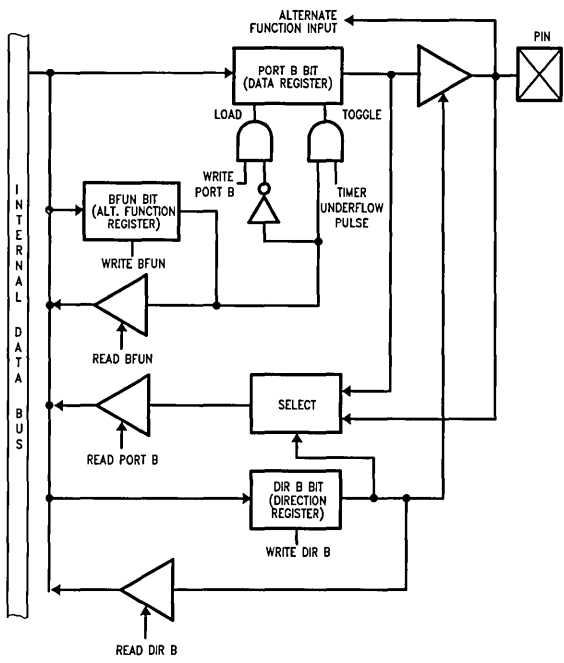
TL/DD/11046-19

Ports A & B (Continued)



TL/DD/11046-20

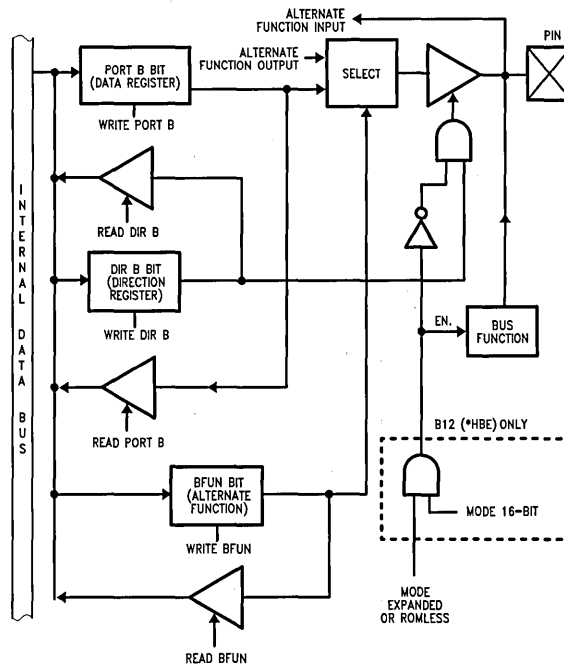
FIGURE 12. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)



TL/DD/11046-21

FIGURE 13. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)

Ports A & B (Continued)



TL/DD/11046-22

FIGURE 14. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC167064 has four operating modes. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip EPROM will be accessed or external memory will be accessed within the address range of the on-chip EPROM. The on-chip EPROM range of the HPC167064 is C000 to FFFF (16 kbytes).

A logic "0" state on the EXM pin will cause the HPC device to address on-chip EPROM when the Program Counter (PC) contains addresses within the on-chip EPROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip EPROM addresses. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip EPROM range and on-chip RAM and Register range, and the "illegal address detection" feature of the WATCH-

DOG logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64 kbytes address range and the "illegal address detection" feature of the WATCHDOG logic is disabled.

All HPC devices can be used with external memory. External memory may be any combination of RAM and EPROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, Port A becomes the Address/Data bus. Four pins of Port B become the control lines ALE, \overline{RD} , \overline{WR} and \overline{HBE} . The High Byte Enable pin (\overline{HBE}) is used in 16-bit mode to select high order memory bytes. The \overline{RD} and \overline{WR} signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling \overline{HBE} high at reset. If \overline{HBE} is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC167064.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC167064 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC167064 functions as a self-contained microcomputer (see *Figure 14*) with all memory (RAM and EPROM) on-chip. It can address internal memory only, consisting of 16 kbytes of EPROM (C000 to FFFF) and 512 bytes of on-chip RAM and Registers (0000 to 02FF). The "illegal address detection" feature of the WATCHDOG is enabled in the Single-Chip Normal mode and a WATCHDOG Output (\overline{WO}) will occur if an attempt is made to access addresses that are outside of the on-chip EPROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC167064 to address external memory in addition to the on-chip ROM and RAM (see Table I). WATCHDOG illegal address detection is disabled and memory accesses may be made anywhere in the 64 kbyte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

TABLE I. HPC167064 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	C000–FFFF On-Chip
Expanded Normal	0	1	C000–FFFF On-Chip 0300–BFFF Off-Chip
Single-Chip ROMless	1	0	C000–FFFF Off-Chip
Expanded ROMless	1	1	0300–FFFF Off-Chip

SINGLE-CHIP ROMless MODE

In this mode, the on-chip EPROM of the HPC167064 is not used. The address space corresponding to the on-chip EPROM is mapped into external memory so 16k of external memory may be used with the HPC167064 (see Table I). The WATCHDOG circuitry detects illegal addresses (addresses not within the on-chip EPROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

EXPANDED ROM MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64 kbytes of external memory may be used. The "illegal address detection" feature of WATCHDOG is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

Wait States

The internal EPROM can be accessed at the maximum operating frequency with one wait state. With 0 wait states, internal ROM accesses are limited to $\frac{2}{3} f_C$ max. The HPC167064 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

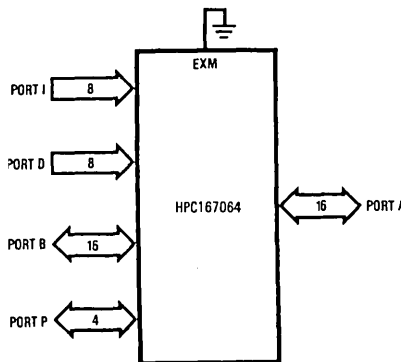


FIGURE 15. Single-Chip Mode

TL/DD/11046-23

Power Save Modes

Two power saving modes are available on the HPC167064: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC167064 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC167064 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the \overline{RESET} or the NMI. The \overline{RESET} input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC167064 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the \overline{RESET} or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC167064 to resume normal operation.

Note: If an NMI interrupt is received during the instruction which puts the device in Halt or Idle Mode, the device will enter that power saving mode. The interrupt will be held pending until the device exits that power saving mode. When exiting Idle mode via the T0 overflow, the NMI interrupt will be serviced when the device exits Idle. If another NMI interrupt is received during either Halt or Idle the processor will exit the power saving mode and vector to the interrupt address.

HPC167064 Interrupts

Complex interrupt handling is easily accomplished by the HPC167064's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table II.

HPC167064 Interrupts (Continued)

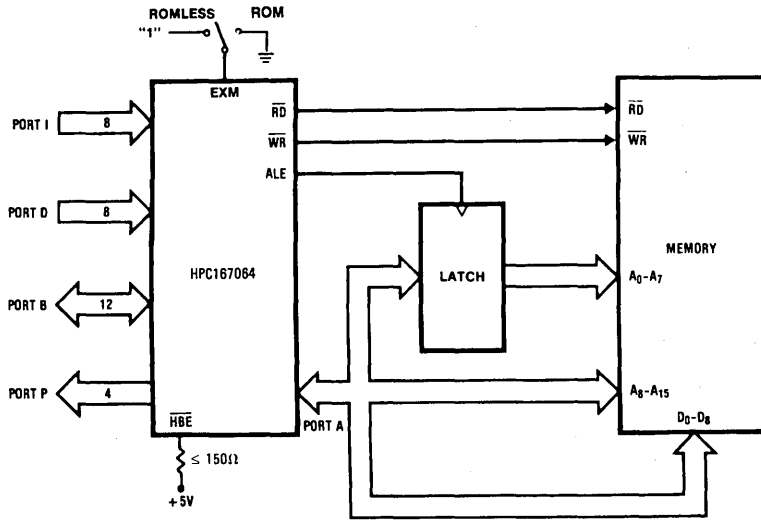


FIGURE 16. 8-Bit External Memory

TL/DD/11046-24

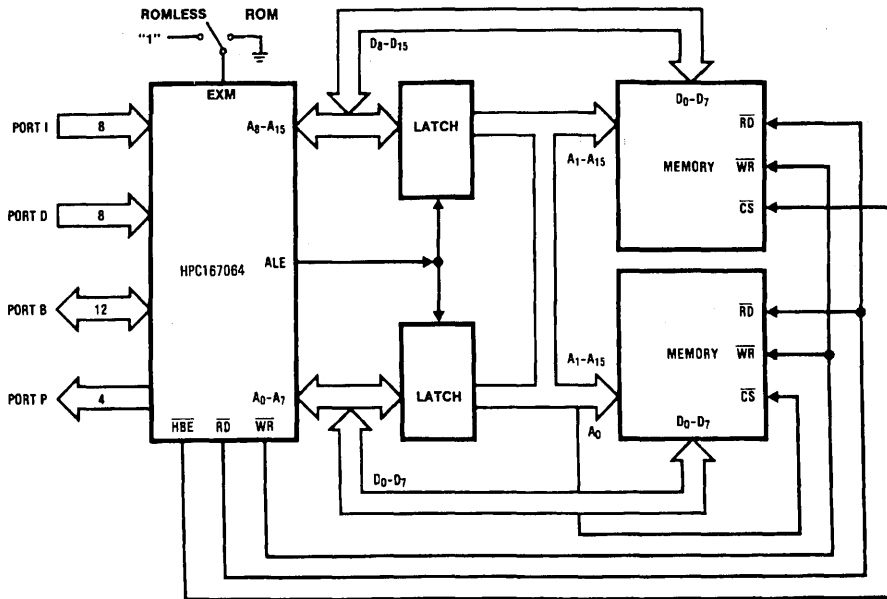


FIGURE 17. 16-Bit External Memory

TL/DD/11046-25

HPC167064 Interrupts (Continued)

TABLE II. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
FFFF:FFFE	RESET	0
FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
FFFB:FFFA	External interrupt on I2 pin	2
FFF9:FFF8	External interrupt on I3 pin	3
FFF7:FFF6	External interrupt on I4 pin	4
FFF5:FFF4	Overflow on internal timers	5
FFF3:FFF2	Internal on the UART transmit/receive complete or external on EXUI	6
FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC167064 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table II. The interrupt on RESET has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately. RESET and EXUI are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt (EXUI) is shared with UART interrupt. This interrupt is level-low sensitive. To select this interrupt disable the ERI and ETI UART interrupt bits in the ENUI register. To select the UART interrupt leave this pin floating or tie it high.

Interrupt Control Registers

The HPC167064 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC167064 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. Figure 17 shows the Interrupt Enable Logic.

RESET

The RESET input initializes the processor and sets Ports A and B in the TRI-STATE condition and Port P in the LOW state. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between C000 and FFFF when emulating the HPC16064 and between E000 and FFFF when emulating the HPC16003.

Timer Overview

The HPC167064 contains a powerful set of flexible timers enabling the HPC167064 to perform extensive timer functions not usually associated with microcontrollers. The HPC167064 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16

4-150

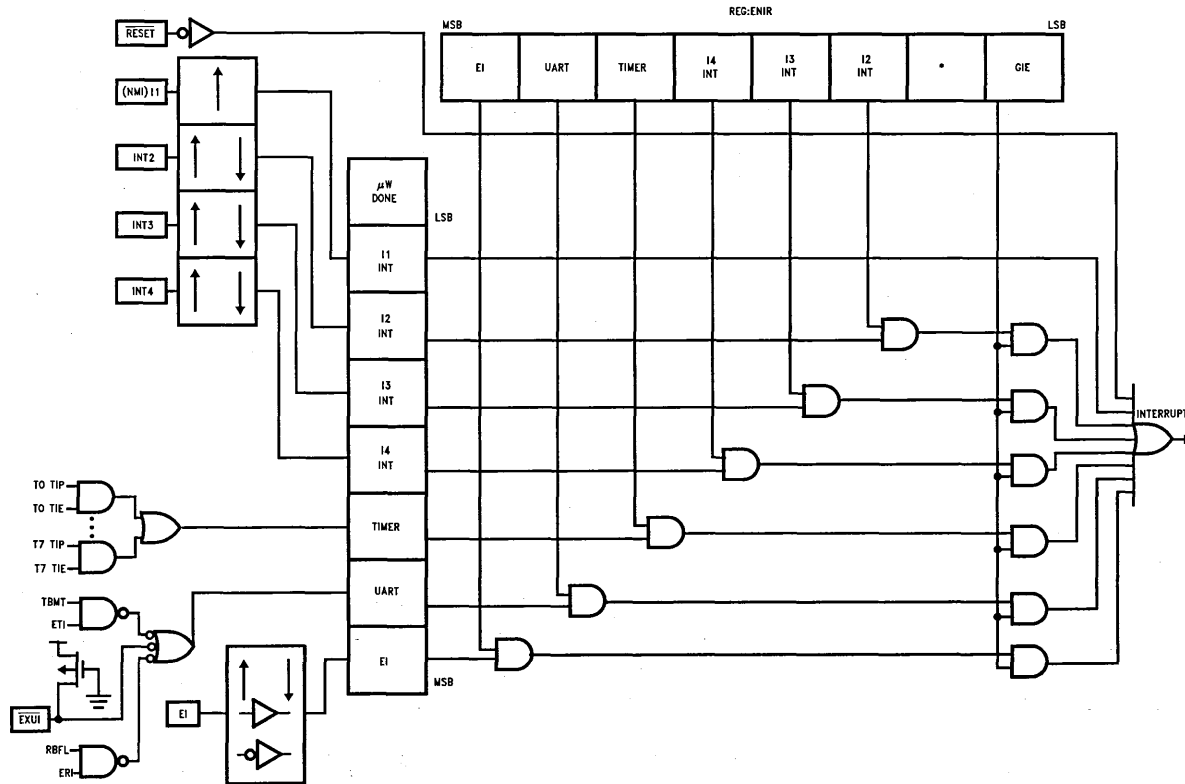


FIGURE 18. Block Diagram of Interrupt Logic

Timer Overview (Continued)

(Clock Input/16) rate. It is used for WATCHDOG logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see Figure 18).

The HPC167064 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by

dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see Figure 19).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC167064 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see Figure 19).

Timer/register pairs 4-7 form four identical units which can generate synchronous outputs on Port P (see Figure 20).

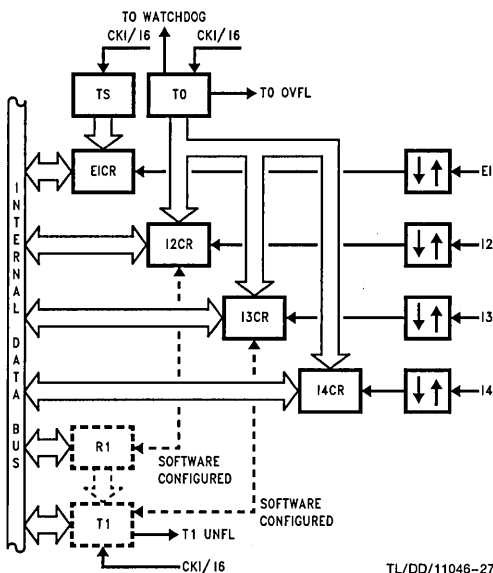


FIGURE 19. Timers T0, T1 and T8 with Four Input Capture Registers

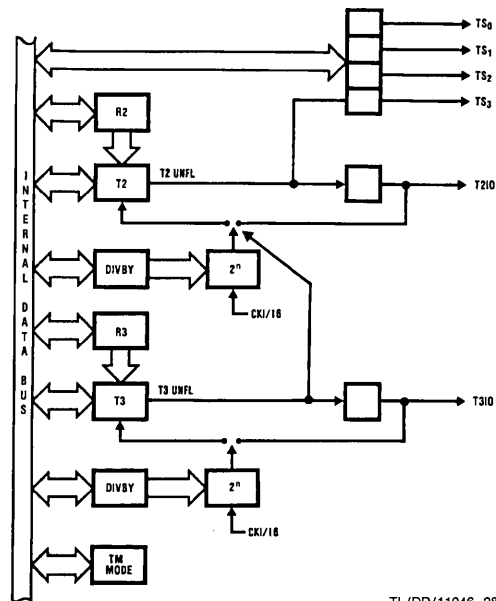


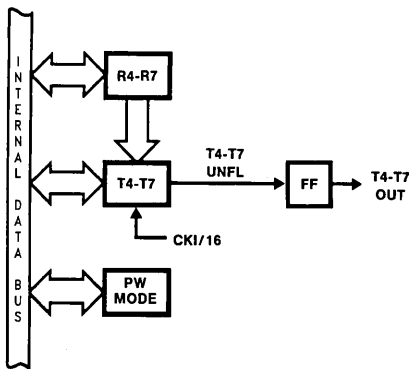
FIGURE 20. Timers T2-T3 Block

Timer Overview (Continued)

Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to $\frac{1}{2}$ the frequency of the source used for clocking the timer.

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.



TL/DD/11046-29
FIGURE 21. Timers T4-T7 Block

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC167064.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

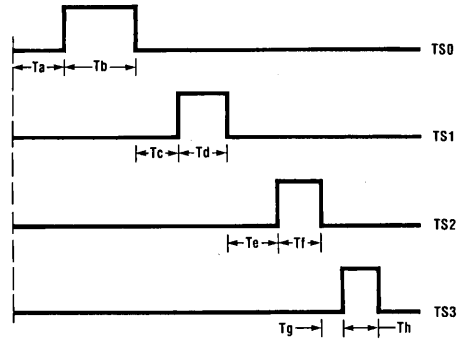
Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0-TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. Figure 22 is an example of synchronous pulse train generation.



TL/DD/11046-31
FIGURE 22. Square Wave Frequency Generation

WATCHDOG Logic

The WATCHDOG Logic monitors the operations taking place and signals upon the occurrence of any illegal activity.



TL/DD/11046-30
FIGURE 23. Synchronous Pulse Generation

The illegal conditions that trigger the WATCHDOG logic are potentially infinite loops and illegal addresses. Should the WATCHDOG register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the WATCHDOG Output (W \bar{O}) pin low. The W \bar{O} pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

*Note: See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see Figure 23). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS Operation

The HPC167064 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC167064 is the master or slave. The shift clock is generated when the HPC167064 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC167064 is configured as a slave. When the HPC167064 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 24 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC167064 microcontrollers interconnected to other MICROWIRE peripherals. HPC167064 1 is set up as the master and initiates all data transfers. HPC167064 2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of a LCD display controlled by the COP472 display driver. The data to be displayed is sent serially to the COP472 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC167064 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.

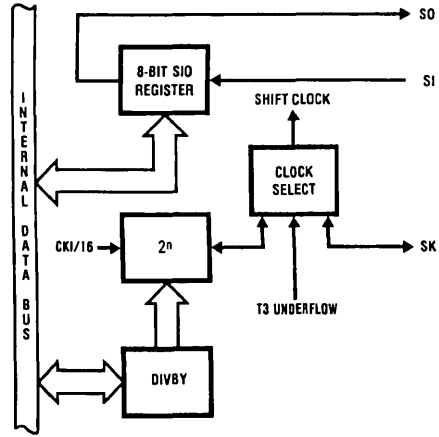


FIGURE 24. MICROWIRE/PLUS

TL/DD/11046-32

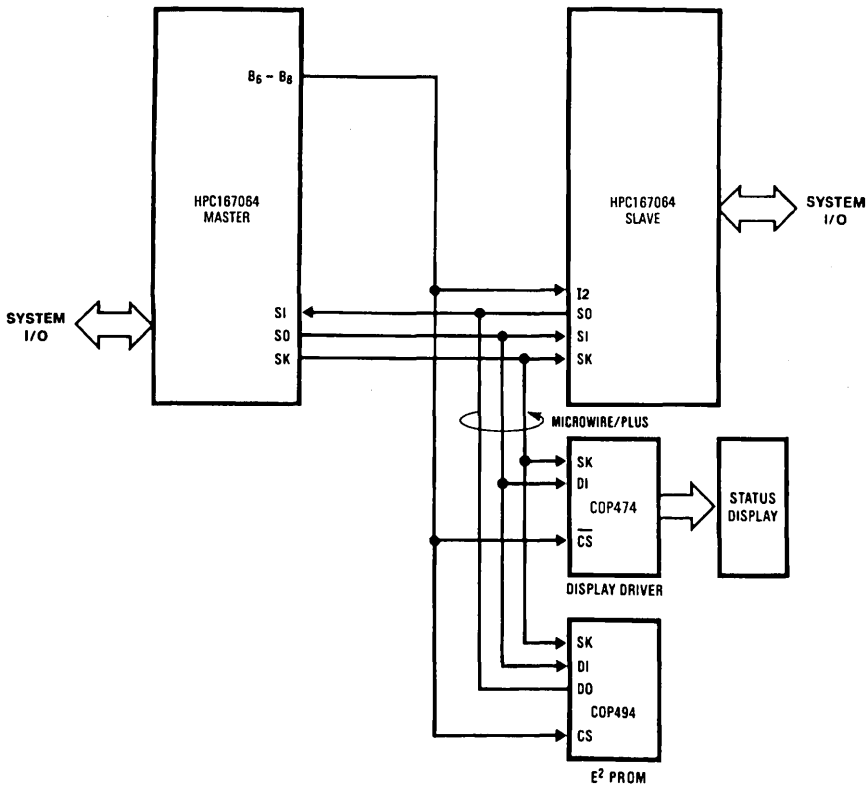


FIGURE 25. MICROWIRE/PLUS Application

TL/DD/11046-33

HPC167064 UART

The HPC167064 contains a software programmable UART. The UART (see *Figure 25*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Attention Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

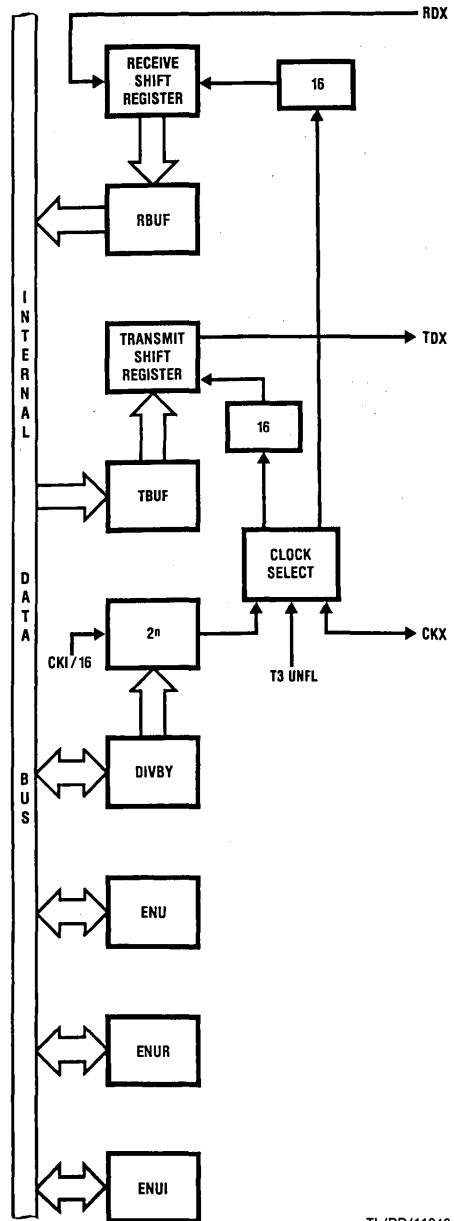
The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC167064 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-Up Mode

The HPC167064 UART features a Wake-Up Mode of operation. This mode of operation enables the HPC167064 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC167064 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.



TL/DD/11046-34

FIGURE 26. UART Block Diagram

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC167064 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC167064's and set up systems with very high data exchange rates. Another area of application could be where a HPC167064 is programmed as an intelligent peripheral to a host system such as the Series 32000[®] microprocessor. *Figure 26* illustrates how a HPC167064 could be used as an intelligent peripheral for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe (\overline{URD}), a Write Strobe (\overline{UWR}), a Read Ready Line (\overline{RDRDY}), a Write Ready Line (\overline{WRRDY}) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The \overline{URD} and \overline{UWR} inputs may be used to interrupt the HPC167064. The \overline{RDRDY} and \overline{WRRDY} outputs may be used to interrupt the host processor.

The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, Port A on the HPC167064 is the data bus. UPI can only be used if the HPC167064 is in the Single-Chip mode.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC167064 supports shared memory access with two pins. The pins are the $\overline{RDY}/\overline{HLD}$ input pin and the \overline{HLDA} output pin. The user can software select either the Hold or Ready function by the state of a control bit. The \overline{HLDA} output is multiplexed onto Port B.

The host uses DMA to interface with the HPC167064. The host initiates a data transfer by activating the \overline{HLD} input of the HPC167064. In response, the HPC167064 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal (\overline{HLDA}) from the HPC167064 indicating that the system bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC167064 resumes normal operations.

To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus. *Figure 27* illustrates an application of the shared memory interface between the HPC167064 and a Series 32000 system.

Memory

The HPC167064 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 8 kbytes of EPROM and 512 bytes of RAM available on the chip itself. The EPROM may contain program instructions, constants or data. The EPROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC167064 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC167064 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table III and Table IV.

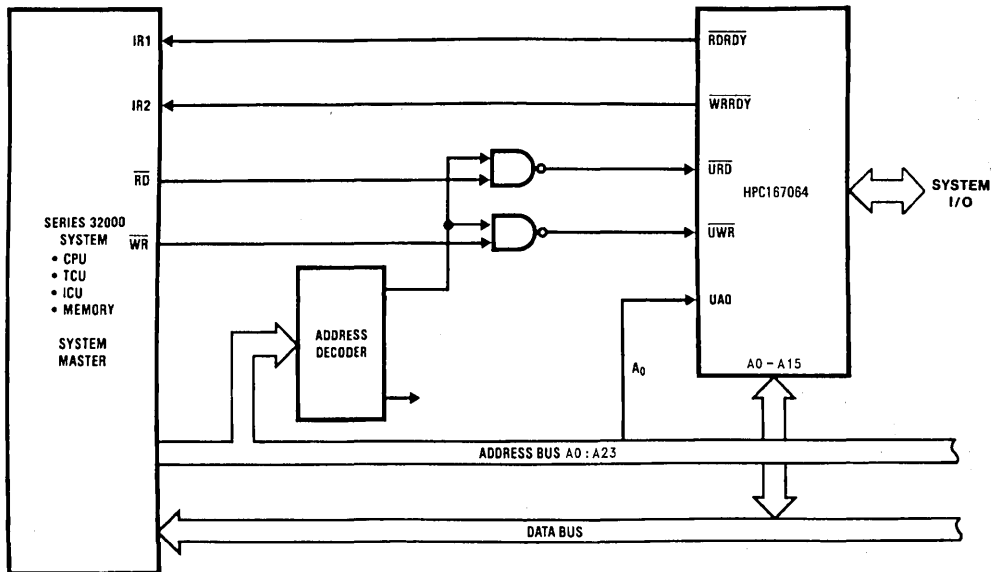


FIGURE 27. HPC167064 as a Peripheral (UPI Interface to Series 32000 Application)

TL/DD/11046-35

Shared Memory Support (Continued)

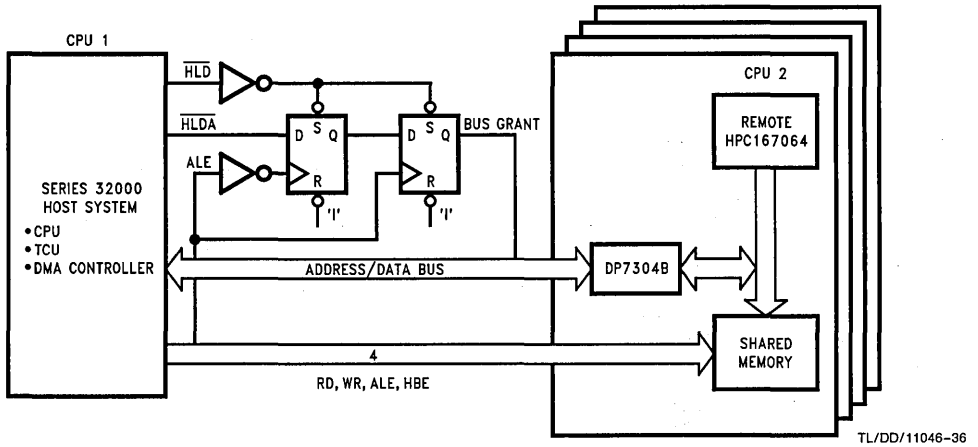


FIGURE 28. Shared Memory Application (HPC167064 Interface to Series 32000 System)

Design Considerations

TABLE III. Memory Map of HPC167064 Emulating an HPC16064

FFFF:FFF0	Interrupt Vectors	User Memory		
FFEF:FFD0	JSRP Vectors			
FFCF:FFCE	On-Chip ROM			
: :				
C001:C000				
BFFF:BFFE	External Expansion Memory			
: :				
0301:0300				
02FF:02FE	On-Chip RAM	User RAM		
: :				
01C1:01C0				
0195:0194	WATCHDOG Register	WATCHDOG Logic		
0192	T0CON Register	Timer Block T0:T3		
0191:0190	TMMODE Register			
018F:018	DIVBY Register			
018D:018C	T3 Timer			
018B:018A	R3 Register			
0189:0188	T2 Timer			
0187:0186	R2 Register			
0185:0184	I2CR Register/ R1			
0183:0182	I3CR Register/ T1			
0181:0180	I4CR Register			
015E:015F	EICR	Timer Block T4:T7		
015C	EICON			
0153:0152	Port P Register			
0151:0150	PWMODE Register			
014F:014E	R7 Register			
014D:014C	T7 Timer			
014B:014A	R6 Register			
0149:0148	T6 Timer			
0147:0146	R5 Register			
0145:0144	T5 Timer			
0143:0142	R4 Register			
0141:0140	T4 Timer			
0128	ENUR Register			UART
0126	TBUF Register			
0124	RBUF Register			
0122	ENUJ Register			
0120	ENU Register			
0104	Port D Input Register			
00F5:00F4	BFUN Register	Ports A & B Control		
00F3:00F2	DIR B Register			
00F1:00F0	DIR A Register/IBUF			
00E6	UPIC Register	UPI Control		
00E3:00E2	Port B	Ports A & B		
00E1:00E0	Port A/OBUF			
00DE	Reserved	Port Control & Interrupt Control Registers		
00DD:00DC	HALT Enable Register			
00D8	Port I Input Register			
00D6	SIO Register			
00D4	IRCD Register			
00D2	IRPD Register			
00D0	ENIR Register			
00CF:00CE	X Register	HPC Core Registers		
00CD:00CC	B Register			
00CB:00CA	K Register			
00C9:00C8	A Register			
00C7:00C6	PC Register			
00C5:00C4	SP Register			
00C3:00C2	Reserved			
00C0	PSW Register			
00BF:00BE	On-Chip RAM	User RAM		
: :				
0001:0000				

Design Considerations (Continued)

TABLE IV. Memory Map of HPC167064 Emulating an HPC16083

FFFF:FFF0 FFEF:FFD0 FFCF:FFCE : : E001:E000	Interrupt Vectors JSRP Vectors } On-Chip EPROM	User Memory
DFFF:DFFE : : 0201:0200	} External Expansion Memory	
01FF:01FE : : 01C1:01C0	} On-Chip RAM	User RAM
0195:0194	WATCHDOG Register	WATCHDOG Logic
0192 0191:0190 018F:018E 018D:018C 018B:018A 0189:0188 0187:0186 0185:0184 0183:0182 0181:0180	T0CON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/R1 I3CR Register/T1 I4CR Register	Timer Block T0:T3
015E:015F 015C 0153:0152 0151:0150 014F:014E 014D:014C 014B:014A 0149:0148 0147:0146 0145:0144 0143:0142 0141:0140	EICR EICON Port P Register PWMODE Register R7 Register T7 Timer R6 Register T6 Timer R5 Register T5 Timer R4 Register T4 Timer	Timer Block T4:T7

0128 0126 0124 0122 0120	ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART
0104	Port D Input Register	
00F5:00F4 00F3:00F2 00F1:00F0	BFUN Register DIR B Register DIR A Register /IBUF	Ports A & B Control
00E6	UPIC Register	UPI Control
00E3:00E2 00E1:00E0	Port B Port A/OBUF	Ports A & B
00DE 00DD:00DC 00D8 00D6 00D4 00D2 00D0	Reserved HALT Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	Port Control & Interrupt Control Registers
00CF:00CE 00CD:00CC 00CB:00CA 00C9:00C8 00C7:00C6 00C5:00C4 00C3:00C2 00C0	X Register B Register K Register A Register PC Register SP Register Reserved PSW Register	HPC Core Registers
00BF:00BE : : 0001:0000	On-Chip RAM	User RAM

Design Considerations (Continued)

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused output should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least 1 μF and bypass their outputs with a 10 μF to 50 μF tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a 10 μF to 20F tantalum electrolytic capacitor or a 50 μF to 100 μF aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.

A recommended crystal oscillator circuit to be used with the HPC is shown in Figure 29. See table for recommended component values. The recommended values given in Table V have yielded consistent results and are made to match a crystal with a 20 pF load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within 1" distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A 1.0 μF , a 0.1F, and a 0.001F dipped mica or ceramic cap should be mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

TABLE V. HPC Oscillator

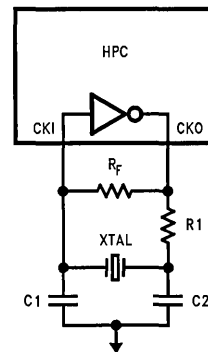
XTAL Frequency (MHz)	R_1 (Ω)
2	1500
4	1200
6	910
8	750
10	600
12	470
14	390
16	300
18	220
20	180

$R_F = 3.3 \text{ M}\Omega$
 $C_1 = 27 \text{ pF}$
 $C_2 = 33 \text{ pF}$

XTAL Specifications: The crystal used was an M-TRON Industries MP-1 Series XTAL. "AT" cut, parallel resonant.

$C_L = 20 \text{ pF}$

Series Resistance is
 25 Ω @ 25 MHz
 40 Ω @ 10 MHz
 600 Ω @ 2 MHz



TL/DD/11046-37

FIGURE 29. Recommended Crystal Circuit

HPC167064 CPU

The HPC167064 CPU has a 16-bit ALU and six 16-bit registers.

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC167064 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	MA + Mem1 → MA carry → C
ADC	Add with carry	MA + Mem1 + CMA carry → C
ADDS	Add short imm8	A + imm8 → A carry → C
DADC	Decimal add with carry	MA + Mem1 + C → MA (Decimal) carry → C
SUBC	Subtract with carry	MA - Mem1 + C → MA carry → C
DSUBC	Decimal subtract w/carry	MA - Mem1 + C → MA (Decimal) carry → C
MULT	Multiply (unsigned)	MA * Mem1 → MA & X, 0 → K, 0 → C
DIV	Divide (unsigned)	MA/Mem1 → MA, rem → X, 0 → K, 0 → C
DIVD	Divide Double Word (unsigned)	X & MA/Mem1 → MA, rem → X, 0 → K, carry → C
IFEQ	If equal	Compare MA & Mem1, Do next if equal
IFGT	If greater than	Compare MA & Mem1, Do next if MA > Mem1
AND	Logical AND	MA and Mem1 → MA
OR	Logical OR	MA or Mem1 → MA
XOR	Logical Exclusive-OR	MA xor Mem1 → MA
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	Mem + 1 → Mem
DECSZ	Decrement, skip if 0	Mem - 1 → Mem, Skip next if Mem = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Mem1 → MA
ST	Load, incr/decr X	Mem(X) → A, X ± 1 (or 2) → X
X	Store to Memory	A → Mem
	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ± 1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP
POP	Pop Stack to Memory	SP-2 → SP, W(SP) → W
LDS	Load A, incr/decr B,	Mem(B) → A, B ± 1 (or 2) → B,
	Skip on condition	Skip next if B greater/less than K
XS	Exchange, incr/decr B,	Mem(B) ↔ A, B ± 1 (or 2) → B,
	Skip on condition	Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B, imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INCA	Increment A	A + 1 → A
DECA	Decrement A	A - 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A[15:12] ← A[11:8] ← A[7:4] ↔ A[3:0]
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP), SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP), SP+2 → SP, PC+ # → PC (# is +1025 to -1023)
JSRL	Jump subroutine long	PC → W(SP), SP+2 → SP, PC+ # → PC
JP	Jump relative short	PC+ # → PC (# is +32 to -31)
JMP	Jump relative	PC+ # → PC (# is +257 to -255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+ A + 1 → PC then Mem(PC) + PC → PC
JIDW		
NOP	No Operation	PC + 1 → PC
RET	Return	SP-2 → SP, W(SP) → PC
RETSK	Return then skip next	SP-2 → SP, W(SP) → PC, & skip
RETI	Return from interrupt	SP-2 → SP, W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory

MA is Accumulator A or direct memory (8-bit or 16-bit)

Mem is 8-bit byte or 16-bit word of memory

Mem1 is 8-bit or 16-bit memory or 8-bit or 16-bit immediate data

imm is 8-bit or 16-bit immediate data

imm8 is 8-bit immediate data only

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC family has been designed to be extremely code-efficient. The HPC looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC167064 are single-byte. There are two especially code-saving instructions: JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte subroutine call. The user makes a table of the 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into the table; the assembler can give this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC167064 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC167064 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC167064 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

HPC MICROCONTROLLER DEVELOPMENT SYSTEM

The HPC microcontroller development system is an in-system emulator (ISE) designed to support the entire family of HPC Microcontrollers. The complete package of hardware and software tools combined with a host system provides a powerful system for design, development and debug of HPC based designs. Software tools are available for IBM® PC-AT® (MS-DOS, PC-DOS) and for UNIX® based multi-user Sun® SPARCstation (SunOSTM).

The stand alone units comes complete with a power supply and external emulation POD. This unit can be connected to various host systems through an RS-232 link. The software package includes an ANSI compatible C-Compiler, Linker, Assembler and librarian package. Source symbolic debug capability is provided through a user friendly MS-windows 3.0 interface.

The ISE provides fully transparent in-system emulation at speeds up to 20 MHz 1 waitstate. A 2k word (48-bit wide) trace buffer gives trace trigger and non intrusive monitoring of the system. External triggering is also available through an external logic interface socket on the POD. Direct EPROM programming can be done through the use of externally mounted EPROM socket. Form-Fit-Function emulator programming is supported by a programming board included with the system. Comprehensive on-line help and diagnostics features reduce user's design and debug time. 8 hardware breakpoints (Address/range), 64k bytes of user memory, and break on external events are some of the other features offered.

Hewlett Packard model HP64775 Emulator/Analyzer providing in-system emulation for up to 30 MHz 1 waitstate is also available. Contact your local sales office for technical details and support.

Development Support (Continued)

PROGRAMMING SUPPORT

The HPC167064 EPROM array can be programmed using the HPC ISE with the appropriate programming adaptor board, 7064-PRGM-LDCC. The procedure for doing this is documented in the HPC Emulator Programmer User's Manual that is shipped with every ISE. The programming adaptor board must be ordered separately. The EPROM array can also be programmed using a DATA I/O Unisite model with a pinsite module. No adaptor board is required with the DATA

I/O programmer. Programming of the configuration bytes and security bits is described in the HPC Family User's Manual.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Product	Order Number	Description	Includes	Manual Number
HPC16003/16083	HPC-DEV-ISE1 HPC-DEV-ISE1-E	HPC In-System Emulator HPC In-System Emulator for Europe and South East Asia	HPC MDS User's Manual MDS Comm User's Manual HPC Emulator Programmer User's Manual HPC16083/16004/16064 Manual	420420184-001 424420188-001 420421313-001 424410897-001
	HPC-DEV-IBMA	Assembler/Linker/ Library Package for IBM PC/AT	HPC Assembler/Linker Librarian User's Manual	424410836-001
	HPC-DEV-IBMC	C Compiler/Assembler/ Linker/Library Package for IBM PC/AT	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424410883-0 424410836-001
	HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC/AT C Compiler/Assembler/ Linker/Library Package for IBM PC/AT	Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	424420189-001 424410883-001 424410836-001
	HPC-DEV-SUNC HPC-DEV-SUNDB	C Compiler/Assembler/ Linker Library Package for SUN SPARCstation Source/Symbolic Debugger for Sun SPARCstation C Compiler/Assembler/ Linker Library Package	HPC C Compiler User's Manual HCP Assembler/Linker/Library User's Manual Source/Symbolic Debugger User's Manual HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual	

COMPLETE SYSTEM

HPC16003/16083	HPC-DEV-SYS1 HPC-DEV-SYS1-E	HPC In-System Emulator with C Compiler/Assembler/Linker/Library and Source Symbolic Debugger Same for Europe and South East Asia		
----------------	------------------------------------	--	--	--

VAX™ UNIX will be supported in the near future. Contact field sales for more information.

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. Dial-A-Helper is an Electronic Bulletin Board Information system and, additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be downloaded to disk for later use.

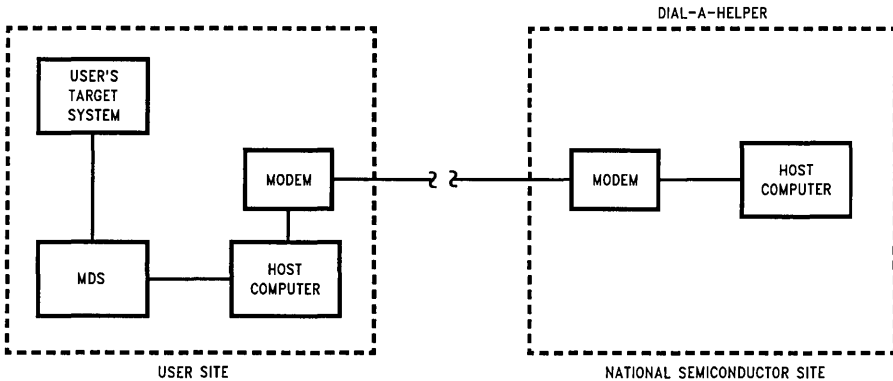
Order P/N: MDS-DIAL-A-HLP

Information System Package Contains:
 Dial-A-Helper Users Manual
 Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MDS, messages can be left on our electronic bulletin board, which we will respond to.

Voice: (408) 721-5582
 Modem: (408) 739-1162
 Baud: 300 or 1200 baud
 Set-Up: Length: 8-bit
 Parity: None
 Stop Bit: 1
 Operation: 24 hrs, 7 Days

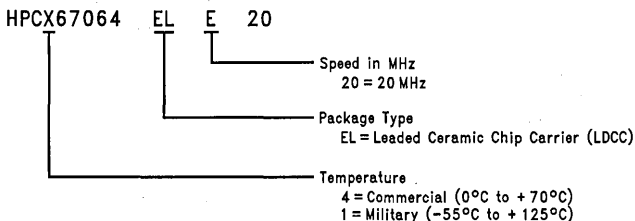


TL/DD/11046-38

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC167064 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/11046-39

Examples:

HPC467064/EL20—16k EPROM, Commercial temperature (0°C to +70°C), LDCC

HPC167064/EL20—16k EPROM Military temperature (-55°C to +125°C), LDCC (to be used for automotive temperature range also)

Socket Selection

Suggested sockets and extractor tool:

Socket #	Amp	PLCC	#821574-1 6141749
	*YAMAICHI	1C51-0684-390 1C120-0684-204	
Extractors Tool #	ENPLAS Amp	PLCC-68-1.27-02 821566-1	

*A shim must be used in conjunction with this socket to ensure proper contacts. For details of the shim and how to obtain it, contact factory applications group at (408) 721-5582.

HPC46100 High-Performance microController with DSP Capability

General Description

The HPC46100 is a member of the HPC™ family of High Performance microControllers. Each member of the family has a similar core CPU with unique memory, resources, and I/O configuration to suit specific applications. The HPC46100 is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, high speed computation, and low power consumption.

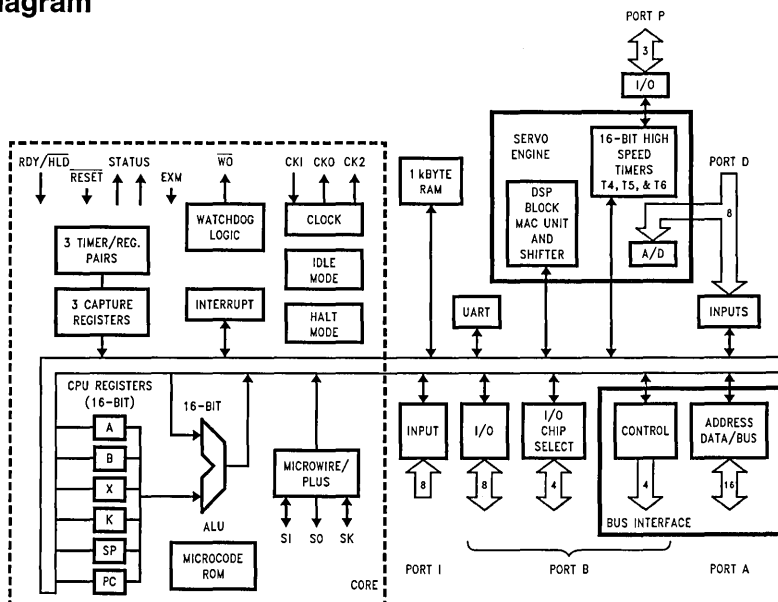
Throughput is enhanced by operating the HPC46100 at frequencies up to 40 MHz, by integrating a Multiply/Accumulate Unit (MAU) onto the chip, and by optimizing instructions to increase efficiency. These features increase performance in closed loop digital servo and filter applications.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as an MAU unit, PWM outputs, Chip Select Signals, UART, up to seven 16-bit timers with input capture capability, WATCHDOG™ logic, vectored interrupts, and MICROWIRE/PLUS™ provide a high level of system integration. The ability to directly address up to 64 kbytes of memory enables the HPC to be used in powerful applications typically performed by microprocessors and peripheral chips.

Features

- Multiply/Accumulate Unit for fast signed multiply or multiply-accumulate
- High speed 16 bit timers with PWM outputs or input capture logic
- 4 Chip select output logic with programmable control
- 8-channel, 8-bit A/D Converter
- 1024 bytes of on-chip 0 wait state RAM
- FAST 100 ns for fastest instruction when using 40.0 MHz clock
- Very low power with two power save modes: IDLE and HALT
- UART full duplex, with a programmable baud rate generator and parity checking/detection
- MICROWIRE/PLUS serial I/O interface
- 8 vectored interrupt sources
- Signed overflow flag for add and subtract instructions
- 16 x 16 multiply and 32 x 16 divide
- 16-bit architecture with byte and word operations
- 64 kbytes of direct memory addressing
- 8- or 16-bit wide external memory
- Program instructions can be executed from RAM
- Up to 32 general purpose I/O lines that are memory mapped
- WATCHDOG logic

Block Diagram



TL/DD/11289-1

40 MHz**Absolute Maximum Ratings**

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
V _{CC} with Respect to GND	-0.5V to +6.5V
V _{REF} with Respect to GND	V _{CC}
All Other Pins	(V _{CC} + 0.5V) to (GND - 0.5V)

DC Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified. T_A = 0°C to +70°C unless otherwise specified.

Symbol	Parameter	Test Conditions	Min	Max	Units
SUPPLIES					
I _{CC1}	Supply Current	V _{CC} = 5.5V, f _{IN} = 40 MHz (Note 1) V _{CC} = 5.5V, f _{IN} = 20 MHz (Note 1)		65 40	mA mA
I _{CC2}	IDLE Mode Supply Current	V _{CC} = 5.5V, f _{IN} = 40 MHz (Note 1) V _{CC} = 5.5V, f _{IN} = 20 MHz (Note 1)		25 20	mA mA
I _{CC3}	HALT Mode Supply Current with $\overline{\text{NMI}}$ High	V _{CC} = 5.5V, f _{IN} = 0 (Note 1) V _{CC} = 2.5V, f _{IN} = 0 (Note 1)		300 100	μA μA
I _{CC4}	HALT Mode Supply Current with $\overline{\text{NMI}}$ Low	V _{CC} = 5.5V, f _{IN} = 40 MHz (Note 1)		25	mA
V _{RAM}	RAM Keep-Alive Voltage	(Note 2)	2.5		V
INPUT VOLTAGE LEVELS FOR SCHMITT TRIGGERED INPUTS RESET, NMI AND W0; AND ALSO CKI					
V _{IH1}	Logic High		0.9 V _{CC}		V
V _{IL1}	Logic Low			0.1 V _{CC}	V
INPUT VOLTAGE LEVELS FOR PORT A					
V _{IH2}	Logic High		2.0		V
V _{IL2}	Logic Low			0.8	V
INPUT VOLTAGE LEVELS FOR ALL OTHER INPUTS					
V _{IH3}	Logic High (except Port D)		0.7 V _{CC}		V
V _{IL3}	Logic Low (except Port D)			0.2 V _{CC}	V
V _{IH4}	Logic High (Port D only)	(Note 6 in AC Characteristics)	0.7 V _{CC}	V _{CC}	V
V _{IL4}	Logic Low (Port D only)	(Note 6 in AC Characteristics)	GND	0.2 V _{CC}	V
I _{LI1}	Input Leakage Current; all pins except below			10	μA
I _{LI2}	Input Leakage Current; pin RDY/HLD only		-3	-50	μA
I _{LI3}	Input Leakage Current; pin B12 (HBE) only	RESET = GND, V _{IN} = V _{CC}	0.5	7.0	mA
C _I	Input Pin Capacitance	(Note 3)		10	pF
C _{IO}	Input/Output Pin Capacitance	(Note 3)		20	pF
OUTPUT VOLTAGE LEVELS					
CMOS USAGE: ALL OUTPUTS AND I/O PINS					
V _{OH1}	Logic High	I _{OH} = -10 μA (Note 3)	V _{CC} - 0.1		V
V _{OL1}	Logic Low	I _{OL} = 10 μA (Note 3)		0.1	V

40 MHz (Continued)

DC Characteristics

$V_{CC} = 5.0V \pm 10\%$ unless otherwise specified. $T_A = 0^\circ C$ to $+70^\circ C$ unless otherwise specified. (Continued)

Symbol	Parameter	Test Conditions	Min	Max	Units
BUS DRIVERS: PORT A AND PINS B8–B15, PALE, CK2, ST1 AND ST2					
V_{OH2}	Logic High	$I_{OH} = -1\text{ mA}$	2.4		V
V_{OL2}	Logic Low	$I_{OL} = 3\text{ mA}$		0.4	V
OTHER I/O PORT DRIVERS: B0–B7, W0, P0–P2					
V_{OH3}	Logic High	$I_{OH} = -1.6\text{ mA}$	2.4		V
V_{OL3}	Logic Low	$I_{OL} = 0.5\text{ mA}$		0.4	V
I_{OZ}	TRI-STATE® Leakage Current			10	μA

Note 1: I_{CC1} , I_{CC2} , I_{CC3} and I_{CC4} are measured with no external drive (I_{OH} and $I_{OL} = 0$; I_{IH} and $I_{IL} = 0$). I_{CC1} is measured with $\overline{RESET} = GND$, I_{CC3} is measured with $NMI = V_{CC}$, I_{CC4} is measured with $NMI = GND$, CK1 driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: Test duration is 100 ms.

Note 3: This is guaranteed by design and not tested.

AC Electrical Characteristics

See Notes 1 and 4 and Figure 1 thru Figure 5. $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 25^\circ C$, one wait state.

Symbol	Parameter	Min	Max	Units
CLOCKS				
f_C	CKI Operating Frequency	2	40	MHz
$t_{C1} = 1/f_C$	CKI Period	25	500	ns
t_{C1H}	CKI High Time	11.25		ns
t_{C1L}	CKI Low Time	11.25		ns
$t_C = 2/f_C$	Bus Timing Cycle	50	1000	ns
t_{WAIT}	Wait State Period	50		ns
t_{DC1C2R}	CK2 Rising Edge after CK1 Falling Edge	0	55	ns
t_{DC1C2F}	CK2 Falling Edge after CK1 Falling Edge	0	55	ns
f_U	External UART Clock Input Frequency		5	MHz
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		2.5	MHz
TIMER T0–T3				
$f_{XIN} = f_C/22$	External Timer Input Frequency		2.105	MHz
$t_{XIN} = t_C$	Pulse Width for Timer Input	50		ns
TIMER T4–T6				
$f_{HSXIN} = f_C/5$	External Timer Input Frequency		5	MHz
$t_{HSXIN} = 1.5 t_C$	Pulse Width for Timer Input	75		ns
MICROWIRE/PLUS				
t_{UWS} Master Slave	MICROWIRE Setup Time	100 20		ns
t_{UWH} Master Slave	MICROWIRE Hold Time	20 50		ns
t_{UWS} Master Slave	MICROWIRE Output Valid Time		50 150	ns

40 MHz (Continued)

AC Electrical Characteristics (Continued)

See Notes 1 and 4 and Figure 1 thru Figure 5). $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 25^\circ C$, one wait state.

Symbol	Parameter	Min	Max	Units
EXTERNAL HOLD				
$t_{SALE} = 3/4 t_C + 40$	\overline{HLD} Falling Edge before ALE Rising Edge	115		ns
$t_{HWP} = 3/4 t_C + 85$	\overline{HLD} Pulse Width	115		ns
$t_{HAE} = 3/4 t_C + 100$	HLDA Falling Edge after \overline{HLD} Falling Edge (Note 3)		175	ns
$t_{HAD} = 3/4 t_C + 85$	HLDA Rising Edge after \overline{HLD} Rising Edge		210	ns
t_{BF}	Bus TRI-Stated after HLDA Falling Edge (Note 5)		66	ns
$t_{BE} = t_C - 66$	Bus Enable after HLDA Rising Edge	34		ns
NATIVE BUS TIMING: ADDRESS CYCLE				
$t_{LL} = 0.5 t_C - 10$	ALE Pulse Width	15		ns
t_{1ALR}	ALE Rising Edge after CK1 Rising Edge (Note 2)	0	35	ns
t_{1ALF}	ALE Falling Edge after CK1 Falling Edge (Note 2)	0	35	ns
$t_{2ALR} = 1/4 t_C + 20$	ALE Rising Edge after CK2 Rising Edge		40	ns
$t_{2ALF} = 1/4 t_C + 20$	ALE Falling Edge after CK2 Falling Edge		40	ns
$t_{ST} = 0.25 t_C - 9$	Address Valid to ALE Falling Edge	3.5		ns
$t_{VP} = 0.5 t_C - 10$	Address Hold after ALE Falling Edge	15		ns
READ CYCLE				
$t_{RW} = 0.25 t_C + WS - 15$	\overline{RD} Pulse Width	47.5		ns
$t_{ARD} = 0.75 t_C - 20$	Address Valid to \overline{RD} Falling Edge	17.5		ns
$t_{ARR} = 1/2 t_C - 20$	ALE Falling Edge to \overline{RD} Falling Edge	30		ns
$t_{RD} = 0.25 t_C + WS - 20$	\overline{RD} Falling Edge to Input Data Valid		42.5	ns
t_{DR}	Data Hold after \overline{RD} Rising Edge	0	50	ns
$t_{ACC} = t_C + WS - 20$	Address Valid to Input Data Valid		80	ns
WRITE CYCLE				
$t_{WW} = 0.75 t_C + WS - 15$	\overline{WR} Pulse Width	72.5		ns
$t_V = 0.5 t_C + WS - 20$	Data Valid before \overline{WR} Rising Edge	55		ns
$t_{HW} = 0.5 t_C - 10$	Data Hold after \overline{WR} Rising Edge	15		ns
$t_{AWR} = 0.75 t_C - 20$	Address Valid to \overline{WR} Falling Edge	17.5		ns
READY INPUT				
t_{RDYS}	\overline{RDY} Falling Edge before CK2 Falling Edge	45		ns
t_{RDYH}	\overline{RDY} Rising Edge after CK2 Falling Edge	0		ns
$t_{RDYV} = WS - 1/4 t_C - 47$	\overline{RDY} Falling Edge after \overline{RD} or \overline{WR} Falling Edge		28	ns
CHIP SELECT NATIVE BUS TIMING				
$t_{CS30RD} = 0.75 t_C - 30$	CS3, CS0 Valid to \overline{RD} Falling Edge	7.5		ns
$t_{ACCS30} = t_C + WS - 30$	CS3, CS0 Valid to Input Data Valid		70	ns
$t_{CS21RD} = 0.75 t_C - 35$	CS2, CS1 Valid to \overline{RD} Falling Edge	2.5		ns
$t_{ACCS21} = t_C + WS - 35$	CS2, CS1 Valid to Input Data Valid		65	ns
$t_{CSHR} = t_C - 15$	Chip Select Hold after \overline{RD} Rising Edge	35		ns
$t_{CS30WR} = 0.75 t_C - 30$	CS3, CS0 Valid to \overline{WR} Falling Edge	7.5		ns
$t_{CS21WR} = 0.75 t_C - 35$	CS2, CS1 Valid to \overline{WR} Falling Edge	2.5		ns
$t_{CSHW} = 0.5 t_C - 15$	Chip Select Hold after \overline{WR} Rising Edge	10		ns

40 MHz (Continued)

AC Electrical Characteristics (Continued)

See Notes 1 and 4 and Figure 1 thru Figure 5. $V_{CC} = 5.0V \pm 10\%$ unless otherwise specified, $T_A = 25^\circ C$, one wait state.

Symbol	Parameter	Min	Max	Units
E SIGNAL TIMING PARAMETERS				
$t_{RWSE} = 0.25 t_C - 7$	\overline{WR} Falling Edge to E Rising Edge	5.5		ns
$t_{RWHE} = 0.5 t_C - 7$	E Falling Edge to \overline{WR} Rising Edge	18		ns
$t_{ASE} = t_C - 20$	Address Valid to E Rising Edge	30		ns
$t_{RDE} = WS - 20$	E Falling Edge to Data Input Valid	30		ns
SLOW PERIPHERAL TIMING PARAMETERS				
$t_{PLL} = t_C - 5$	PALE Pulse Width	45		ns
$t_{PST} = 0.75 t_C - 10$	Address Valid to PALE Falling Edge	27.5		ns
$t_{PVL} = 0.75 t_C - 15$	Address Hold from PALE Falling Edge	22.5		ns
$t_{PVP} = 0.75 t_C - 10$	PALE Falling Edge to \overline{RD} or \overline{WR} Falling Edge	27.5		ns
$t_{PCSA} = 0.25 t_C - 12.5$	Chip Select Setup to PALE Falling Edge	0		ns
$t_{PAS} = 1.5 t_C - 20$	Address Setup to \overline{RD} or \overline{WR} Falling Edge	55		ns
$t_{PCSS} = t_C - 15$	Chip Select Setup to \overline{RD} or \overline{WR} Falling Edge	35		ns
$t_{PCSH} = 0.5 t_C - 15$	Chip Select Hold from \overline{RD} or \overline{WR} Rising Edge	10		ns
$t_{PACC} = 5 t_C - 25$	Address Valid to Input Data Valid		225	ns
$t_{PRD} = 3.5 t_C - 25$	\overline{RD} Falling Edge to Data In Valid		150	ns
$t_{PDR} = t_C$ (max)	Data Hold after \overline{RD} Rising Edge	0	50	ns
$t_{PRW} = 3.5 t_C - 15$	\overline{RD} Strobe Width	160		ns
$t_{PSW} = 3.0 t_C - 20$	Data Setup before \overline{WR} Rising Edge	130		ns
$t_{PHW} = t_C - 20$	Data Hold after \overline{WR} Rising Edge	30		ns
$t_{PWW} = 3.5 t_C - 15$	\overline{WR} Strobe Width	160		ns

Note: $C_L = 40$ pF.

Note 1: These AC characteristics are guaranteed with external clock drive on CKI having 50% duty cycle and with less than 15 pF load on CKO with rise and fall times (t_{CKIR} and t_{CKIL}) on CKI input less than 2.5 ns.

Note 2: Do not design with these parameters unless CKI is driven with an active signal. When using a passive crystal circuit, its stability is not guaranteed if either CKI or CKO is connected to any external logic other than the passive components of the crystal circuit.

Note 3: t_{HAE} is spec'd for case with \overline{RD} falling edge occurring at the latest time it can be accepted during the present CPU cycle begin executed. If \overline{RD} falling edge occurs later, t_{HAE} as long as $(3 t_C + 4 WS + 72 t_C + 100)$ may occur depending on the following CPU instruction cycles, its wait states and ready input.

Note 4: WS (t_{WAIT}) x (number of preprogrammed wait states). Minimum and maximum values are calculated at maximum operating frequency, $t_C = 30$ MHz, with one wait state programmed.

Note 5: Due to emulation restrictions—actual limits will be better.

A/D Converter Specifications

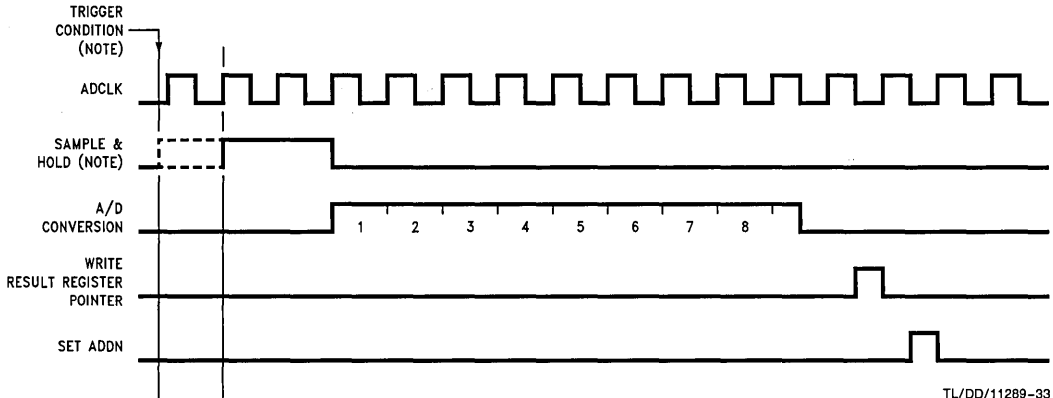
$V_{CC} = 5V \pm 10\%$ ($V_{SS} - 0.050V$) \leq Any Input $\leq (V_{CC} + 0.050V)$, $f_{IN} = 24$ MHz

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			± 1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the Best Straight Line			$\pm 1/2$	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			$\pm 1/2$	LSB
Input Reference Resistance		1.6		4.8	k Ω
Common Mode Input Range (Note 6)		AGND		V_{REF}	V
DC Common Mode Error				$\pm 1/4$	LSB
Off Channel Leakage Current			1		μ A
On Channel Leakage Current			1		μ A
A/D Clock Frequency (Note 7)		0.1		2.0	MHz
Conversion Time (Note 8)			13.0		A/D Clock Cycle

Note 6: For $V_{IN(-)} \geq V_{IN(+)}$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 7: See Prescaler description.

Note 8: Conversion Time includes sample and hold time. See following diagrams.



TL/DD/11289-33

Note: The trigger condition generated by the start conversion method selected by the SC bits requires one CK2 to propagate through before the trigger condition is known. Once the trigger condition is known, the sample and hold will start at the next rising edge of ADCLK. The diagram shows worst case.

General Description (Continued)

The HPC46100 has, as an on-chip peripheral, an 8-channel 8-bit Analog-to-Digital Converter. This A/D converter can operate in a single-ended mode where the analog input voltage is applied across one of the eight input channels (D0-D7) and AGND. The A/D converter can also operate in a differential mode where the analog input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in the single-ended mode and up to four channel pairs in the differential mode.

A group of three high speed timers provide processor independent PWM signal generation. These timers and their support logic provide independent control of PWM frequency and PWM duty cycle with a minimum resolution of 50 ns when running at 40 MHz.

The HPC46100 is upwards source code compatible with the HPC family except for Decimal Add and Subtract.

The HPC46100 is available in an 80-pin QFP package.

Timing Waveforms

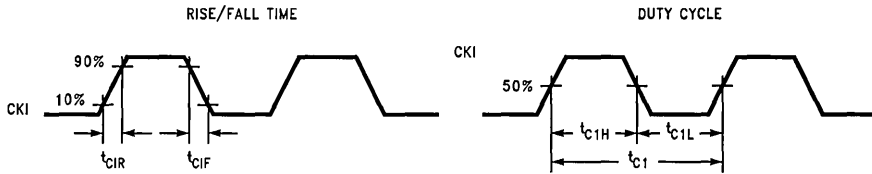
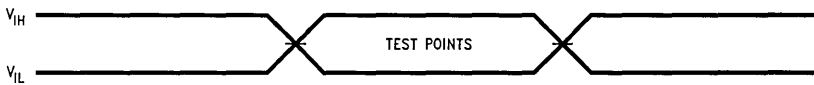


FIGURE 1. CKI Input Signal

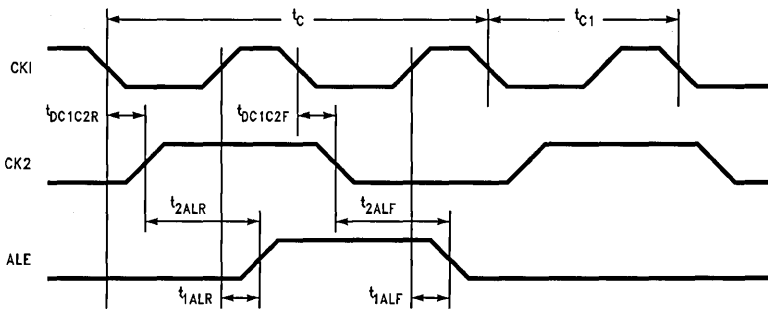
TL/DD/11289-2



TL/DD/11289-3

Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at $V_{CC}/2$ for both logic "1" and logic "0".

FIGURE 2. Input and Output for AC Tests



TL/DD/11289-4

FIGURE 3. CKI, CK2 ALE Timing Diagram

Timing Waveforms (Continued)

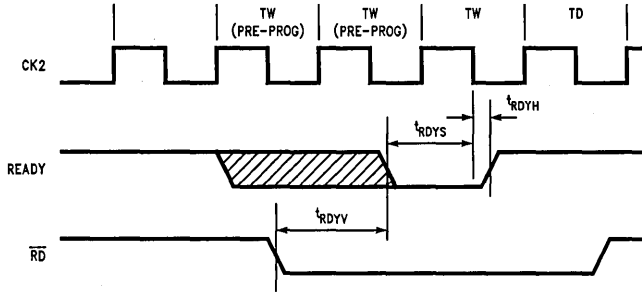


FIGURE 4. Ready Mode Timing

TL/DD/11289-5

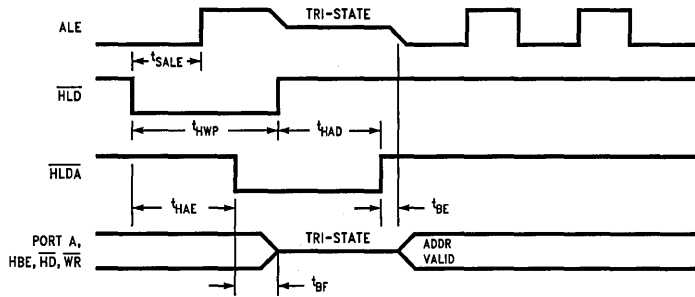


FIGURE 5. External Hold Timing

TL/DD/11289-6

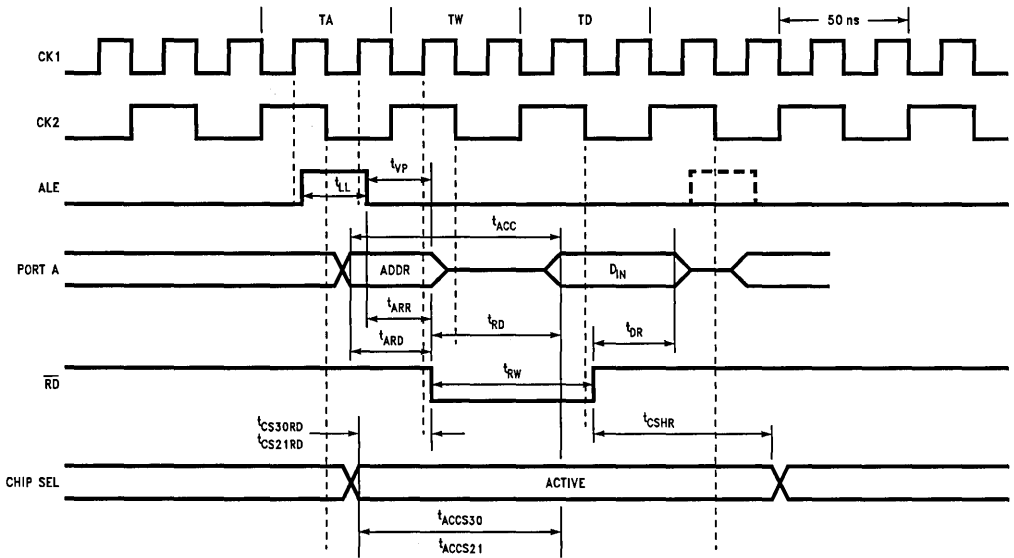


FIGURE 6. Native Bus Mode Read Cycle (1 Wait State)

TL/DD/11289-7

Timing Waveforms (Continued)

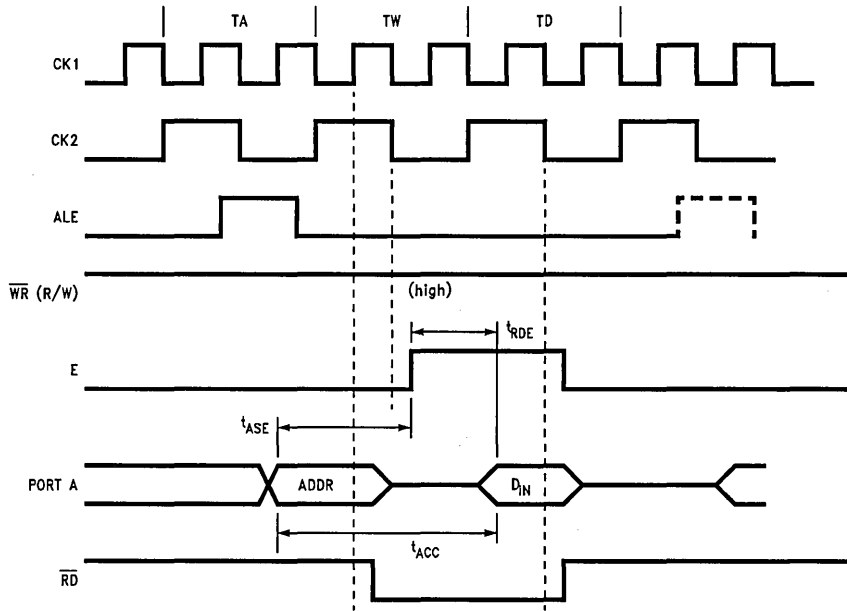


FIGURE 7. E Signal Read Cycle (1 Wait State)

TL/DD/11289-8

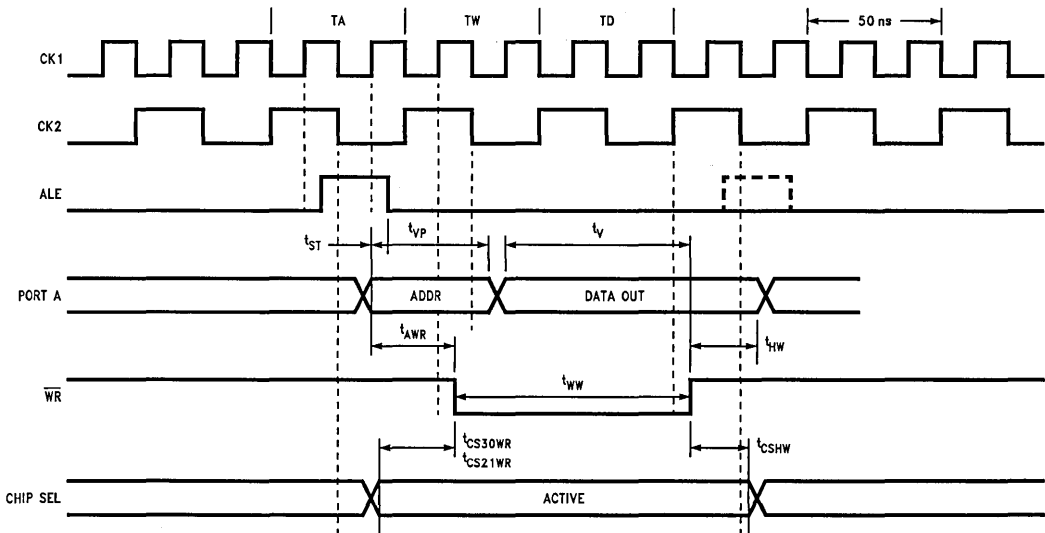


FIGURE 8. Native Bus Mode Write Cycle (1 Wait State)

TL/DD/11289-9

Timing Waveforms (Continued)

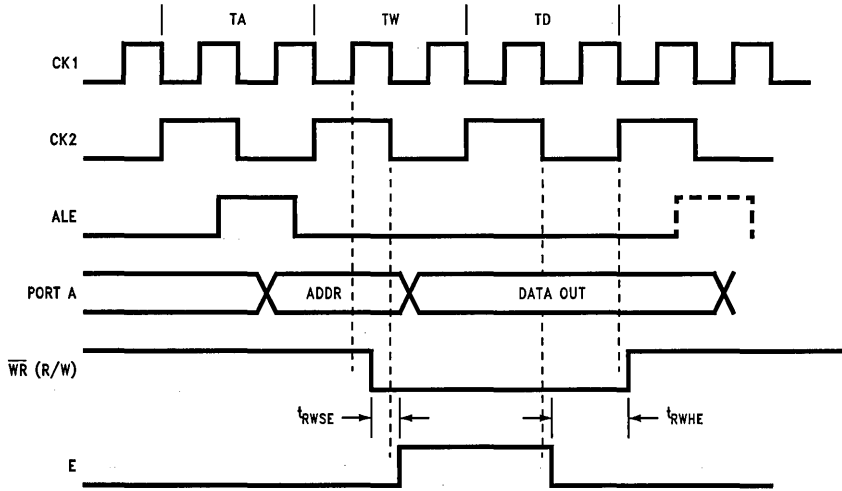


FIGURE 9. E Signal Write Cycle (1 Wait State)

TL/DD/11289-10

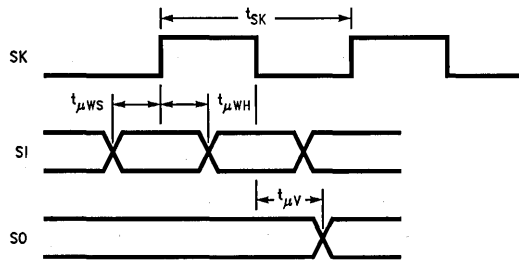
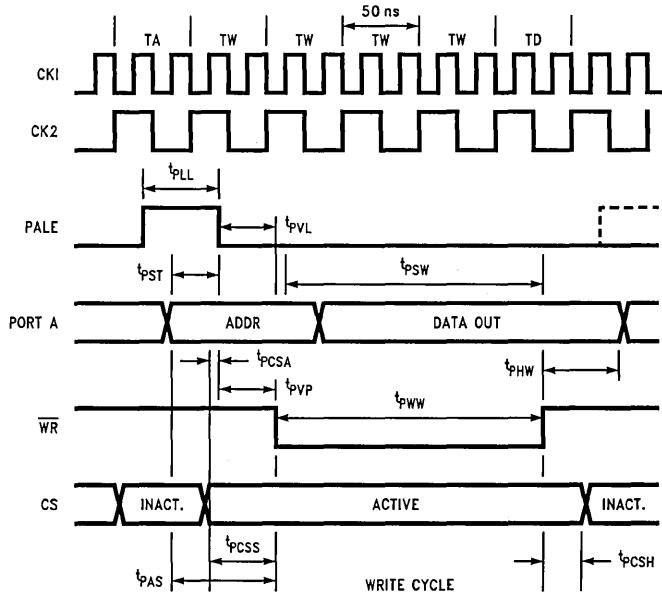


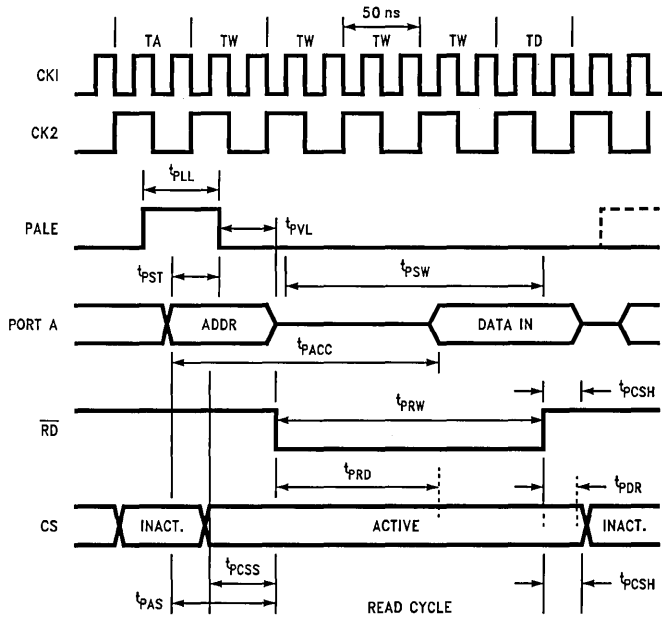
FIGURE 10. MICROWIRE™ Setup/Hold Timing

TL/DD/11289-11

Timing Waveforms (Continued)



TL/DD/11289-12



TL/DD/11289-13

FIGURE 11. Slow Peripheral Bus Timing

I/O Ports

PORT A

Port A is a 16-bit multiplexed address/data bus used for accessing external program and data memory. Four associated bus control signals are available on port B. The Address Latch Enable (ALE) signal is used to provide timing to demultiplex the bus. Reading from and writing to external memory are signalled by \overline{RD} and \overline{WR} respectively. External memory can be addressed as either bytes or words with the decoding controlled by two lines. Bus High Byte Enable (\overline{HBE}) and Address/Data line 0 (A0).

PORT B

Port B is a 16-bit port with 12 bits of bidirectional I/O. B10, B11, B12 and B15 are the control bus signals for the address/data bus. Port B may also be configured via a function register BFUN to individually allow each bidirectional I/O pin to have an alternate function. The direction of port B is determined by the direction register (DIRB). This register is used to set up each pin to be individually defined as an input or output. A specific I/O bit is selected as a high impedance input by clearing the corresponding bit in the direction register. It is selected as an output by setting this bit. The data register (PORTB) is used to hold data to be output on the B port. A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have

the same value, reading the pins returns the value of the data register. *Figure 12 through Figure 14* show the structure of Port B.

Port B may also be configured via a 16-bit alternate function register BFUN, to individually allow each pin to have an alternate function. The alternate functions are enabled by setting the corresponding bits in the BFUN register. The alternate B port functions are as follows:

Pin	Alternate	Function
B0	TDX	UART Data Output
B1	E	E signal output
B2	CKX	UART Clock
B3	T2IO	Timer2 I/O Pin
B4	T3IO	Timer3 I/O Pin
B5	SO	MICROWIRE/PLUS Output (data)
B6	SK	MICROWIRE/PLUS Clock
B7	\overline{HLDA}	Hold Acknowledge Output
B8	TS0/CS0	Timer Synchronous or Chip Select Output
B9	TS1/CS1	Timer Synchronous or Chip Select Output
B10	ALE	Address/data bus Address Latch Enable
B11	\overline{WR}	Address/data bus Write Output
B12	\overline{HBE}	Address/data bus High Byte Enable
B13	TS2/CS2	Timer Synchronous or Chip Select Output
B14	TS3/CS3	Timer Synchronous or Chip Select Output
B15	\overline{RD}	Address/data bus Read Output

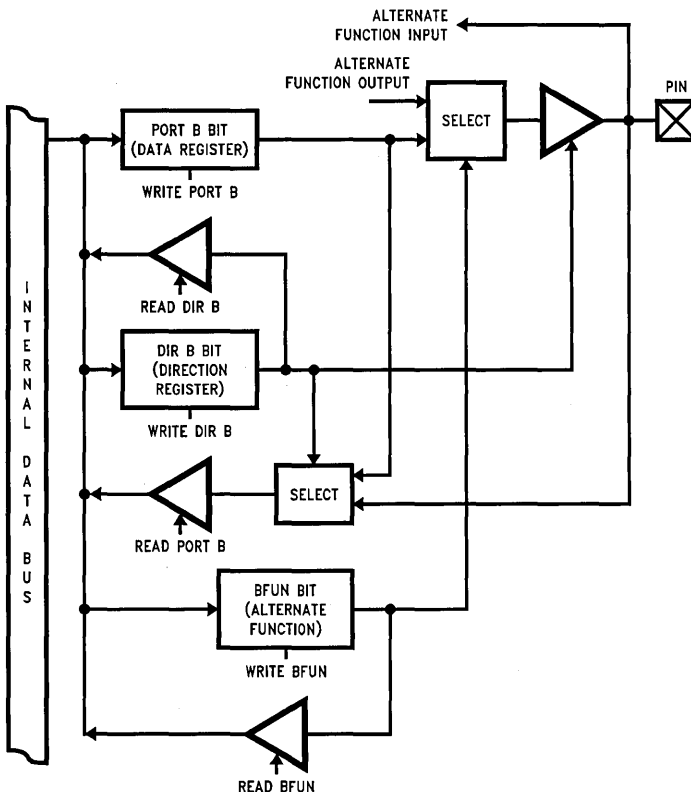


FIGURE 12. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

TL/DD/11289-14

I/O Ports (Continued)

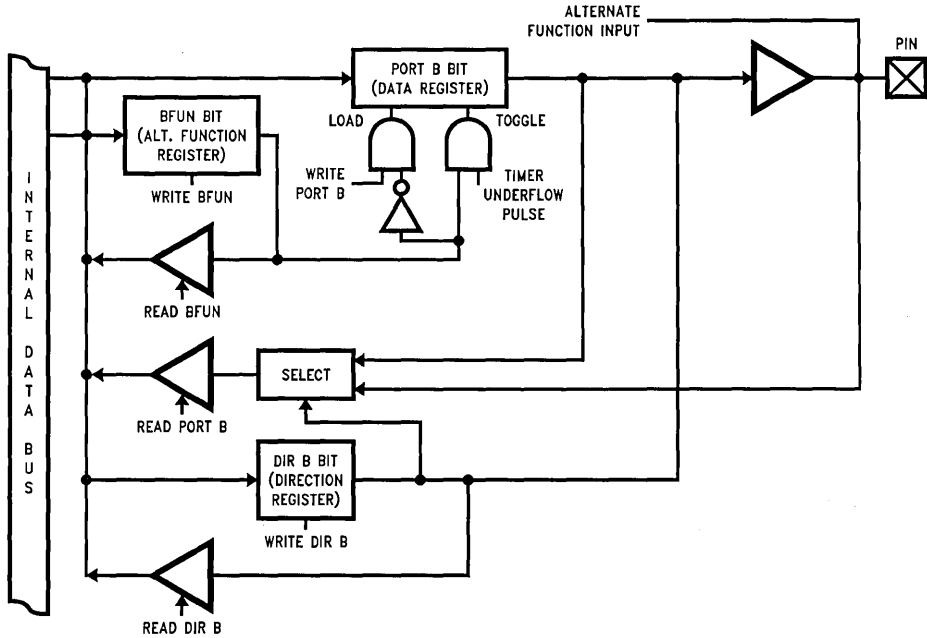


FIGURE 13. Structure of Port B Pins B3 and B4 (Timer Synchronous)

TL/DD/11289-15

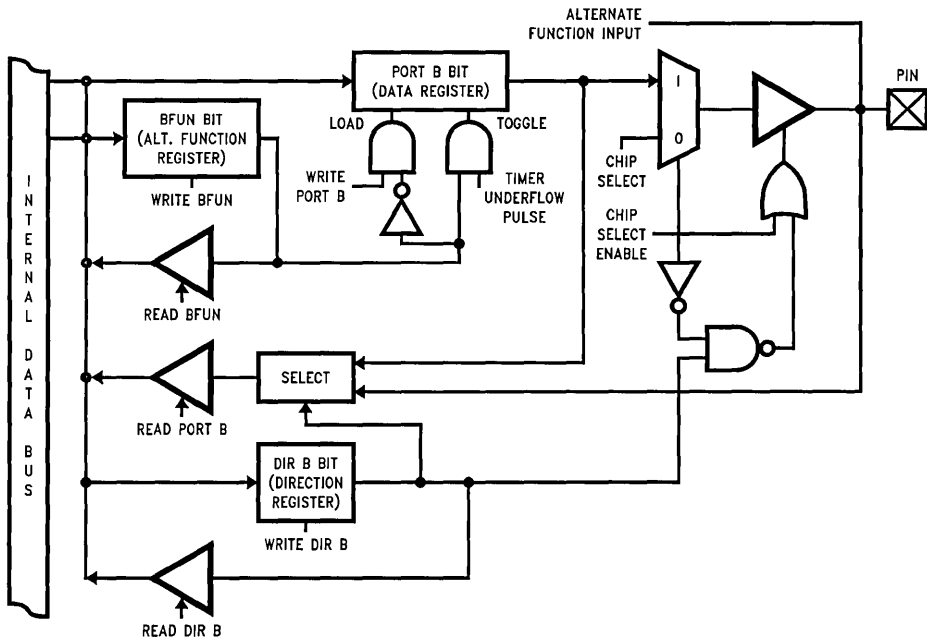


FIGURE 14. Structure of Port B Pins B8, B9, B13 and B14 (Timer Synchronous)

TL/DD/11289-16

I/O Ports (Continued)**PORT I**

Port I is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

Pin	Alternate	Function
I0	R5B	Timer T5 R5B Input
I1	NMI	Nonmaskable Interrupt Input
I2	INT2	Maskable Interrupt/Input Capture
I3	INT3	Maskable Interrupt/Input Capture
I4	INT4	Maskable Interrupt/Input Capture
I5	SI	MICROWIRE/PLUS Data Input
I6	RDX	UART Data Input
I7	R6B	Timer T6 R6B Input and A/D Trigger Input

PORT D

Port D is an 8-bit input port that can be used as general purpose digital inputs and as analog inputs for the A/D converter.

PORT P

Port P is a 3-bit input/output port that is used as general purpose outputs, or I/O that is controlled by timers T4, T5 and T6. These pins can be configured as Pulse Width Modulated Outputs (PWM), capture inputs or event counter inputs.

POWER SUPPLY PINS

Four pairs of power supply pins are provided to minimize cross talk between the analog, digital, and output driver sections of the chip.

Pin	Function
V _{CC}	Power for Digital Logic
GND	Ground for Digital Logic
DV _{CC}	Power for Output Drivers
DGND	Ground for Output Drivers
AV _{CC}	Power for Analog Logic
AV _{SS}	Ground for Analog Logic
V _{REF}	A/D Converter Reference Voltage Input
AGND	Ground Reference for Analog Logic

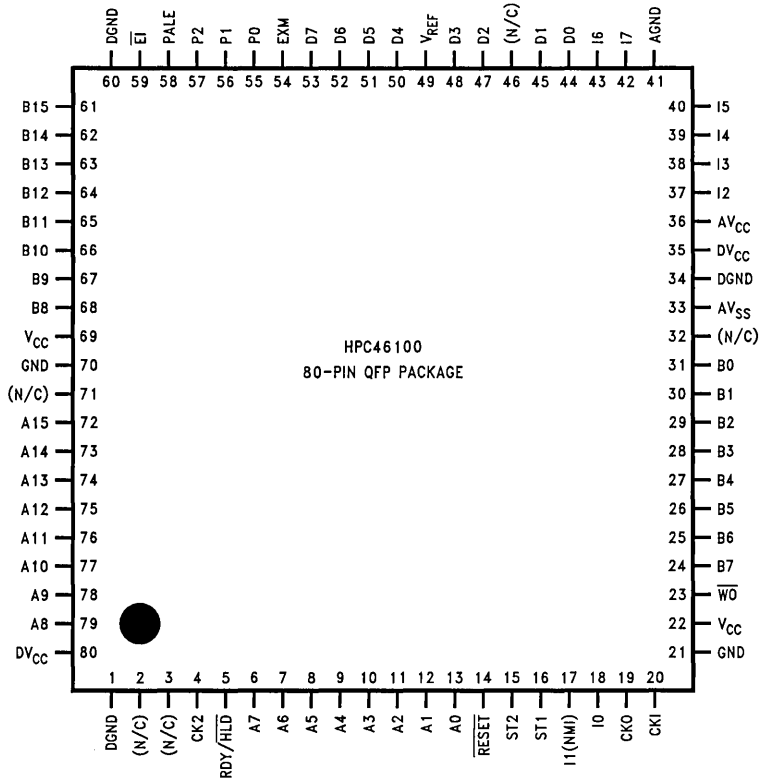
CLOCK PINS

Pin	Function
CKI	System Oscillator Input/External Clock Input
CKO	System Oscillator Output (Inversion of CKI)
CK2	Clock Output (CKI divided by 2)

OTHER PINS

Pin	Function
$\overline{\text{RESET}}$	System reset input, active low.
PALE	Slow peripheral address latch enable.
RDY/ $\overline{\text{HLD}}$	Has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or HOLD request input to put the bus in a high impedance state for DMA purposes.
$\overline{\text{WO}}$	This is an active low open drain output that signals an illegal situation has been detected by the WATCHDOG logic.
ST1	Bus Cycle Status Output: indicates first opcode fetch.
ST2	Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
EXM	External memory enable (active high) disables internal ROM and maps it to external memory.
$\overline{\text{EI}}$	Has two uses, it's either an active low level external interrupt with vector address FFF3:FFF2 which is shared with the UART, or Timer T4 R4B input.

Connection Diagram



TL/DD/11289-17

Operating Modes

The HPC46100 does not have any internal ROM, and has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1"). The EA bit in the PSW register of the HPC46100 is hard wired to a logic "1". The use of this bit is reserved. Currently, the HPC46100 is intended for use with external memory. The external memory may be any combination of ROM, RAM, or peripherals and may be configured with an 8-bit or 16-bit external address/data bus (see *Figure 16* and *Figure 17*). Up to 62k bytes of external memory may be accessed.

Wait States

The HPC46100 provides four selectable Wait States that allow access to slower memories. The Wait States are selectable by the state of two bits in the PSW register or by

two bits in the Chip Select Control registers. Additionally, the RDY input may be used to extend the instruction or memory access cycle, allowing the user to interface with slow memories and peripherals. There also is a slow peripheral bus mode when using the Chip Select logic.

Power Save Modes

Two power saving modes are available on the HPC46100: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the internal oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all internal RAM, registers and I/O are unaffected.

HALT MODE

The HPC46100 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities,

Power Save Modes (Continued)

including the clock and timers, are stopped when the HALT mode is entered with the NMI input high. When the HALT mode is entered with the NMI input low, the high speed timers (T4, T5 and T6), remain active. In the HALT mode, power requirements for the HPC46100 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the \overline{RESET} or the NMI. The \overline{RESET} input re-initializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC46100 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the internal oscillator, the high speed timers (T4, T5, and T6), and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the \overline{RESET} or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC46100 to resume normal operation.

HPC46100 Interrupts

Complex interrupt handling is easily accomplished by the HPC46100's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table I.

TABLE I. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
FFFF:FFFE	\overline{RESET}	0
FFFD:FFFC	Non-maskable external on rising edge of I1 pin	1
FFFB:FFFA	External interrupt on I2 pin	2
FFF9:FFF8	External interrupt on I3 pin	3
FFF7:FFF6	External interrupt on I4 pin	4
FFF5:FFF4	Overflow on internal timers	5
FFF3:FFF2	Internal by UART or external on \overline{EI} pin	6
FFF1:FFF0	A/D converter	7

INTERRUPT ARBITRATION

The HPC46100 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table I. The interrupt on \overline{RESET} has the highest rank and is serviced first.

INTERRUPT PROCESSING

Interrupts are serviced after the current instruction is completed and except for the \overline{RESET} , which is serviced immediately. \overline{RESET} and \overline{EI} are level-LOW-sensitive interrupts. All other external interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge.

INTERRUPT CONTROL REGISTERS

The HPC46100 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

\overline{RESET} and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector excluding the \overline{EI} interrupt which has a dedicated register containing an interrupt enable and pending bit. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC46100 after servicing the interrupts. For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software. The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no effect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3 and I4.

\overline{EI} INTERRUPT CONFIGURATION REGISTER (EICON)

The \overline{EI} pin is an active low level sensitive interrupt. Interrupts from the \overline{EI} pin are enabled by using the EICON register.

SERVICING THE INTERRUPTS

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack then incrementing the stack pointer (SP) by two. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 15* shows the interrupt enable logic.

RESET

\overline{RESET} is an active-low Schmitt trigger input that initializes the processor and sets all pins in a TRI-STATE condition except for CK0, CK1, ST1, ST2, \overline{HBE} and \overline{WO} when held low. When rising edge is detected on \overline{RESET} , the processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location.

HPC46100 Interrupts (Continued)

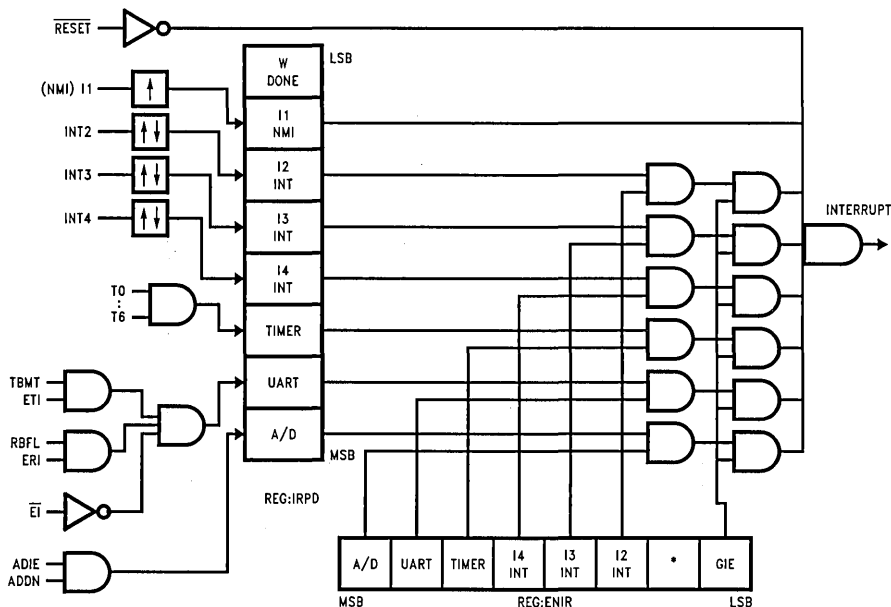


FIGURE 15. Interrupt Enable Logic

TL/DD/11289-18

Multiply/Accumulate Unit (MAU)

This is dedicated hardware that is supported by instructions which perform basic multiply-accumulate DSP steps for FIR and IIR filter calculations, an arithmetic right shift of the Math unit Result Register (MRR), and a signed multiply of two 16-bit values producing a 32-bit result.

The MACZ and MAC instructions support the MAU by fetching data and performing circular buffer management in parallel with the multiplication. The MACZ instruction is used initially, and it is followed by a string of additional MAC instructions, one instruction per filter tap (including the MACZ). The source values are taken as 16-bit values; there is not a form that operates on byte-wide inputs. The MACZ instruction clears the result register before completing the first multiply operation.

The MAC instruction is opcode 38 hex. The MACZ instruction is opcode 39 hex. Both instructions are one byte in length, and neither allows use of an Addressing Directive prefix.

The specific function performed is as follows:

- Clear MRR to zero (MACZ instruction only).
- Fetch 16-bit data pointed to by B, and issue it to the MAU as the first operand.
- Increment B by two (bytes), compare B with K: if $B > K$ then load B from A.
- Fetch 16-bit data pointed to by X, increment X by two (bytes).
- Issue data to the MAU to start multiply-accumulate operation. The MAU multiplies the two operands issued to it, and adds the 32-bit result to the current 32-bit contents of the MRR register.

On completion, the result goes to the MRR register. The MVP bit in the PSW is set to a "1" if a signed (2's complement) overflow occurred in the positive direction as a result of the accumulation substep (overflow from the multiplication substep is impossible). If the overflow occurred in the negative direction, the MVN bit is set instead. Neither of these bits is affected by the MAU if the other is already set.

By stringing together a sequence of MAC instructions, the HPC46100 can do a multiply-accumulate every 9 cycles (assuming a 1 wait state instruction fetch). At 40 MHz, this gives a 450 ns multiply-accumulate (including data fetching and circular buffer management). This can be reduced to 400 ns if executed from internal RAM.

Chip Select Signals

CHIP SELECT LOGIC

The chip select logic can produce up to four chip select signals without any off-chip logic. The chip select logic supports two bus timing modes. The first bus timing mode is native bus mode, which is the standard bus mode of all HPC family members. The second bus mode is the slow peripheral bus mode, which allows the HPC46100 to interface with slow peripherals without external chip select logic. There is an additional data strobe provided for auxiliary bus timing. This auxiliary strobe is called the E signal.

Each of chip select signals is controlled by a dedicated chip select control register (CSC0–CSC3). The control registers contain one bit to enable/disable the chip select signals, one bit to select the polarity of the chip select signal, two bits program the wait states of the data accesses, four bits that select the address range which the chip select signals are active in and one bit to enable or disable the E signal.

Chip Select Signals (Continued)

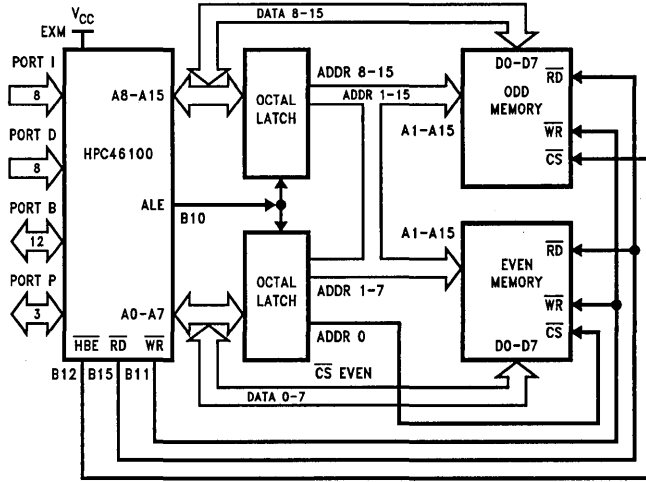


FIGURE 16. 16-Bit External Address/Data Bus

TL/DD/11289-19

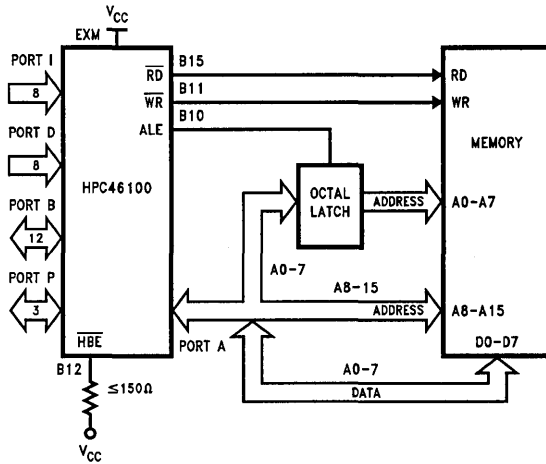
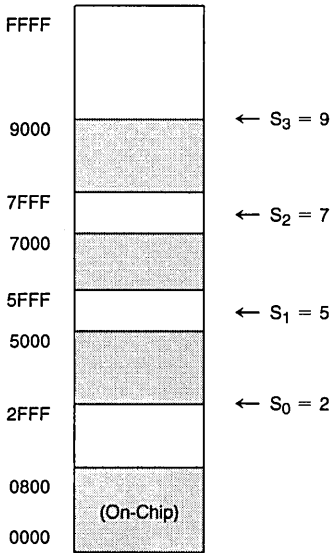


FIGURE 17. 8-Bit External Address/Data Bus

TL/DD/11289-20

Chip Select Signals (Continued)

The Chip Select Address Range Selection (SEL) defines the address range the chip select is valid over. For Chip Select 0 (CSC0), the address range starts at location 0800 hex and extends to S₀FFF hex, where S₀ is the 4-bit contents of the SEL field. For both Chip Select 1 (CSC1) and Chip Select 2 (CSC2) the SEL field defines a single 4 kbyte range: S₁000 through S₁FFF for CSC1, and S₂000 through S₂FFF for CSC2, where S₁ and S₂ are their respective SEL field contents. These ranges can be used to control peripherals by using the "Slow Peripheral" bus timing mode. Chip Select 3 (CSC3) defines a range beginning at S₃000 hex, and continuing through FFFF hex. This range will typically define the off-chip ROM space. See Figure 18.



Shaded Ranges Present
No Chip Select, and
Global Bus Features
are Selected

FIGURE 18. Chip Select Address Ranges

Chip Select 0 and Chip Select 3 can be programmed independently to operate with 1, 2 or 4 wait states. Chip Select 1 and 2 can be programmed independently to operate in native bus mode with 1, 2 or 4 wait states or slow peripheral bus mode.

Timer Overview

The HPC46100 contains seven 16-bit timers, four core timers and three high speed timers. Timers T0–T3 are the standard core timers and are fully compatible with the core timers on other HPC family members. See Figure 19 and Figure 20.

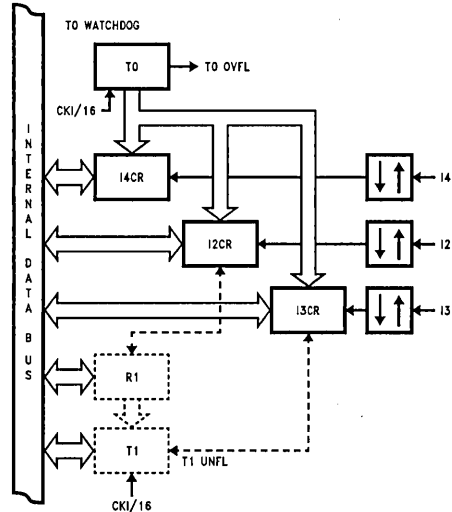


FIGURE 19. Timers T0–T1 Block

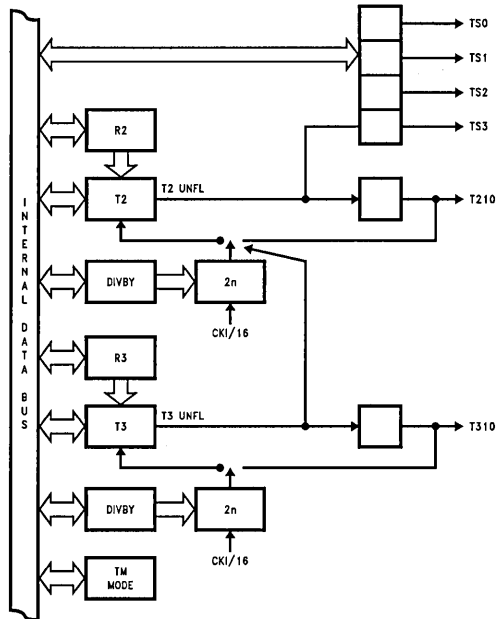


FIGURE 20. Timers T2–T3 Block

Timer Overview (Continued)

CORE TIMERS

Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watchdog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. Registers I2CR and I3CR have the alternate function of being R1 and T1 respectively. The function of these registers (I2CR/R1 and I3CR/T1) are mutually exclusive and under the control of software. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt.

The timers T2 and T3 have a clock rate which is selectable. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter.

The timers T1 through T3 in conjunction with their registers form Timer-Register pairs. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

Synchronous Outputs

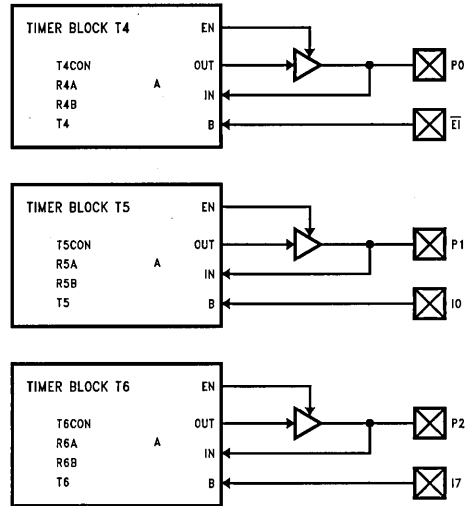
There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflow.

Note: These outputs are shared with the chip select outputs. The use of these two functions are mutually exclusive.

TIMERS T4, T5 AND T6

The HPC46100 has a set of three powerful timers/counters, T4, T5 and T6. Since the three timers, T4, T5 and T6 are identical, all comments are equally applicable to any of the three timer blocks.

These timers are designed to allow the device to easily perform all timer functions with minimal software overhead. All timers are synchronized on the first overflow of timer T0. Each timer has four 16-bit registers dedicated to it, a control register (TnCON), timer register (Tn), and two auto-load/capture registers (RnA, RnB). *Figure 21* shows a block diagram of the three high speed timers.



TL/DD/11289-23

FIGURE 21. High Speed Timers Block

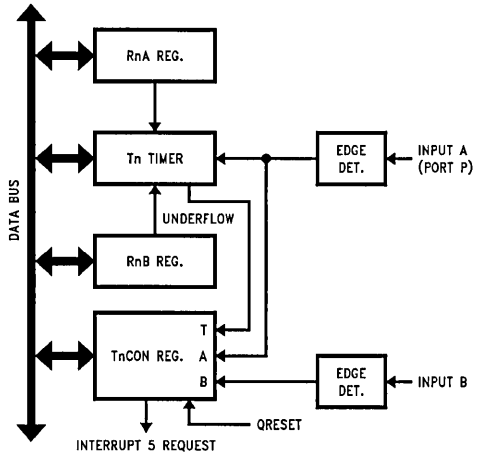
TIMER CONTROL REGISTERS

There are three timer control registers (T4CON, T5CON and T6CON). These control registers have bits which set the clock input rate, mode of operation and interrupt control structure. Each timer control register has interrupt pending and interrupt acknowledge bits for Tn, RnA and RnB, and a global interrupt pending bit for that specific timer. There are bits to enable/disable the interrupts from Tn, RnA and RnB. The clock input rate can be selected to be CKI/2, CKI/4, CKI/8 or CKI/16. The Four modes of operation are: External Event Counter mode, Input Capture mode, Processor Independent PWM mode and externally triggered PWM mode.

MODE 0. EXTERNAL EVENT COUNTER MODE

This mode is the default after RESET. In this mode the timer register Tn is decremented each time there is an active edge on the A input. The active edge is determined by the value of a bit in the control register. Upon every underflow of the timer register (Tn), the timer (Tn) is alternately reloaded with the contents of the supporting registers RnA and RnB. The first underflow of the timer after entry into this mode will cause the timer to reload from register RnA. All following underflows will alternate which reload is used beginning with RnB. Every underflow from the timer will set a Tn global interrupt pending bit in the control register. The selected edge on Input A and Input B will also set corresponding pending bits in the control register. *Figure 22* shows a block diagram of the high speed timers in Mode 0.

Timer Overview (Continued)

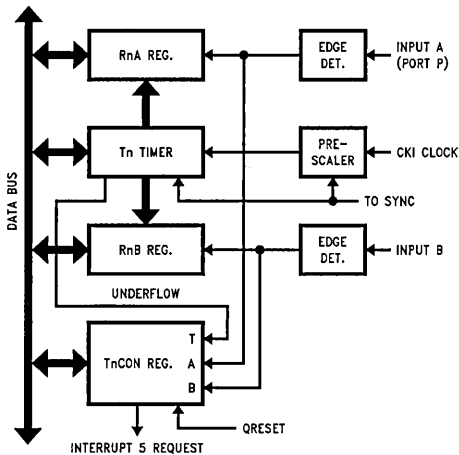


TL/DD/11289-24

FIGURE 22. External Event Counter

MODE 1. DUAL INPUT CAPTURE MODE

The device can precisely measure external frequencies or time external events by placing the timer in the input capture mode. In this mode, the timer T_n is constantly running at a fixed rate as selected in the timer control register. The two registers, R_nA and R_nB , act as capture registers. Each register is loaded with the contents of the timer register T_n when an active edge on it's associated pin is detected. The active edge is determined by the value of two bits in the control register. The active edge for each input pin can be specified independently, and can be programmed to generate an interrupt. The interrupt can be generated on an input from the A or B input along with an underflow of the timer register (T_n). Figure 23 shows a block diagram of the timer in Input Capture mode.

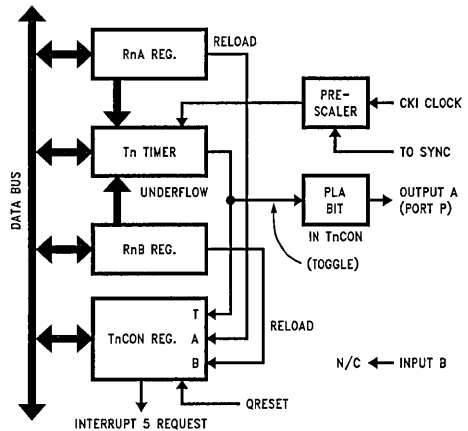


TL/DD/11289-25

FIGURE 23. Dual Input Capture

MODE 2. PROCESSOR INDEPENDENT PWM MODE

As the name suggests, this mode allows the device to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating. In this mode the timer T_n counts down at a fixed rate as programmed in the control register. Upon every underflow the timer is alternately reloaded with the contents of its supporting registers, R_nA and R_nB . The very first underflow of the timer after entry into this mode causes the timer to reload from the register R_nA . Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register R_nB . Figure 24 shows a block diagram of the timer in PWM mode.



TL/DD/11289-26

FIGURE 24. PWM

The underflow can be programmed to toggle the A output pin (Port P). The underflow can also be programmed to generate interrupts. These interrupts can occur on a reload from R_nA or R_nB . This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

MODE 3. EXTERNALLY TRIGGERED PWM/PORT OUTPUT

In this mode the timer is stopped and the corresponding port P pin can be programmed as a general purpose output pin. The timer block can be programmed to remain halted using its A pin for use as a general purpose output, or it can be programmed to exit mode 3 and enter mode 2 (Processor Independent PWM) when a rising edge is detected on the B input.

The external triggering of this feature provides the capability of generating a delayed pulse triggered by an external event as follows: From the rising edge of the B input the timer enters PWM mode. When the timer register underflows the output toggles and the value of R_nA is loaded to the timer. The next underflow will cause the R_nB register to load into

Timer Overview (Continued)

the timer and operation will continue as in the PWM mode. Alternately, an interrupt from the RnB load can be used to branch to a routine that would set the timer into mode 3 waiting for the trigger for another pulse. *Figure 25* shows a block diagram of the timer in this mode.

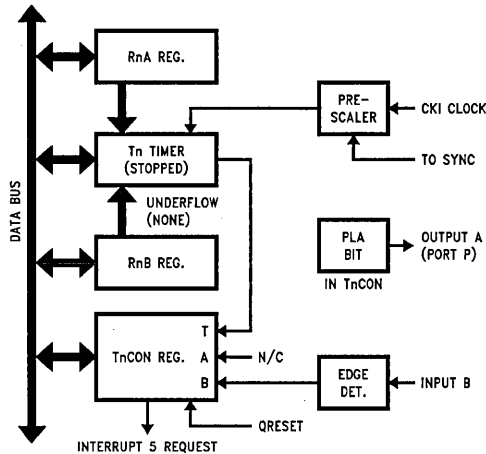


FIGURE 25. Port Output/Externally Triggered PWM

TL/DD/11289-27

WATCHDOG LOGIC

The WATCHDOG Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the WATCHDOG logic are potentially infinite loops. Should the WATCHDOG register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. Any illegal condition forces the WATCHDOG Output ($\overline{W0}$) pin low. The $\overline{W0}$ pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications and has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed. The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS OPERATION

The HPC46100 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC46100 is the master or slave. The shift clock is generated when the HPC46100 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC46100 is configured as a slave.

When the HPC46100 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz. The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

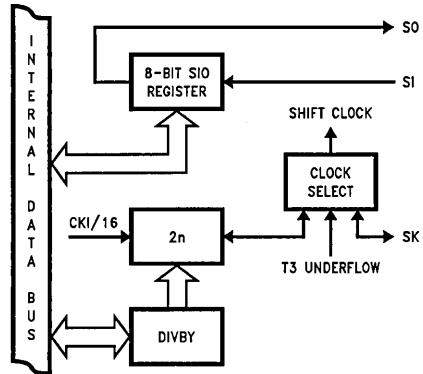


FIGURE 26. MICROWIRE/PLUS

TL/DD/11289-28

HPC46100 UART

The HPC46100 contains a software programmable UART. The UART (see *Figure 27*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing, parity, and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame, reporting receiving and transmitting status, and enabling or disabling the UART's Attention Mode of operation.

The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits ($7/8$, 1, $1\frac{1}{8}$ or 2 stop bits), selecting between the synchronous or asynchronous mode and enabling or disabling transmit and receive interrupts. The clock inputs to the Transmitter and Receiver sections of the UART can be individually selected to come from either an off-chip source on the CKX pin or one of the two on-chip sources. The Divide-By (DIVBY) Register provides upward compatibility from earlier HPC family members, and the most flexible and accurate on-chip clocking is provided by the Baud Rate Generator (BRG).

HPC46100 UART (Continued)

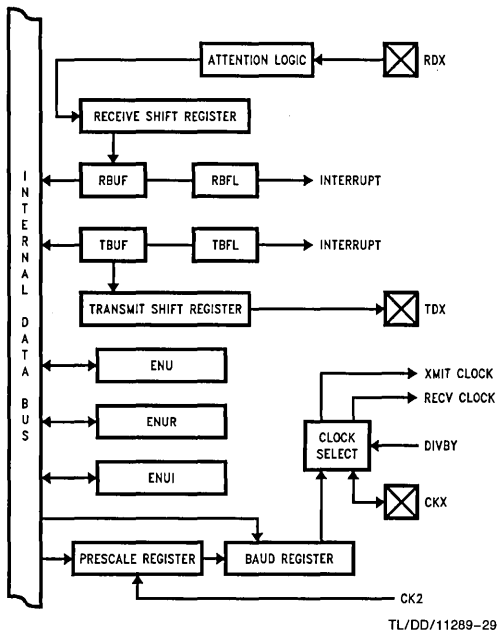


FIGURE 27. UART Block Diagram

The Baud Rate Generator is controlled by the register pair PSR and BAUD. The Prescaler factor is selected by the upper 5 bits of the PSR register (the PRESCALE field), in units of the CK2 clock from 1 to 16 in $\frac{1}{2}$ step increments. The lower 3 bits of the PSR register, in conjunction with the 8 bits of the baud register, form the 11-bit BAUDRATE field, which defines a baud rate divisor ranging from 1 to 2048, in units of the prescaled clock selected by the PRESCALE field.

In Asynchronous Mode, the resulting baud rate is $\frac{1}{16}$ of the clocking rate selected through the BRG circuit. The maximum baud rate generated using the BRG is 625 kbaud.

In the Synchronous Mode data is transmitted on the rising edge and received on the falling edge of the external clock. Although the data is transmitted and received synchronously, it is still contained within an asynchronous frame; i.e., a start bit, parity bit (if selected) and stop bit(s) are still present.

UART ATTENTION MODE

The HPC46100 UART features an Attention Mode of operation. This mode of operation enables the HPC46100 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data.

Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0. The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC46100 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

A/D Converter

The HPC46100 has an on-board eight-channel 8-bit Analog to Digital converter. Conversion is performed using a successive approximation technique. The A/D converter cell can operate in single-ended mode where the input voltage is applied across one of the eight input channels (D0–D7) and AGND or in differential mode where the input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel-pairs in differential mode.

OPERATING MODES

The operating modes of the converter are selected by 4 bits called ADMODE (ADCR2.4–7) see Table II. Associated with the eight input channels in single-ended mode are eight result registers, one for each channel. The A/D converter can be programmed by software to convert on any specific channel storing the result in the result register associated with that channel. It can also be programmed to stop after one conversion or to convert continuously. If a brief history of the signal on any specific input channel is required, the converter can be programmed to convert on that channel and store the consecutive results in each of the result registers before stopping. As a final configuration in single-ended mode, the converter can be programmed to convert the signal on each input channel and store the result in its associated result register continuously.

Associated with each even-odd pair of input channels in differential mode of operation are four result register-pairs. The A/D converter performs two conversions on the selected pair of input channels. One conversion is performed assuming the positive connection is made to the even channel and the negative connection is made to the following odd channel. This result is stored in the result register associated with the even channel. Another conversion is performed assuming the positive connection is made to the odd channel and the negative connection is made to the preceding even channel. This result is stored in the result register associated with the odd channel. This technique does not require that the programmer know the polarity of the input signal. If the even channel result register is non-zero (meaning the odd channel result register is zero), then the input signal is positive with respect to the odd channel. If the odd channel result register is non-zero (meaning the even channel result register is zero), then the input signal is positive with respect to the even channel.

A/D Converter (Continued)

TABLE II. Operating Modes

Mode 0	single-ended, single channel, single result register, one-shot (default value on power-up)
Mode 1	single-ended, single channel, single result register, continuous
Mode 2	single-ended, single channel, multiple result registers, stop after 8
Mode 3	single-ended, multiple channel, multiple result register, continuous
Mode 4	differential, single channel-pair, single result register-pair, one-shot
Mode 5	differential, single channel-pair, single result register-pair, continuous
Mode 6	differential, single channel-pair, multiple result register-pairs, stop after 4 pairs
Mode 7	differential, multiple channel-pair, multiple result register-pairs, continuous
Mode 8	single-ended, single channel, single result register, one-shot (default value on power-up), quiet address/data bus
Mode C	differential, single channel-pair, single result register-pair, one-shot, quiet address/data bus

The same operating modes for single-ended operation also apply when the inputs are taken from channel-pairs in differential mode. The programmer can configure the A/D to convert on any selected channel-pair and store the result in its associated result register-pair then stop. The A/D can also be programmed to do this continuously. Conversion can also be done on any channel-pair storing the result into four result register-pairs for a history of the differential input. Finally, all input channel-pairs can be converted continuously.

The final mode of operation suppresses the external address/data bus activity during the single conversion modes. These quiet modes of operation utilize the RDY function of the HPC Core to insert wait states in the instruction being executed in order to limit digital noise in the environment due to external bus activity when addressing external memory.

CONTROL

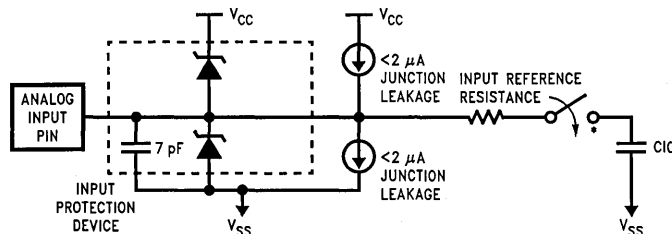
The conversion clock supplied to the A/D converter can be selected by three bits in ADCR1. These bits are used as a prescaler on CKI, and can provide a clock rate from CKI/4 to CKI/32. These bits can be used to ensure that the A/D is clocked as fast as possible when different external crystal frequencies are used. Controlling the starting of conversion cycles in each of the operating modes can be done by four different methods. The method is selected by two bits called SC (ADCR3.0–1). Conversion cycles can be initiated through software by resetting a bit in a control register, through hardware by an underflow of Timer T2, or externally by a rising or falling edge of a signal input on I7.

INTERRUPTS

The A/D converter can interrupt the HPC when it completes a conversion cycle if one of the non-continuous modes has been selected. If one of the cycle modes was selected, then the converter will request an interrupt after eight conversions. If one of the one-shot modes was selected, then the converter will request an interrupt after every conversion. When this interrupt is generated, the HPC vectors to the A/D converter interrupt vector location at address FFF0.

Analog Input and Source Resistance Considerations

Figure 28 shows the A/D pin model for the HC46100 in single ended mode. The differential mode has similar A/D pin model. The leads to the analog inputs should be kept as short as possible. Both noise and digital clock coupling to an A/D input can cause conversion errors. The clock lead should be kept away from the analog input line to reduce coupling. The A/D channel input pins do not have any internal output driver circuitry connected to them because this circuitry would load the analog input signals due to output buffer leakage current.



TL/DD/11289-34

*The analog switch is closed only during the sample time.

FIGURE 28. Port D Input Structure

A/D Converter (Continued)

Source impedances greater than 1 kΩ on the analog input lines will adversely affect internal RC charging time during input sampling. As shown in Figure 28, the analog switch to the DAC array is closed only during the 2 A/D cycle sample time. Large source impedances on the analog inputs may result in the DAC array not being charged to the correct voltage levels, causing scale errors.

If large source resistance is necessary, the recommended solution is to slow down the A/D clock speed in proportion to the source resistance. The A/D converter may be operated at the maximum speed for R_S less than 1 kΩ. For R_S greater than 1 kΩ, A/D clock speed needs to be reduced. For example, with R_S = 2 kΩ, the A/D converter may be operated at half the maximum speed. A/D converter clock speed may be slowed down by either increasing the A/D prescaler divide-by or decreasing the CK1 clock frequency. The A/D clock speed may be reduced to its minimum frequency of 100 kHz.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block (see Figure 29). The HPC46100 supports shared memory access with two pins. The pins are the RDY/HLD input pin and the HLDA output pin. The user can software select either the Hold or Ready function by the state of a control bit. The HLDA output is multiplexed onto port B.

The host uses DMA to interface with the HPC46100. The host initiates a data transfer by activating the HLD input of the HPC46100. In response, the HPC46100 places its system bus in a TRI-STATE Mode, freeing it for use by the host.

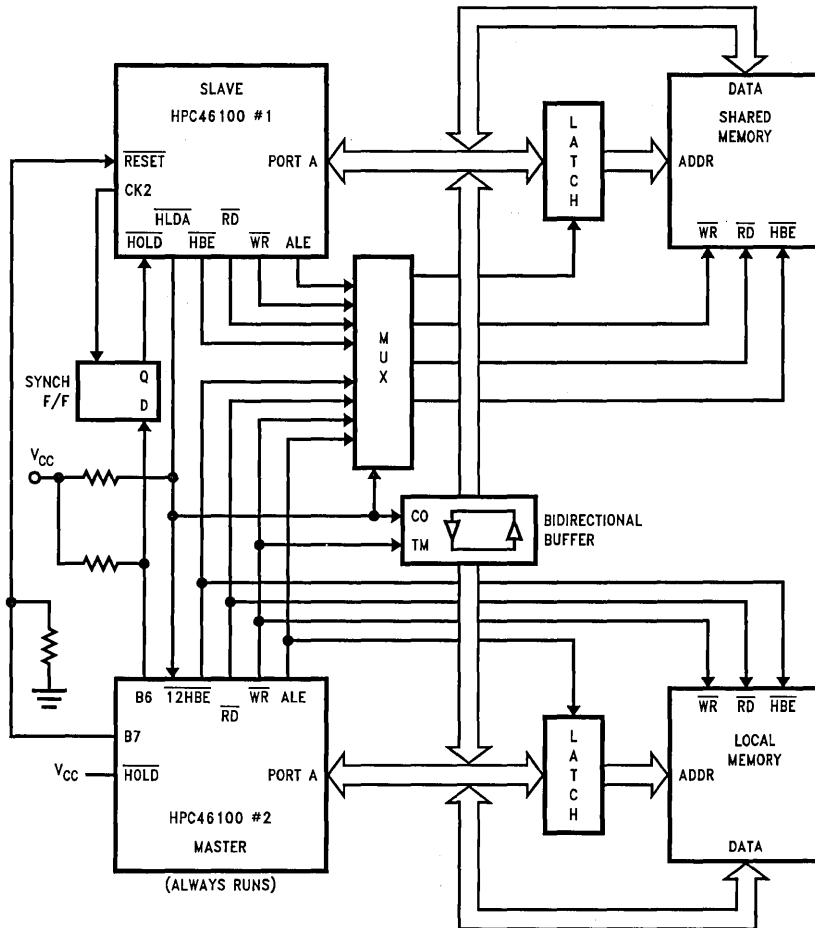


FIGURE 29. Shared Memory Application, Using $\overline{\text{HOLD}}$

TL/DD/11289-31

Shared Memory Support (Continued)

The host waits for the acknowledge signal (HLDA) from the HPC46100 indicating that the system bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC46100 resumes normal operations.

Memory

The HPC46100 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be

directly addressed including 1024 bytes of RAM available on the chip itself.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through registers or any memory word in the first 256 bytes of memory (On-Chip Basepage RAM). Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC46100 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions. The HPC46100 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table III.

TABLE III. HPC46100 Memory Map

FFFF:FFF0 FFEF:FFD0 FFCF:0800	Interrupt Vectors JSRP Vectors External Memory	User Memory
07FF:04C0	On-Chip RAM	User RAM
04BF:0196	RESERVED	
0195:0194	Watchdog Register	Watchdog Logic
0192 0191:0190 018F:018E 018D:018C 018B:018A 0189:0188 0187:0186 0185:0184 0183:0182 0181:0180	T0CON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/R1 I3CR Register/T1 I4CR Register	Timer Block T0:T3
017F:0168	RESERVED	
0167:0166 0165:0164 0163:0162 0161:0160	CSC3 Register CSC2 Register CSC1 Register CSC0 Register	Chip Select Control
015F:0158	RESERVED	
0157:0156 0155:0154 0153:0152 0151:0150	T6 Timer T6CON Register R6A Register R6B Register	Timer T6
014F:012D	RESERVED	
012C 012A 0128 0126 0124 0122 0120	Baud Register PSR Register ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART
011E 011C 011A 0118 0116 0114 0112 0110 010F:0109	A/D Result Register 7 A/D Result Register 6 A/D Result Register 5 A/D Result Register 4 A/D Result Register 3 A/D Result Register 2 A/D Result Register 1 A/D Result Register 0 RESERVED	A/D Converter

0108	EICON Register	EI Pin Control
0106 0104 0102 0100	ADCR3 Register PORTD Register ADCR2 Register ADCR1 Register	A/D Control
00FF:00FE 00FD:00FC 00FB:00FA 00F9:00F8	T5 TIMER T5CON Register R5A Register R5B Register	Timer 5
00F7:00F6	RESERVED	
00F5:00F4 00F3:00F2 00F1:00F0	BFUN Register DIR B Register RESERVED FOR DIRA	Ports A&B Control
00EF:00EE 00ED:00EC 00EB:00EA 00E9:00E8 00E7:00E6	T4 TIMER T4CON Register R4A Register R4B Register RESERVED FOR UPIC	Timer 4
00E5:00E6	RESERVED	
00E3:00E2 00E1:00E0	PORTB Register RESERVED FOR PORTA	Ports A&B
00DF:00DE 00DD:00DC 00DA	MRU (MRR upper) MRL (MRR lower) MIR	Math Unit
00D8 00D6 00D4 00D2 00D0	PORTI Register SIO Register IRCD Register IRPD Register ENIR Register	Interrupt Control Registers
00CF:00CE 00CD:00CC 00CB:00CA 00C9:00C8 00C7:00C6 00C5:00C4 00C3:00C2 00C1:00C0	X Register B Register K Register A Register PC Register SP Register HALTEN Register PSW Register	HPC Core Registers
00BF:0000	On-Chip RAM	Basepage RAM

Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused outputs should be left floating to allow the output to switch without drawing any DC current.

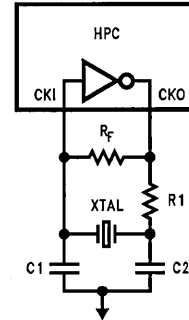
To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least $1\ \mu\text{F}$ and bypass their outputs with a $10\ \mu\text{F}$ to $50\ \mu\text{F}$ tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a $10\ \mu\text{F}$ to $20\ \mu\text{F}$ tantalum electrolytic capacitor or a $50\ \mu\text{F}$ to $100\ \mu\text{F}$ aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately $10\ \text{nF}$ (spaced within $12\ \text{cm}$) per every two to five packages, and $100\ \text{nF}$ for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.

A recommended crystal oscillator circuit to be used with the HPC is shown in *Figure 30*. See table for recommended component values. The recommended values given in the table have yielded consistent results and are made to match a crystal with a $18\ \text{pF}$ load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within "1" distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal. It is

very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A $1.0\ \mu\text{F}$, a $0.1\ \mu\text{F}$, and a $0.001\ \mu\text{F}$ dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.



TL/DD/11289-32

FIGURE 30. Recommended Crystal Circuit

XTAL Frequency (MHz)	R1 (Ω)
≤ 2	1500
4	1200
6	910
8	750
10	600
12	470
14	390
16	300
18	220
20	180
22	150
24	120

$R_F = 3.3\ \text{M}\Omega$
 $C_1 = 27\ \text{pF}$
 $C_2 = 33\ \text{pF}$

XTAL Specifications: The crystal used was an M-TRON Industries MP-1 Series XTAL. "AT" cut parallel resonant

$C_L = 18\ \text{pF}$
 Series Resistance is
 $25\ \Omega @ 25\ \text{MHz}$
 $40\ \Omega @ 10\ \text{MHz}$
 $600\ \Omega @ 2\ \text{MHz}$

HPC46100 CPU

The HPC46100 CPU has a 16-bit ALU and six 16-bit registers.

ARITHMETIC LOGIC UNIT (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle.

The ALU has two carry bits; one for signed overflow (V bit) and one for unsigned overflow (C bit).

ACCUMULATOR (A) REGISTER

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

ADDRESS (B AND X) REGISTERS

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

BOUNDARY (K) REGISTER

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory. The K register can also be used as a pointer register.

STACK POINTER (SP) REGISTER

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits. The SP register can also be used as a pointer register.

PROGRAM (PC) REGISTER

The 16-bit PC register addresses program memory.

MAU RESULT REGISTER (MRR)

The 32-bit MAU Result Register holds the results from MAC (Multiply/Accumulate) instructions. In addition, it receives the result of the MULS (Multiply Signed) instruction, and can be shifted in place by the ASHR (Arithmetic Shift Right) operation.

Addressing Modes

ADDRESSING MODES WITH THE ACCUMULATOR AS DESTINATION

Register Indirect

The operand is the memory addressed by the A, B, X or K register. This mode of addressing for the HPC46100 produces single byte instructions when using the B or X register (depending on the instruction).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect Auto Increment and Decrement with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES WITH DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set, IF and IF NOT bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC46100 has been designed to be extremely code-efficient. The HPC46100 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC46100, and the code savings over other popular microcontrollers has been considerable. Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC46100 are single-byte. There are two especially code-saving instructions: JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

Code Efficiency (Continued)

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTION FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC46100 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following.

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC46100 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	$MA + Mem1 \rightarrow MA$, carry $\rightarrow C$
ADC	Add with carry	$MA + Mem1 + C \rightarrow MA$, carry $\rightarrow C$
SUB	Subtract without carry	$MA - Mem1 \rightarrow MA$
SUBC	Subtract with carry	$MA - Mem1 + C \rightarrow MA$, carry $\rightarrow C$
MULT	Multiply (unsigned)	$MA * Mem1 \rightarrow MA \& X$, 0 $\rightarrow K$, 0 $\rightarrow C$
DIV	Divide (unsigned)	$MA / Mem1 \rightarrow MA$, rem. $\rightarrow X$, 0 $\rightarrow K$, 0 $\rightarrow C$
DIVD	Divide Double Word (unsigned)	$X \& MA / Mem1 \rightarrow X$, 0K, Carry $\rightarrow C$
IFEQ	If equal	Compare MA & Mem1, Do next if equal
IFGT	If greater than	Compare MA & Mem1, Do next if $MA > Mem1$
IFGE	If greater than or equal	Compare MA & Mem1, Do next if $MA =$ or $> Mem1$
IFGESS	If greater than or equal signed	Signed compare MA & Mem1, Do next if $MA =$ or $> Mem1$
IFGTS	If greater than signed	Signed compare MA & Mem1, Do next if $MA > Mem1$
AND	Logical and	$MA \text{ and } Mem1 \rightarrow MA$
OR	Logical or	$MA \text{ or } Mem1 \rightarrow MA$
XOR	Logical exclusive-or	$MA \text{ xor } Mem1 \rightarrow MA$
MACZ	Multiply-accumulate From Zero signed word	0 $\rightarrow MRR$, $[B+] * [X+] + MRR \rightarrow MRR$ OVERFLOW $\rightarrow MVP$ or MVN , IF $B > K$ THEN $A \rightarrow B$
MAC	Multiple-accumulate signed word	$[B+] * [X+] + MRR \rightarrow MRR$, OVERFLOW $\rightarrow MVP$ or MVN , IF $B > K$ then $A \rightarrow B$
MULS	Multiply, signed	$MA * Mem1 \rightarrow MRR$
ASHR	Arithmetic right	$MRR / 2^{imm} \rightarrow MRR$, or $MRR / 2^A \rightarrow MRR$
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	$Mem + 1 \rightarrow Mem$
DECSZ	Decrement, skip if 0	$Mem - 1 \rightarrow Mem$, Skip next if $Mem = 0$
BIT INSTRUCTIONS		
SBIT	Set bit	1 $\rightarrow Mem.bit$
RBIT	Reset bit	0 $\rightarrow Mem.bit$
IFBIT	If bit	If $Mem.bit$ is = 1, do next instruction
IFNBIT	If not bit	If $Mem.bit$ is = 0, do next instruction

Notes: W is 16-bit word of memory

MA is Accumulator A or direct memory (8- or 16-bit)

Mem is 8-bit byte or 16-bit word of memory

Mem1 is 8- or 16-bit memory or 8- or 16-bit immediate data

imm is 8-bit or 16-bit immediate data

imm8 is 8-bit immediate data only

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
MEMORY TRANSFER INSTRUCTIONS		
LD	Load Load, incr/decr X	Mem1 → MA Mem(X) → A, X ± 1 (or 2) → X
LD B, mode	Load B	Mem1 → B
LD X, mode	Load X	Mem1 → X
LD K, mode	Load K	Mem1 → K
ST	Store to Memory	A → Mem
X	Exchange	A → Mem
	Exchange, incr/decr X	A → Mem(X), X ± 1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP
FETCH	Dummy read	Gen. addressed byte is read and discarded
POP	Pop Stack to Memory	SP - 2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ± 1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	MEM(B) ← → A, B ± 1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B	Mem1 → B
LD K	Load B	Mem1 → K
LD X	Load X	Mem1 → X
LD BK	Load B and K immediate	imm → B, imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A + 1 → A
COMP A	Complement A	1's complement A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ← → A3:0
RRC A	Rotate A right thru C	C → A15 → .. → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← .. ← A0 ← C
SHR A	Shift A right	0 → A15 → .. → A0 → C
SHL A	Shift A left	C ← A15 ← ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump Subroutine from table	PC → W(SP), SP + 2 → SP, W(table #) → PC
JSR	Jump Subroutine relative	PC → W(SP), SP + 2 → SP, PC + # → PC (# is + 1025 to - 1023)
JSRL	Jump Subroutine long	PC → W(SP), SP + 2 → SP, PC + # → PC
JP	Jump relative short	PC + # → PC (# is + 32 to -31)
JMP	Jump relative	PC + # → PC (# is + 257 to -255)
JMPL	Jump relative long	PC + # → PC
JID	Jump indirect at PC + A	PC + A + 1 → PC
JIDW		then Mem(PC) + PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP - 2 → SP, W(SP) → PC
RETSK	Return then skip next	SP - 2 → SP, W(SP) → PC, & skip
RETI	Return from interrupt	SP - 2 → SP, W(SP) → PC, interrupt re-enabled

Notes: W is 16-bit word of memory

MA is Accumulator A or direct memory (8- or 16-bit)

Mem is 8-bit byte or 16-bit word of memory

Mem1 is 8- or 16-bit memory or 8- or 16-bit immediate data

imm is 8-bit or 16-bit immediate data

imm8 is 8-bit immediate data only



Section 5 HPC Applications



Section 5 Contents

AN-474 HPC MICROWIRE/PLUS Master-Slave Handshaking Protocol	5-3
AN-484 Interfacing Analog Audio Bandwidth Signals to the HPC	5-11
AN-485 Digital Filtering Using the HPC	5-21
AN-486 A Floating Point Package for the HPC	5-36
AN-487 A Radix 2 FFT Program for the HPC	5-89
AN-497 Expanding the HPC Address Space	5-114
AN-510 Assembly Language Programming for the HPC	5-125
AN-550 A Software Driver for the HPC Universal Peripheral Interface Port	5-130
AN-551 The HPC as a Front-End Processor	5-185
AN-552 Interfacing a Serial EEPROM to the National HPC16083	5-249
AN-561 I ² C-Bus—Interface with HPC	5-266
AN-577 Extended Memory Support for HPC	5-286
AN-585 High Performance Controller in Information Control Applications	5-330
AN-586 Pulse Width Modulation Using HPC	5-338
AN-587 C in Embedded Systems and the Microcontroller World	5-346
AN-593 HPC16400 A Communication Microcontroller with HDLC Support	5-352
AN-603 Signed Integer Arithmetic on the HPC	5-362
AN-643 EMI/RFI Board Design	5-374
AN-736 Interfacing the HPC46064 to the DP83200 FDDI Chip Set	5-391
AN-786 LCD Direct Drive Using HPC	5-397
AN-798 Improved UART Clocking Techniques on New Generation HPCs	5-413

HPC MICROWIRE/PLUS™ Master-Slave Handshaking Protocol

National Semiconductor
Application Note 474
Richard Lazovick



INTRODUCTION

This applications note describes how to use National Semiconductor's MICROWIRE/PLUS to communicate between two members of the HPC family of microcontrollers, and will discuss the implications of adding other MICROWIRE™ peripherals. MICROWIRE/PLUS (μWIRE) may be effectively used to communicate between chips, such as in Small Area Networks (SANs). Possible applications range from setting up a communications network within an automobile to home security systems. Among the standard MICROWIRE peripherals available are display drivers (LCD, VF, LED), memories (RAM, EEPROM), A/D converters, and frequency generators/timers. Each MICROWIRE peripheral requires its own handshaking protocol, however the HPC's MICROWIRE is flexible enough to work with any peripheral and allows you to define your own handshaking protocol when having two HPC family members communicate.

MICROWIRE

MICROWIRE/PLUS is an extension of National Semiconductor's MICROWIRE communications interface. It allows

high speed two way serial communications between a master processor and one or more slave processors or peripherals. MICROWIRE/PLUS uses only three wires plus chip selects, therefore it saves on intricate bus routing and does not waste 8-bit ports. *Figure 1* shows the block diagram of a sample application using two HPC family members and an 8-bit A/D peripheral to monitor and control certain environmental conditions within a system.

MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register (SIO) using SI as the serial input and SO as the serial output. The contents of the SIO register may be accessed through any of the memory access instructions. SK is the clock for the SIO register (see *Figure 2*). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. Data to be transmitted from the SIO register is shifted out on the falling edge of the SK clock. Serial data on the SI pin is latched in on the rising edge of the SK clock (see *Figure 3 μWIRE Timing*).

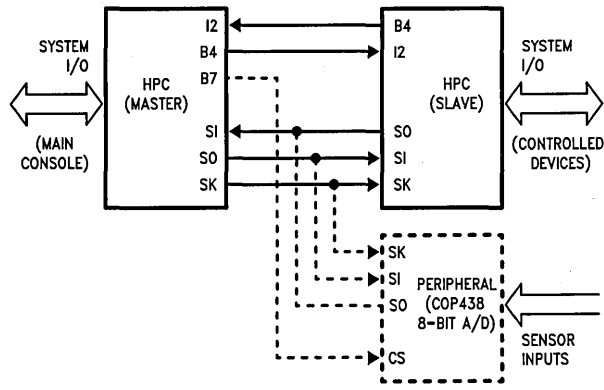
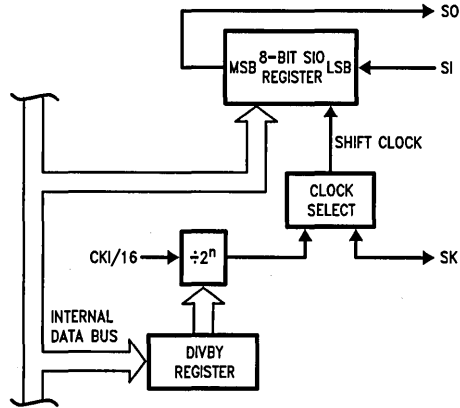


FIGURE 1. HPC μWIRE Block Diagram
(Environmental Control System)

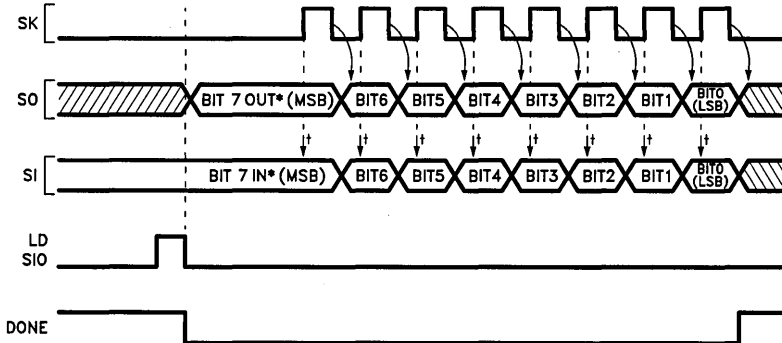
TL/DD/9140-1



TL/DD/9140-2

Note: The most significant bit is shifted out first. The SO pin reflects the contents of the MSB in the SIO register.

FIGURE 2. MICROWIRE/PLUS Block Diagram



TL/DD/9140-3

Note: The first bit of every eight bits in the SIO register being shifted out will have a longer duration than the other bits. This results from the hardware implementation used for MICROWIRE.

* This bit becomes valid immediately when the transmitting device loads its SIO register.

† Arrows indicate points at which SI is sampled.

FIGURE 3. μ WIRE Timing

A μ WDONE flag in the IRPD (Interrupt Pending) register indicates when the data shift is completed.

The HPC can enter the MICROWIRE/PLUS mode as a master or a slave. The μ WMODE control bit in the IRCD (Interrupt Condition) register determines whether the HPC is a master or slave. The shift clock is generated internally when the HPC is configured as a master. An externally generated shift clock on the SK pin is used when the HPC is configured as a slave. When the HPC is a master, the DIVBY register allows the SK clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz when CKI is 16 MHz (see Table I).

HOW TO USE MICROWIRE/PLUS

To use MICROWIRE, start by setting up the B port appropriately for the MICROWIRE functions. The SO and SK functions are multiplexed onto Port B pins B5 and B6 respectively. For the master, set bits 5 and 6 in the DIRB register (direction register for Port B) to set SO and SK as outputs. For the slave, set bit 5 and reset bit 6 in the DIRB register to set SO as an output and SK as an input. The BFUN register (Port B function register) is used to set SO and SK as alternate functions in the master and only SO as an alternate function in the slave. The MICROWIRE/PLUS mode can be enabled or disabled any time under program control. This is done through the BFUN register. Placing a "1" in the corresponding bit location causes the alternate function to be activated, a "0" causes the alternate function to be disabled. It is good practice to initialize the output pins by setting PORTB (Port B data register) to a known state.

The SI function is multiplexed onto Port I pin I5. This pin is always an input and the SI function is automatically selected when in the MICROWIRE mode. Setting the μ WMODE control bit, bit 1, in the IRCD register will enable the part to be a

master, resetting the bit will make it a slave. For the master, the DIVBY register has to be initialized to set the appropriate SK frequency (see Table I.). For example if the crystal frequency is 16 MHz and an SK frequency of 1 MHz is desired, load the least significant nibble of the DIVBY register with 2 (16 MHz/16 = 1 MHz).

For a summary of the register and pin configurations for the master and slave modes see Table II.

TABLE I. HPC μ WIRE DIVBY Register

μ WIRE SK Divisor				
MSB			LSB	CLOCK
0	0	0	0	not allowed
0	0	0	1	not recommended*
0	0	1	0	CKI/16
0	0	1	1	CKI/32
0	1	0	0	CKI/64
0	1	0	1	CKI/128
0	1	1	0	CKI/256
0	1	1	1	CKI/512
1	0	0	0	CKI/1024
1	0	0	1	CKI/2048
1	0	1	0	CKI/4096
1	0	1	1	CKI/8192
1	1	0	0	CKI/16384
1	1	0	1	CKI/32768
1	1	1	0	CKI/65536
1	1	1	1	CKI/131072

*This option uses timer T3 output, but does not generate a square wave. (See HPC users manual for more details.)

TABLE II. μ WIRE Register and Pin Conditions for Master and Slave Operation

Operation	μ WMODE bit	BFUN B5	BFUN B6	DIRB B5	DIRB B6	PIN B5	PIN B6	PIN I5
MICROWIRE Master	1	1	1	1	1	SO	INT. SK	SI
MICROWIRE Master	1	1	1	0	1	TRI-STATE®	INT. SK	SI
MICROWIRE Slave	0	1	0	1	0	SO	EXT. SK	SI
MICROWIRE Slave	0	1	0	0	0	TRI-STATE	EXT. SK	SI

DEFINING THE MASTER/SLAVE HANDSHAKING PROTOCOL

There are a few things to keep in mind when defining a handshaking protocol for the HPC:

- 1) Only the master can generate SK clocks.
- 2) As 8 bits are shifted into the SIO register, the 8 bits already in there are shifted out.
- 3) After 8 bits are shifted into (or out of) the SIO register the MICROWIRE done (μ WIRE DONE) flag gets set.
- 4) ANY access to the SIO register in the master that performs a write operation causes the contents of SIO to be shifted out.
- 5) No data will be shifted into or out of the slave's SIO register if its μ WIRE DONE flag is set.
- 6) Any write to the SIO register in the master or slave resets its μ WIRE DONE flag.

Keeping the above six points in mind, let's look at one possible handshaking protocol between a master HPC and a slave HPC. Number two above tells us we can send and receive data at the same time, however since only the master initiates data transfer we want to be sure the slave is ready before we get started with the exchange. Since the master initiates the transfer process there is no need for the master's MICROWIRE routine to be interrupt driven (though it can be if it is desired to have the slave initiate data transfers also). On the other hand, since the slave will be off doing other tasks it is most effective to have its MICROWIRE routine be interrupt driven.

A FEW THINGS TO NOTE ABOUT THE PROGRAMS

The following programs refer to the system configuration shown in *Figure 1*. This example code does a simple data transfer. The master reads in data on Port D, sends it via MICROWIRE to the slave, and reads it back. They both start by initializing the chip mode and number of wait states

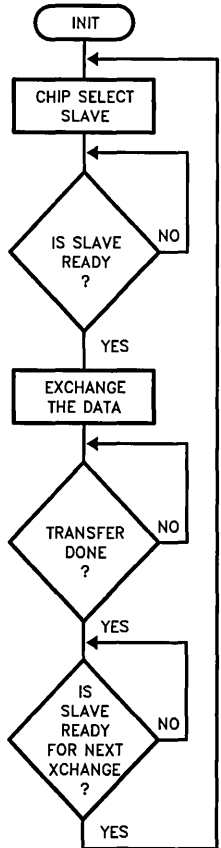
(PSW), disabling interrupts, setting the DIVBY register as necessary, initializing Port B, and enabling the appropriate MICROWIRE mode (IRCD). Then the slave continues with its main code (a wait loop) until interrupted. When the master decides it's ready to send MICROWIRE data, it signals the slave by setting the slave interrupt pin on Port B, then it waits for the slave to respond.

Meanwhile, the slave goes into action. It clears the μ WDONE flag and loads the SIO register (X A, SIO), then notifies the master that it is ready to continue. Once the master realizes the slave is ready to continue, it removes the interrupt signal to the slave (RESET PORTB.SLAVI), reads in the data to be sent (LD A, PORTD), and starts transmitting it (X A, SIO). At the same time the master reads in the data received at the last data exchange with the slave. Then the master loops until it is done transferring data and loops again until the slave is finished with its interrupt routine. In a real program the master would be off executing code and not having to wait in these loops. Once the transmission is complete the slave reads in the new data (LD A, SIO), lets the master know it is done with its interrupt routine (RESET PORTB.MASTR), and re-enables interrupts as it returns to the main routine (RETI).

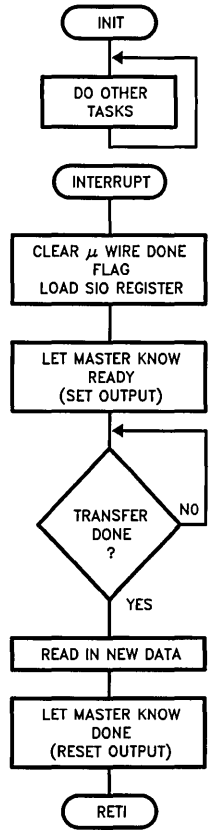
In the master's code there is only one access to the SIO register and that access is an exchange. Remember point #4, we can take advantage of the exchange instruction (X A, SIO), which is a read-modify-write instruction. Therefore, with one instruction, we can read the data from the previous transfer into the accumulator, and write the data to be transferred into the SIO register. If this method is not practical, then separate read and write instructions must be used.

When accessing the SIO register be sure the μ WIRE DONE flag is set so you know the data is not changing. At other times we have to be sure the flag is reset or no data will ever be transferred (shifted in or out). Notice that the "X A, SIO" was used to reset the μ WIRE DONE flag as well as load the register with the data to be sent.

MASTER'S Flow Chart



SLAVE'S Flow Chart



TL/DD/9140-4

TL/DD/9140-5

MASTER's SAMPLE CODE

```

;
;VARIABLE DECLARE
;
PSW   = M(00C0)
BFUN  = W(0F4)           ;Port B ALTERNATE FUNCTION REGISTER
DIRB  = W(0F2)           ;Port B DIRECTION REGISTER
PORTB = W(0E2)           ;Port B DATA REGISTER
PORTD = M(0104)          ;Port D (INPUT PORT)

ENIR  = M(0D0)           ;INTERRUPT ENABLE REGISTER
IRPD  = M(0D2)           ;INTERRUPT PENDING REGISTER
IRCD  = M(0D4)           ;INTERRUPT CONDITION REGISTER
SIO   = M(0D6)           ;SERIAL I/O REGISTER
PORTI = M(0D8)           ;INTERRUPT (AND uWIRE SERIAL IN) INPUT PORT
DIVBY = W(018E)          ;TIMER DIVIDE BY REGISTER

SLAVI = 4                 ;SLAVE INTERRUPT BIT (IN Port B)
uWDONE = 0                ;uWIRE DONE BIT (IN IRPD)
uWMODE = 1                ;uWIRE MASTER/SLAVE BIT (IN IRCD)
SK     = 6                 ;uWIRE SERIAL CLOCK (IN Port B)
SLAVR = 2                 ;SLAVE RESPONSE BIT (IN Port B)
  
```


MASTER's SAMPLE CODE (Continued)

```

.=OF800                ;START PROGRAM

BEGIN:
  LD   PSW,008          ;SINGLE CHIP MODE, 1 WAIT STATE
  LD   ENIR,00          ;DISABLE ALL INTERRUPTS
  LD   DIVBY,02222      ;uWIRE CLOCK = /16
  LD   DIRB,0FFFF       ;Port B ALL OUTPUTS
  LD   BFUN,00060       ;ONLY SO & SK HAVE ALTERNATE FUNCTIONS
  LD   PORTB,00000      ;INIT PORTB TO ALL ZEROS
  SET  IRCD,uWMODE      ;SET THIS HPC AS MASTER

DOITAG:                ;JUMP TO HERE TO DO IT AGAIN
  SET  PORTB.SLAVI      ;NOTIFY SLAVE (INTERRUPT THE SLAVE)

WAIT:
  IF   PORTI.SLAVR      ;SLAVE READY?
  JP   SLAVRS           ;GO SEND/RECEIVE uWIRE DATA
  JP   WAIT             ;NO IT IS NOT READY YET

SLAVRS:
  RESET PORTB.SLAVI     ;REMOVE SLAVE NOTIFIER
  LD   A,PORTD          ;LOAD A W/ DATA TO SEND
  X    A,SIO            ;SEND NEW DATA AND READ DATA FROM
                          ;...LAST uWIRE EXCHANGE

DONE:
  IF   IRPD.uWDONE      ;WAIT TILL DONE EXCHANGING
  JP   CONT             ;uWIRE IS DONE
  JP   DONE             ;uWIRE NOT DONE (KEEP TESTING)

CONT:
  IF   PORTI.SLAVR      ;IS SLAVE READY TO CONTINUE?
  JP   CONT             ;NO
  JP   DOITAG          ;START ALL OVER (DO IT AGAIN)

.END BEGIN

```

SLAVE's SAMPLE CODE

```

;
;VARIABLE DECLARE
;
PSW   = M(00C0)
BFUN  = W(0F4)          ;Port B ALTERNATE FUNCTION REGISTER
DIRB  = W(0F2)          ;Port B DIRECTION REGISTER
PORTB = W(0E2)          ;Port B DATA REGISTER

```

SLAVE's SAMPLE CODE (Continued)

```

ENIR = M(OD0)           ;INTERRUPT ENABLE REGISTER
IRPD = M(OD2)           ;INTERRUPT PENDING REGISTER
IRCD = M(OD4)           ;INTERRUPT CONDITION REGISTER
SIO  = M(OD6)           ;SERIAL I/O REGISTER
SO   = 5                 ;uWIRE SERIAL OUTPUT PIN (ON Port B)
MASTR = 4                ;MASTER RESPONSE BIT (IN Port B)
uWDONE = 0               ;uWIRE DONE BIT (IN IRPD)
uWMODE = 1               ;uWIRE MASTER/SLAVE BIT (IN IRCD)
INT2  = 2                ;INTERRUPT 2 BIT

.=OFFFA                 ;INT2 - INTERRUPT VECTOR
.WORD MASNOT            ;...MASTER NOTIFICATION
.=OF800                 ;START PROGRAM

BEGIN:
    LD    PSW,008        ;SINGLE CHIP MODE, 1 WAIT STATE
    LD    ENIR,01        ;DISABLE ALL INTERRUPTS, BUT ENABLE GIE

    LD    DIRB,OFF10     ;Port B UPPER, & MASTR ARE OUTPUTS
                        ;...(use LD DIRB,OFF30 to set SO as an
                        ;...output if not using any peripherals)

    LD    BFUN,00020     ;ONLY SO HAS ALTERNATE FUNCTION
                        ;...NOTE: SK is NOT an alternate
                        ;...function in the slave!

    LD    PORT B,00000   ;INIT PORTB TO ALL ZEROS

    RESET IRCD.uWMODE    ;SET THIS HPC AS A SLAVE
    SET   IRCD.INT2      ;SET INT2 INTERRUPT (+) POLARITY
    SET   ENIR.INT2      ;ENABLE EXTERNAL INTERRUPT TO
                        ;...RECEIVE SLAVE RESPONSE

PAU:
    JP    PAU            ;WAIT HERE FOR INTERRUPT FROM MASTER

MASNOT:
                        ;uWIRE INTERRUPT ROUTINE
    X    A,SIO           ;CLEAR uWDONE FLAG (AND LOAD DATA FROM
                        ;...ACCUMULATOR TO SEND)
    SET   PORTB.SO       ;ENABLE SO (needed only if using a peripheral)
    SET   PORTB.MASTR    ;NOTIFY MASTER THAT READY TO CONTINUE

NOTDN:
    IF    IRPD.uWDONE    ;WAIT TILL DONE SHIFTING
    JP    DONE           ;DONE, GO CONTINUE
    JP    NOTDN          ;NOT DONE, CONTINUE LOOPING

DONE:
    LD    A,SIO          ;READ IN NEW DATA
    RESET PORT B.SO      ;TRI-STATE SO (needed only if
                        ; using a peripheral)
    RESET PORTB.MASTR    ;REMOVE SIGNAL TO MASTER
    RETI

.END BEGIN

```

ADDING PERIPHERALS OR ANOTHER SLAVE

Adding another slave HPC or a peripheral to the above Microwire configuration can add more power to your design with minimal extra cost and design time. In *Figure 1*, an extra peripheral is shown in dotted outline form. The hardware and software modifications are straightforward, however there are a few considerations to keep in mind:

- Tri-state the SO pin on the slave HPC by resetting B5 in the DIRB register when the slave is not 'chip-selected' by the master.
- When adding more HPC slaves, the master's and slave's routines remain the same. Only different B port pins for chip select and I or B port pins for slave acknowledgement need to be used.
- For peripherals the principals of operation are still the same and so are the initialization procedures, however some of the code will have to be modified to accommodate the specific handshaking required by the peripheral. (Note: some of the peripherals require 16 or more consecutive bits without interruption of the SK clock. To provide continuous SK clocks, set up the accumulator with next byte of data to send, loop until μ WDONE is set, then exchange the contents of the accumulator and the SIO register (X A, SIO). The above steps will provide nearly continuous SK clocks—the slower the SK clock is set for, the more continuous they will appear.)

APPLICATIONS

Now that you are more familiar with MICROWIRE/PLUS, where can you get experience using it?

- It can be used in a security system where the on-site master lets the periphery slaves know which security codes they can now let in, while at the same time the slaves monitor fire alarms and smoke detectors.
- It can be used in automotive brakes to allow all the wheels to communicate with each other. The wheels can trade information on road conditions and a master can monitor all four wheels to coordinate them and check for malfunctions.
- It can be used in a robot arm to allow each joint to make the decision as to how it will help the entire arm reach its final position. This application is one example of how MICROWIRE/PLUS can be used for system task partitioning.
- It can be used in a MUX-WIRING system.

When using MICROWIRE to communicate between two chips on the same board, a high data rate can be used. When communicating over longer distances, slower speeds should be used.

SUMMARY

MICROWIRE/PLUS can be a very powerful tool that can easily add power to a microcontroller based system. It is easy to use and does not require much hardware to implement. To add a new feature to your current design, choose a peripheral and add a small amount of code. To start using MICROWIRE, define the handshake protocol best suited for your application keeping in mind the six points given above in the 'Defining the Master/Slave Handshaking Protocol' section. Then initialize the appropriate registers: BFUN, DIRB, PORTB, DIVBY, and IRCD. The MICROWIRE circuitry will then run independent of the CPU except to exchange data between the SIO register and the CPU, and to initiate the data exchange between the master and slaves. With a CPU clock of 16 MHz, MICROWIRE/PLUS may achieve a maximum data rate of 1 MHz. MICROWIRE can be used to add display controllers, A/D's, memories, timers, and even other microcontrollers to an HPC microcontroller based design. Remember MICROWIRE/PLUS is not a trivial piece of very fine wire, it is a high speed two way serial communications interface!

Interfacing Analog Audio Bandwidth Signals to the HPC

National Semiconductor
Application Note 484
Ashok Krishnamurthy



INTRODUCTION

This report describes a method of interfacing analog audio bandwidth signals to the National Semiconductor HPC microcontroller. The analog signal is converted to a digital value using the National Semiconductor TP3054 codec/filter combo. The digital value is then transferred to the HPC using the MICROWIRE/PLUSTM synchronous serial interface. The digital output sample computed by the HPC is also transferred to the TP3054 using the MICROWIRE/PLUS interface. The TP3054 then converts this digital value to an analog signal.

ADVANTAGES OF USING A CODEC

There are a number of advantages in using a codec for A/D and D/A conversion of analog signals.

1. The codec/filter combos such as the TP3054 integrate a number of functions on a single chip. Thus the TP3054 includes the analog anti-aliasing filters, the Sample-and-Hold circuitry and the A/D and D/A converters for analog signal interfacing.
2. The μ -law coding effectively codes a 14-bit conversion accuracy in 8 bits. This allows the interface to the HPC to be greatly simplified.

DISADVANTAGES IN USING A CODEC

While the use of a codec is appropriate for audio (in particular speech) processing applications, it has a number of disadvantages in other cases.

1. The sampling rate is fixed at 8 kHz. If lower or higher sampling rates are desired, the codec cannot be used. Note that the real-time signal processing that the HPC can perform at a 8 kHz sampling rate is limited.
2. The resolution is fixed, and is about 14 bits/sample.
3. Digital filtering algorithms require that the samples used in the processing be linear coded PCM. Thus the 8-bit μ -law PCM values output by the codec need to be converted to linear coded PCM. Correspondingly, the output of the digital filter, which is in linear coded PCM needs to be converted to 8-bit μ -law PCM before outputting to the codec. This requires additional processing per sample.

DESCRIPTION OF THE INTERFACE

The circuit schematic of the interface is shown in *Figure 1*. Note that the schematic does not show complete details of the HPC. Only the HPC pins that are relevant to this interface are shown. A wire-wrapped version of the circuit has been constructed on a NSC HPC 16040 Chip Carrier Board.

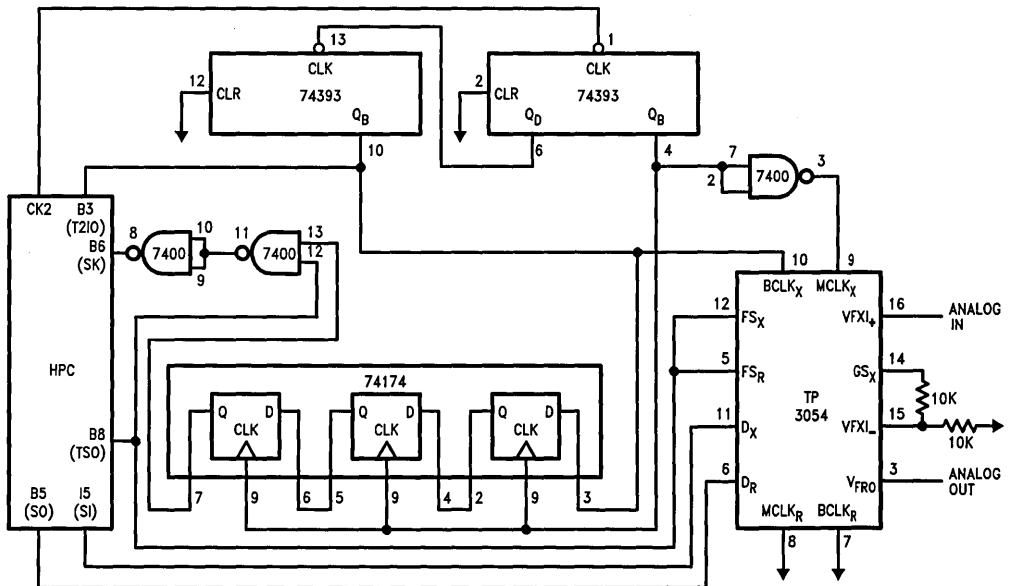


FIGURE 1. Circuit Schematic

TL/DD/9246-1

Note that this report does not go into the details about the use of the TP3054 codec chip or programming the HPC. It also does not discuss the μ -law to linear and linear to μ -law code conversion in detail. For more information on these issues, please consult the references listed at the end.

1. **Codec Signalling Considerations.** The TP3054 can operate in either synchronous or asynchronous modes. Further, in each of these modes, it uses short or long frame sync operation. The circuit shown in *Figure 1* runs the codec in synchronous mode with long-frame-sync operation.

The codec requires 4 clock sources for proper operation in the synchronous mode. These are MCLK-x, BCLK-x, FS-x and FS-r. MCLK-x is a master clock and is used to clock the switched-capacitor filters. BCLK-x is the bit shift clock, and FS-x and FS-r are the frame sync clocks. These clocks need to be synchronous.

These clocks are obtained in the circuit as follows. MCLK-x is obtained by dividing the HPC CK2 clock output by 4. If the HPC is using a 16 MHz crystal, this results in MCLK-x being 2 MHz.

BCLK-x is obtained by dividing CK2 by 64. This gives an effective value for BCLK-x of 125 kHz. Note that MCLK-x is inverted before being fed to the codec. This is done to synchronize MCLK-x and BCLK-x on their leading edges.

FS-x and FS-r are the same clocks in the circuit. They are obtained by dividing BCLK-x by 16 using the timer T2 on the HPC. BCLK-x is used as the external clock input on pin T2IO of the HPC and FS-x (FS-r) is obtained from the timer synchronous output TS0. Note that the delay inherent in the HPC between the underflow of a timer and the toggling of the corresponding output allows FS-x and BCLK-x to be leading edge synchronized (more accurately, the delay is within the codec's acceptable limits.) Note that in order to accomplish these functions, the HPC pins need to be properly configured. This is not described here. Please refer to the appropriate HPC documentation and consult the sample program included with this report.

2. **MICROWIRE/PLUS Interface Considerations.** MICRO-WIRE/PLUS is a National Semiconductor defined 8-bit serial synchronous communication interface. It is designed to allow easy interfacing of NSC microcontrollers and peripheral chips. The HPC microcontroller has a MICROWIRE/PLUS interface; however the TP3054 codec does not. Thus some external "glue logic" is necessary to allow the HPC and the TP3054 to be interfaced.

The HPC MICROWIRE/PLUS interface is operated in Slave mode for this application. This means that the shift clock needed to latch-in/shift-out data from the Micro-wire SIO register is provided externally on the SK pin. Micro-wire latches in data on the leading edge of the SK clock and shifts out data on the trailing edge of SK. Also SK needs to be a burst clock for proper operation.

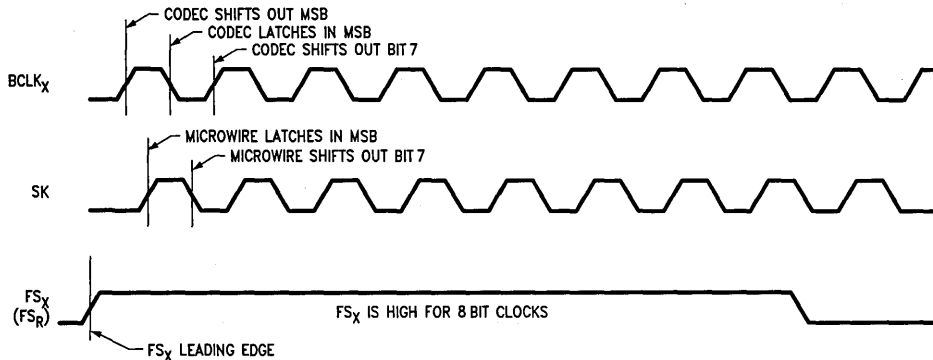


FIGURE 2. Timing Waveforms

TL/DD/9246-2

The codec shifts out data on the D-x pin on the first 8 leading edges of BCLK-x after a FS-x leading edge. Also, it latches in data on the D-r pin on the first 8 trailing edges of BCLK-x after a FS-r leading edge. Note that FS-x and FS-r are the same in this application. Refer to the timing diagram in *Figure 2*.

Thus, it is seen that there is a timing difference in the way the codec and the Micro-wire interfaces work. However, as seen in *Figure 2*, if the shift clock, SK, to the Microwire interface is delayed with respect to BCLK-x, the two interfaces should work compatibly. This delay is accomplished by clocking BCLK-x through a shift register using MCLK-x as the clock source. This can be seen in the circuit schematic in *Figure 1*. (The author thanks Mr. Richard Lazovick for this suggestion.)

μ -LAW TO LINEAR/LINEAR TO μ -LAW CONVERSION

It was explained earlier that the codec outputs digital values that are expanded using the MU-255 PCM standard. However, for linear digital filtering applications, the input needs to be in linear PCM format. Similarly, it is necessary to provide the conversion from linear PCM to MU-255 PCM before output to the codec. The HPC accomplishes this in software.

1. μ -law to linear conversion. The codec output is actually the complement of the μ -law value. Thus, this first needs to be complemented to obtain the true μ -law value. The simplest way to obtain the corresponding linear value is through table look-up. The output of the table is the 16-bit 2's complement linear value. The sample program included with this report utilizes this technique. A macro that constructs this table is also provided.
2. Linear to μ -law conversion. An algorithm to convert a 13-bit positive linear number to 7-bit μ -law is described in *Figure 3*. The algorithm is based on the description in the book by Bellamy listed in the reference. The most significant 8th bit for the μ -law code is obtained from the sign of the input linear code.
1. Get 13-bit positive input value.
2. Add to it the bias value of 31-decimal.
3. The compressed μ -law word is then obtained as follows:

Biased Linear Value												
Bits												
12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	Q3	Q2	Q1	Q0	a
0	0	0	0	0	0	1	Q3	Q2	Q1	Q0	a	b
0	0	0	0	0	1	Q3	Q2	Q1	Q0	a	b	c
0	0	0	0	1	Q3	Q2	Q1	Q0	a	b	c	d
0	0	1	Q3	Q2	Q1	Q0	a	b	c	d	e	f
0	1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g
1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g	h

μ -Law Value						
Bits						
6	5	4	3	2	1	0
0	0	0	Q3	Q2	Q1	Q0
0	0	1	Q3	Q2	Q1	Q0
0	1	0	Q3	Q2	Q1	Q0
0	1	1	Q3	Q2	Q1	Q0
1	0	0	Q3	Q2	Q1	Q0
1	0	1	Q3	Q2	Q1	Q0
1	1	0	Q3	Q2	Q1	Q0
1	1	1	Q3	Q2	Q1	Q0

FIGURE 3. 13-Bit Linear to 8-Bit μ -Law Conversion

POSSIBLE APPLICATIONS

The codec/HPC interface described above can be used in a number of speech processing applications. One application, ADPCM coding of speech, is presently under development. Other applications include: a voiced/unvoiced/silence classifier, a voice pitch tracker, speech detection circuitry etc. Note that the main limitation here (at least for real-time applications) is the amount of effective computation that can be done by the HPC between samples.

REFERENCES

1. National Semiconductor Corporation, *Telecommunications Databook*, Santa Clara, California, 1984.
2. National Semiconductor Corporation, *HPC Programmers Reference Manual*, Santa Clara, California, 1986.
3. National Semiconductor Corporation, *HPC Hardware Reference Manual*, Santa Clara, California, 1986.
4. J. C. Bellamy, *Digital Telephony*, John Wiley & Sons, New York, 1982.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communication package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

PROGRAM TO TEST CODEC INTERFACE

NATIONAL SEMICONDUCTOR CORPORATION Page: 1
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 TSTCDC

```

1      ;
2      ;
3      ;           .TITLE TSTCDC
4      ;
5 01C0      YOFK = M(01C0)           ; OUTPUT SAMPLE STORAGE.
6 00C0      PSW = M(00C0)
7 00D0      ENIR = M(00D0)
8 00D2      IRPD = M(00D2)
9 00D4      IRCD = M(00D4)
10 00D6     SIO = M(00D6)
11 00D8     PORTI = M(00D8)
12 00E2     PORTBL = M(00E2)
13 00E3     PORTBH = M(00E3)
14 00E2     PORTE = W(00E2)
15 00F2     DIRBL = M(00F2)
16 00F3     DIRBH = M(00F3)
17 00F2     DIRB = W(00F2)
18 00F4     BFUNL = M(00F4)
19 00F5     BFUNH = M(00F5)
20 00F4     BFUN = W(00F4)
21 0188     T2TIM = W(0188)
22 0186     T2REG = W(0186)
23 018E     DIVBYL = M(018E)
24 018F     DIVBYH = M(018F)
25 018E     DIVBY = W(018E)
26 0190     TMMDL = M(0190)
27 0191     TMMDH = M(0191)
28 0190     TMMD = W(0190)
29      ;
30      ;
31      ;
32      ;           .MACRO MUTBL,STADR
33      ;
34      ; MACRO TO CREATE LOOKUP TABLE FOR MU-255 LAW PCM TO LINEAR CONVERSION.
35      ; STADR IS THE STARTING ADDRESS FOR THE TABLE, AND MUST BE AN EVEN ADDRESS.
36      ; THE TABLE OCCUPIES 512 BYTES.
37      ;
38      ;           . = STADR
39      ;           .SET SVAL,021
40      ;           .SET INCRM,02
41      ;           .DO 08
42      ;               .SET MVAL,SVAL-021
43      ;               .DO 010
44      ;                   .WORD MVAL
45      ;                   .SET MVAL,MVAL+INCRM
46      ;           .ENDDO
47      ;           .SET SVAL,SVAL*02
48      ;           .SET INCRM,INCRM*02
49      ;           .ENDDO
50      ;
51      ;           .SET SVAL, 021

```

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 2
 HPC CROSS ASSEMBLER, REV:C,30 JUL 86
 TSTCDC

```

52          .SET INCRM, 02
53          .DO 08
54          .SET MVAL,SVAL-021
55          .DO 010
56          .SET RVAL,-1*MVAL
57          .WORD RVAL
58          .SET MVAL,MVAL+INCRM
59          .ENDDO
60          .SET SVAL,SVAL*02
61          .SET INCRM,INCRM*02
62          .ENDDO
63          ;
64          .ENDM
65          ;
66          ;
67          ;
68          .LOCAL
69 F000      MUTBL, OF000
70          ;
71 F200      .= OF200
72          CODEC:
73 F200 B701FOC4      LD SP, 01FO          ; INITIALIZE STACK POINTER.
74          ;
75 F204 3059          JSR INITCD          ; INITIALIZE THE CODEC
76          FLOOP:
77 F206 3005          JSR INPUT          ; GET INPUT SAMPLE, OUTPUT
78          ; PREVIOUS SAMPLE.
79 F208 E7            SHL A
80 F209 E7            SHL A
81 F20A 301F          JSR OUTPUT          ; CONVERT OUTPUT VALUE TO
82          ; MU-255 LAW AND SAVE.
83 F20C 66            JP FLOOP          ; 60 DO NEXT SAMPLE.
84          ;
85          ;
86          INPUT:
87 F20D B601C088      LD A, YOFK          ; GET DATA TO BE OUTPUT.
88          NOTDN:
89 F211 96D210          IF IRPD,0          ; IS MICROWIRE DONE?
90 F214 41            JP MWDONE          ; YES, SO GET DATA.
91 F215 64            JP NOTDN          ; NO, SO TRY AGAIN.
92          MWDONE:
93 F216 BED6          X A, SIO          ; GET NEW SAMPLE, OUTPUT
94          ; COMPUTED DATA.
95 F218 01            COMP A          ; TAKE CARE OF CODEC INVERSION.
96 F219 99FF          AND A, OFF
97 F21B E7            SHL A
98 F21C BAFO00          OR A,OF000          ; FORM MU-LAW TO LINEAR
99          ; TABLE ADDRESS.
100 F21F AECE          X A, X
101 F221 D0            LD A, M(X+)          ; GET LINEAR VALUE
102 F222 AECA          X A, K

```


NATIONAL SEMICONDUCTOR CORPORATION PAGE: 3
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 TSTCDC

```

103 F224 04          LD A, M(X)          ; A BYTE AT A TIME.
104 F225 BCC8CB      LD H(K), I(A)
105 F228 ABCA        LD A, K
106 F22A 3C          RET
107                  ;
108                  ;
109                  OUTPUT:
110 F22B 96D41F      RESET IRCD.7
111 F22E E7           SHL A          ; SIGN BIT TO C.
112 F22F 06           IFN C          ; IS IT POSITIVE?
113 F230 45           JP OPOS
114 F231 96D40F      SET IRCD.7
115 F234 01           COMP A
116 F235 04           INC A          ; NEGATIVE, SO TAKE 2'S
117                  ; COMPLEMENT.
118                  OPOS:
119 F236 B80108      ADD A, 0108    ; ADD BIAS.
120 F239 9107        LD K, 07       ; SET UP COUNTER.
121                  ALIGN:
122 F238 E7           SHL A
123 F23C 07           IF C
124 F23D 44           JP ODONE      ; FOUND MS 1 BIT.
125 F23E AACA        DECSZ K
126 F240 65           JP ALIGN
127 F241 E7           SHL A          ; HAS TO BE 1 IN C NOW.
128                  ODONE:
129 F242 AECA        X A, K
130 F244 E7           SHL A
131 F245 E7           SHL A
132 F246 E7           SHL A
133 F247 E7           SHL A      ; COUNTER VALUE IN BITS 4-6.
134 F248 AECC        X A, B
135 F24A 00           CLR A
136 F24B 88CB        LD A, H(K)
137 F24D 3B           SWAP A
138 F24E 990F        AND A, 0F
139 F250 96CCFA      OR A, B
140 F253 96D417      IF IRCD.7
141 F256 96C80F      SET A.7
142 F259 01           COMP A
143 F25A B601C08B    ST A, YOFK
144 F25E 3C          RET
145                  ;
146                  INITCD:
147 F25F B7FFB7F2    LD DIRB, OFFB7 ; SET B3 (T2IO) AND B6 (SK)
148                  ; ON PORT B AS INPUTS. SET ALL
149                  ; OTHER PINS ON B AS OUTPUT.
150 F263 B70000E2    LD PORTB, 0    ; OUTPUT 0 ON ALL PORT B PINS.
151 F267 96F40B      SET BFUNL.3    ; ALT. FUN. ON B3-T2IO.
152 F26A 96F40D      SET BFUNL.5    ; ALT. FUN. ON B5-S0.
153 F26D 96F508      SET BFUNH.0    ; ALT. FUN. ON B8-TSO.

```

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC

```
154 F270 9700D0          LD ENIR, 0          ; DISABLE INTRPTS.
155 F273 9700D4          LD IRCD,0          ; SELECT SLAVE MODE FOR M-WIRE.
156 F276 83070188AB      LD T2TIM, 07       ; LOAD 7-DEC INTO T2 TIMER.
157 F27B 83070186AB      LD T2REG, 07       ; LOAD 7-DEC INTO T2 REG.
158 F280 8300018F8B      LD DIVBYH, 0       ; SELECT EXT, CLOCK FOR T2 TIMER.
159                      ;
160 F285 8ED6             X A, SIO
161 F287 8740400190AB     LD TMMD,04040      ; START TIMER T2.
162 F28D 3C              RET
163                      ;
164                      ;
165 FFFE 00F2             .END CODEC
```

TSTCDC

SYMBOL TABLE

A	00C8 W	ALIGN	F23B	B	00CC W	BFUN	00F4 W*
BFUNH	00F5 M	BFUNL	00F4 M	CODEC	F200	DIRB	00F2 W
DIRBH	00F3 M*	DIRBL	00F2 M*	DIVBY	018E W*	DIVEYH	018F M
DIVBYL	018E M*	ENIR	00D0 M	FLOOP	F206	INCRM	0200
INITCD	F25F	INPUT	F20D	IRCD	00D4 M	IRPD	00D2 M
K	00CA W	MVAL	205F	MWDONE	F216	NOTDN	F211
ODONE	F242	OPOS	F236	OUTPUT	F22B	PC	00C6 W
PORTB	00E2 W	PORTBH	00E3 M*	PORTBL	00E2 M*	PORTI	00D8 M*
PSW	00C0 M*	RVAL	EDA1	SIO	00D6 M	SP	00C4 W
SVAL	2100	T2REG	D186 W	T2TIM	0188 W	TMDM	0190 W
TMDH	0191 M*	TMDL	0190 M*	X	00CE W	YOFK	01C0 M

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
MACRO TABLE

MUTBL

NO WARNING LINES

NO ERROR LINES

656 ROM BYTES USED

SOURCE CHECKSUM = 81D3

OBJECT CHECKSUM = 0C3C

INPUT FILE C:CODECTST.MAC

LISTING FILE C:CODECTST.PRN

OBJECT FILE C:CODECTST.LM

TSTCDC

SYMBOL TABLE

A	00C8 W	ALIGN	F23B	B	00CC W	BFUN	00F4 W*
BFUNH	00F4 M	BFUNL	00F4 M	CODEC	F200	DIRB	00F2 W
DIRBH	00F3 M*	DIRBL	00F2 M*	DIVBY	018E W*	DIVEYH	018F M
DIVEYL	01B3 M*	ENIR	00D0 M	FLOOP	F206	INCRM	0200
INITCD	F25F	INPUT	F20D	IRCD	00D4 M	IRPD	00D2 M
K	00CA W	MVAL	205F	MWDONE	F216	NOTDN	F211
ODONE	F242	OPOS	F236	OUTPUT	F22B	PC	00C6 W
PORTB	00E2 W	PORTBH	00E3 M*	PORTBL	00E2 M*	PORTI	00D8 M*
PSW	00C0 M*	RVAL	EDAL	SIO	00D6 M	SP	00C4 W
SVAL	2100	T2REG	D186 W	T2TIM	0188 W	TMMD	0190 W
TMDH	0191 M*	TMDL	0190 M*	X	00CE W	YOFK	01C0 M

Digital Filtering Using the HPC

National Semiconductor
Application Note 485
Ashok Krishnamurthy



INTRODUCTION

This report discusses the implementation of Infinite Impulse Response (IIR) digital filters using the National Semiconductor HPC microcontroller. A general program, that can be used to implement cascaded second order sections, up to a maximum of 8 sections, is also included. The program may have to be modified for specific A/D and D/A interfaces.

This report is not intended to be a tutorial on Digital Filter Design methods or their implementation details. Such information can be found in references 1 and 2 below. The general discussion included here closely follows that in reference 3.

DIGITAL FILTERING

The general IIR filter with input $x(n)$ and output $y(n)$ can be described by a transfer function of the form

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a(0) + a(1)z^{-1} + \dots + a(m)z^{-m}}{1 + b(1)z^{-1} + \dots + b(p)z^{-p}}$$

To minimize the effects of coefficient truncation, high order filters are usually implemented as a cascade of second order sections. (Another possible choice is parallel realization—see references below).

In cascade realizations, the numerator and denominator polynomials in the above are factored into second order terms, and the filter is realized as a cascade of such second order sections. This is shown in *Figure 1*. A typical second order section has a transfer function of the form

$$H(z) = \frac{A_0 + A_1 \times z^{-1} + A_2 \times z^{-2}}{1 + B_1 \times z^{-1} + B_2 \times z^{-2}}$$

A second order section such as the above can be realized in a number of ways; the one of concern here is the so-called 1-D form (see Reference 3). The second order 1-D form is shown in *Figure 2*. Based on this figure, we can obtain the following equations:

$$m(k) = x(k) - B_1 \times m(k-1) - B_2 \times m(k-2)$$

$$y(k) = A_0 \times m(k) + A_1 \times m(k-1) + A_2 \times m(k-2)$$

Define $T_1 = -B_1 \times m(k-1) - B_2 \times m(k-2)$,

$$T_2 = A_1 \times m(k-1) + A_2 \times m(k-2)$$

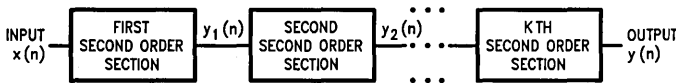
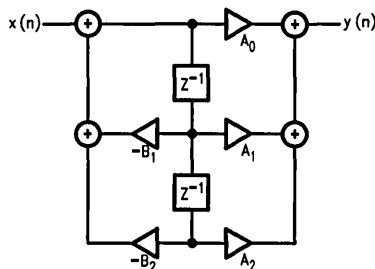


FIGURE 1. Cascade Realization of a Digital Filter



TL/DD/9247-2

FIGURE 2. One Second Order Section

Since T_1 and T_2 depend on signal values at time $k-1$ and $k-2$, we can precompute and store these quantities in the time interval from $k-1$ to k . Then, when $x(k)$ becomes available at time k , $y(k)$ and $m(k)$ can be quickly computed using

$$m(k) = x(k) + T_1,$$

$$y(k) = A_0 \times m(k) + T_2$$

If there are a number of stages, then these computations should be repeated for each stage. Based on these discussions, the operation of a digital filter can be described using the flowchart in *Figure 3*.

USING THE FILTER PROGRAM

Appendix A contains the listing of the program FILTER that can be used to implement cascaded IIR filters as described above. The program as shown uses a codec interfaced to the HPC using MICROWIRE/PLUSTM to do the A/D and D/A conversion. The program can be used with other A/D and D/A converters by suitably modifying the following sub-routines: INPUT, OUTPUT and INIT. Only the portions of INIT that deal with the codec interface need to be modified.

TL/DD/9247-1

The filter coefficients and the number of cascaded stages need to be supplied to the program. This is done as follows:

1. *Specification of filter order.* Define a word address called ROMNST and store the number of cascaded stages in that word. The program is presently set up for 4 cascaded stages.

2. *Specification of filter coefficients.* Each second order stage needs the specification of 5 coefficients, A0, A1, A2, B1 and B2. If the number of stages is m, let the coefficients be

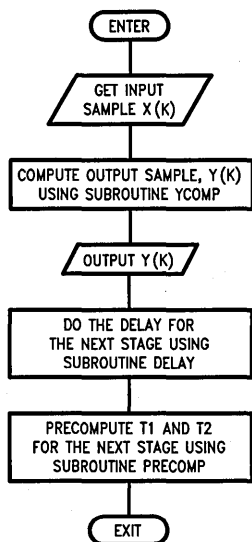
A0-1, A1-1, A2-1, B1-1, B2-1 for stage 1,

A0-2, A1-2, A2-2, B1-2, B2-2 for stage 2,

.

.

A0-m, A1-m, A2-m, B1-m, B2-m for stage m.



TL/DD/9247-3

FIGURE 3. Flowchart for the Computations in a Second Order Module (Based on Reference 3)

Define 5 word addresses called ROMA0, ROMA1, ROMA2, ROMB1, ROMB2 and store these coefficients at these addresses as follows:

ROMA0: WORD A0-1, A0-2, A0-3, ... A0-m

ROMA1: WORD A1-1, A1-2, A1-3, ... A1-m

ROMA2: WORD A2-1, A2-2, A2-3, ... A2-m

ROMB1: WORD B1-1, B1-2, B1-3, ... B1-m

ROMB2: WORD B2-1, B2-2, B2-3, ... B2-m.

Note that the coefficients are signed and need to be in 2's complement representation. Also, the stored coefficients need to be half their actual value. This is because of the way that the program does 2's complement multiplication using the subroutine SMULT.

The FILTER program copies all the coefficients to on-chip RAM for faster execution. Also temporary storage for $m(k)$, $m(k-1)$, $m(k-2)$, T1 and T2 is obtained from on-chip RAM. This, along with the storage of various addresses used by the program consumes the entire 192 bytes of user base page RAM.

Note that the filter program does not check for overflow during the various additions. This is because the HPC does not have a signed addition/subtraction overflow flag, and it was felt that the simulation of this feature in software would add excessive overhead. It is therefore the user's responsibility to ensure that the filter coefficients are properly scaled so that the overflow will not occur.

16 x 16 2's COMPLEMENT MULTIPLICATION

One of the basic operations in digital filtering is that of signed multiplication. Since the HPC supports unsigned multiplication only, a method to perform 2's complement multiply using the unsigned multiply is needed.

Let A and B be 2's complement 16 bit integers. Consider the following cases.

1. $A \geq 0, B \geq 0$. In this case the unsigned multiply result is $A \times B$, which is also the 2's complement multiply result. Thus no further processing is needed.
2. $A \geq 0, B < 0$. In this case the unsigned multiply result is $(2^{16}) \times A - A \times |B|$. However the desired result is $(2^{32}) - A \times |B|$. Thus we need to add $(2^{32}) - (2^{16}) \times A$ to the unsigned multiply result to obtain the correct value.
3. $A < 0, B \geq 0$. This case is similar to the previous one. $(2^{32}) - (2^{16}) \times B$ should be added to the unsigned multiply result to get the correct answer.
4. $A < 0, B < 0$. The unsigned multiply result in this case is $(2^{32}) - (2^{16}) \times (|A| + |B|) + |A| \times |B|$. The desired result in this case is $|A| \times |B|$. To get the correct answer, add $(2^{16}) \times (|A| + |B|)$ to the unsigned multiply result.

Based on the above discussion, an algorithm for 2's complement multiplication, where the result is a 32 bit 2's complement integer is shown in Figure 4.

1. Let A and B be the two 2's complement integers to be multiplied.
2. Compute $C = A \times B$, the unsigned product of A and B. Let the upper half of C be C-hi and its lower half C-lo.
3. If A is negative, then add $(2^{16}) - B$ to C-hi. This can be easily done using the SET C, SUBC instructions of the HPC. Let the result be C-hi1.
4. If B is negative, then add $(2^{16}) - A$ to C-hi1. Again it is easily done using the SET C, SUBC instructions. Let the result be C-hi2.
5. The 2's complement product of A and B is C-hi2. C-lo.

FIGURE 4. Algorithm for 2's Complement Multiplication.

MULTIPLICATION BY FILTER COEFFICIENTS

The coefficients that arise in most IIR filter designs are numbers that are usually in the range from $-2 < \text{coeff} < 2$. The coefficients, in most instances can be scaled to be in this range. The action of digital filtering involves successive multiplications. If we want no loss in accuracy due to multiplication, the word length needed to store successive partial products increases rapidly—clearly an impractical choice. Thus the results of the multiplication at the various stages need to be truncated to 16 bits before proceeding to the next stage. The program FILTER does this as follows: The filter state variables are regarded as integers, while the filter coefficients are regarded as fixed point fractions with the binary point to the immediate right of the sign bit. After the multiplication, the result is shifted so that the integer part of the product is in one word, and the fractional part in another. The integer part is then returned as the result of the multiplication, i.e. the product is truncated to 16 bits. This is per-

formed by the subroutine SMULT. Since the filter coefficients are regarded as fixed point fractions, only coefficients in the range $-1 < \text{coeff} < 1$ can be represented. However, as discussed earlier, the coefficients are usually in the $-2 < \text{coeff} < 2$ range. This is handled by storing half the coefficient value, and SMULT performs a multiplication by 2 (Shift left) to compensate for it. This is why the coefficient values need to be half their value—a fact mentioned earlier.

REFERENCES

1. A.V. Oppenheim and R.W. Schaffer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
2. L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
3. H.T. Nagle and V.P. Nelson, "Digital Filter Implementation on 16-bit Microcomputers", *IEEE Micro*, Feb. 1981, pp. 23-41.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

Listing of Code for the Program FILTER

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 1

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

1          ;
2          ; THIS IS A DEMO PROGRAM TO ILLUSTRATE THE IMPLEMENTATION OF A DIGITAL
3          ; FILTER ON THE HPC. THE PROGRAM CAN BE USED TO IMPLEMENT CASCADED
4          ; SECOND ORDER STAGES. THE MAXIMUM NUMBER OF CASCADED STAGES POSSIBLE
5          ; IS 8 (I.E. THE MAXIMUM FILTER ORDER IS 16).
6          ;
7          ; THE PROGRAM IS DESIGNED FOR THE ANALOG INTERFACE BEING THROUGH
8          ; A CODEC. THE CODEC OUTPUT AND INPUT ARE INTERFACED TO THE HPC USING
9          ; MICROWIRE/PLUS. THIS RESTRICTS THE SAMPLING RATE TO 8 KHZ. ALSO, AT
10         ; THIS SAMPLING RATE, THE HPC CAN ONLY IMPLEMENT A SECOND ORDER FILTER.
11         ; IF A DIFFERENT ANALOG INTERFACE THAT ALLOWS A LOWER SAMPLING RATE IS
12         ; USED, HIGHER ORDER FILTERS CAN BE IMPLEMENTED. THIS WILL INVOLVE CHANGES
13         ; TO THE FOLLOWING SUBROUTINES: INPUT, OUTPUT AND THE PORTIONS OF INIT
14         ; CONCERNED WITH CODEC INITIALIZATION.
15         ;
16         ; THE PROGRAM IS BASED ON THE DESCRIPTION GIVE IN:
17         ;
18         ;           H.T. NAGLE AND V.P. NELSON, "DIGITAL FILTER IMPLEMENTATION
19         ;           ON 16-BIT MICROCOMPUTERS," IEEE MICRO, FEB. 1981, 23-41.
20         ;
21         ;
22         ;           .TITLE FILTER
23         ;
24         ; DEFINE FILTER VARIABLES AND STORAGE.
25         ;
26         0000          YOUT = M(00)          ; OUTPUT SAMPLE STORAGE.
27         0002          YOFK = W(02)          ; TEMPORARY STORAGE.
28         0004          NSTG = W(04)          ; NUMBER OF FILTER STAGES.
29         0006          NCNT = W(06)          ; TEMPORARY STORAGE.
30         0008          PTEMP = W(08)         ; TEMPORARY STORAGE.
31         000A          MTEMP = W(0A)         ; TEMPORARY STORAGE.
32         000C          AOADDR = W(0C)        ; ADDRESS OF START OF AO AREA.
33         000E          A1ADDR = W(0E)        ; ADDR. OF START OF A1 AREA.
34         0010          A2ADDR = W(010)       ; ADDR. OF START OF A2 AREA.
35         0012          B1ADDR = W(012)       ; ADDR. OF START OF B1 AREA.
36         0014          B2ADDR = W(014)       ; ADDR. OF START OF B2 AREA.
37         0016          M0ADDR = W(016)       ; ADDR. OF START OF M0 AREA.
38         0018          M1ADDR = W(018)       ; ADDR. OF START OF M1 AREA.
39         001A          M2ADDR = W(01A)       ; ADDR. OF START OF M2 AREA.
40         001C          T1ADDR = W(01C)       ; ADDR. OF START OF T1 AREA.
41         001E          T2ADDR = W(01E)       ; ADDR. OF START OF T2 AREA.
42         ; MAXIMUM NUMBER OF STAGES IS 8.
43         0020          A0 = W(020)           ; COEFF. A0.
44         0030          A1 = W(030)           ; COEFF. A1.
45         0040          A2 = W(040)           ; COEFF. A2.
46         0050          B1 = W(050)           ; COEFF. B1.
47         0060          B2 = W(060)           ; COEFF. B2.
48         0070          M0 = W(070)           ; M(K).
49         0080          M1 = W(080)           ; M(K-1).
50         0090          M2 = W(090)           ; M(K-2).
51         00A0          T1 = W(0A0)           ; T1.

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 2

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

52      00B0                T2 = W(0B0)                ; T2.
53                ;
54                ; DEFINITION OF HPC REGISTER NAMES.
55                ;
56      00C0                PSW = M(00C0)
57      00D0                ENIR = M(00D0)
58      00D2                IRPD = M(00D2)
59      00D4                IRCD = M(00D4)
60      00D6                SIO = M(00D6)
61      00D8                PORTI = M(00D8)
62      00E2                PORTBL = M(00E2)
63      00E3                PORTBH = M(00E3)
64      00E2                PORTB = W(00E2)
65      00F2                DIRBL = M(00F2)
66      00F3                DIRBH = M(00F3)
67      00F2                DIRB = W(00F2)
68      00F4                BFUNL = M(00F4)
69      00F5                BFUNH = M(00F5)
70      00F4                BFUN = W(00F4)
71      0188                T2TIM = W(0188)
72      0186                T2REG = W(0186)
73      018E                DIVBYL = M(018E)
74      018F                DIVBYH = M(018F)
75      018E                DIVBY = W(018E)
76      0190                TMMDL = M(0190)
77      0191                TMMDH = M(0191)
78      0190                TMMD = W(0190)
79                ;
80                ; INCLUDE THE MU-LAW TO LINEAR CODE CONVERSION TABLE.
81                ;
82                .INCLD MUTBL.MAC
83 F000                MUTBL, OF000
84                ;
85 F200                . = OF200
86      FILTER:
87 F200 B701FOC4        LD SP, 01F0                ; INITIALIZE STACK POINTER.
88 F204 305A            JSR INIT                    ; INITIALIZE THE CODEC
89                                ; AND FILTER VARIABLES.
90                ; NEXT COMES THE BASIC FILTER LOOP.
91      FLOOP:
92 F206 3101            JSR INPUT                    ; GET INPUT SAMPLE, OUTPUT
93                                ; PREVIOUS FILTER OUTPUT.
94 F208 311B            JSR YCOMP                    ; COMPUTE NEW OUTPUT.
95 F20A 315B            JSR OUTPUT                    ; CONVERT OUTPUT VALUE TO
96                                ; MU-255 LAW AND SAVE.
97 F20C 31AA            JSR PRECOM                    ; PRECOMPUTE FOR NEXT SAMPLE.
98 F20E 68              JP FLOOP                    ; GO DO NEXT SAMPLE.
99                                .LOCAL
100                ;
101                ;
102                ;

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 3

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

103 ; THIS SUBROUTINE COMPUTES 2*F*I, WHERE F IS A 2'S COMPLEMENT
104 ; BINARY FRACTION AND I IS A 2'S COMPLEMENT INTEGER. THE INTEGER
105 ; PART OF THE PRODUCT IS RETURNED IN A. ON INPUT, EITHER F OR I
106 ; SHOULD BE IN A AND THE ADDRESS OF THE OTHER IN B.
107 ;
108 ;
109 SMULT:
110 F20F B700000A LD MTEMP, 0 ; CLEAR TEMPORARY STORAGE.
111 F213 A9CC INC B ; B NOW POINTS TO UPPER BYTE
112 ; OF MULTIPLIER.
113 F215 17 IF M(B).7 ; IS IT NEGATIVE?
114 F216 ABOA ST A, MTEMP ; THEN SAVE MULTIPLICAND IN MTEMP.
115 F218 AACC DECSZ B ; B INTO WORD POINTER.
116 F21A 40 NOP
117 F21B AEOA X A, MTEMP ; SWAP A AND MTEMP.
118 F21D 960B17 IF M(($MTEMP) + 1).7 ; IS MULTIPLICAND NEGATIVE?
119 F220 F8 ADD A, W(B) ; THEN ACCUMULATE MULTIPLIER
120 F221 AEOA X A, MTEMP
121 F223 FE MULT A, W(B) ; UNSIGNED MULTIPLY.
122 F224 AECE X A, X ; UPPER HALF IN A.
123 F226 02 SET C
124 F227 960AEB SUBC A, MTEMP
125 F22A E7 SHL A
126 F22B 96CF17 IF H(X).7
127 F22E 04 INC A
128 F22F E7 SHL A
129 F230 96CF16 IF H(X).6
130 F233 04 INC A
131 F234 3C RET
132 ;
133 ;
134 ; THIS SUBROUTINE PERFORMS THE INITIALIZATION FOR THE FILTER.
135 ; IT DOES THE FOLLOWING:
136 ; 1. SET UP THE FILTER VARIABLES.
137 ; 2. COPY THE FILTER COEFFS. FROM ROM TO ON CHIP RAM.
138 ; 3. INITIALIZE AND START THE CODEC.
139 ;
140 ;
141 ; DEFINE FILTER COEFFICIENTS.
142 ;
143 F235 40 .EVEN
144 F236 0400 ROMNST: .WORD 4
145 F238 C430 ROMA0: .WORD 12484, 3217, 4574, 7636
    F23A 910C
    F23C DE11
    F23E D41D
146 F240 C430 ROMA1: .WORD 12484, 3217, 4574, 7636
    F242 910C
    F244 DE11
    F246 D41D
147 F248 C430 ROMA2: .WORD 12484, 3217, 4574, 7636

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 4

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

F24A 910C
F24C DE11
F24E D41D
148 F250 B939      ROMB1:      .WORD 14777, 9826, 19308, 11207
F252 6226
F254 6C4B
F256 C72B
149 F258 A1D5      ROMB2:      .WORD -10847, -15783, -6940, -14068
F25A 59C2
F25C E4E4
F25E OCC9
150                INIT:
151 F260 B6F236A8      LD A, W(ROMNST)
152 F264 AB04          ST A, NSTG          ; SET UP NO. OF STAGES.
153 F266 9020          LD A, $A0
154 F268 AB0C          ST A, AOADDR        ; COPY ADDRESS OF A0 AREA.
155 F26A 9030          LD A, $A1
156 F26C ABOE          ST A, A1ADDR        ; COPY ADDRESS OF A1 AREA.
157 F26E 9040          LD A, $A2
158 F270 AB10          ST A, A2ADDR        ; COPY ADDRESS OF A2 AREA.
159 F272 9050          LD A, $B1
160 F274 AB12          ST A, B1ADDR        ; COPY ADDRESS OF B1 AREA.
161 F276 9060          LD A, $B2
162 F278 AB14          ST A, B2ADDR        ; COPY ADDRESS OF B2 AREA.
163 F27A 9070          LD A, $M0
164 F27C AB16          ST A, MOADDR        ; COPY ADDRESS OF M0 AREA.
165 F27E 9080          LD A, $M1
166 F280 AB18          ST A, M1ADDR        ; COPY ADDRESS OF M1 AREA.
167 F282 9090          LD A, $M2
168 F284 AB1A          ST A, M2ADDR        ; COPY ADDRESS OF M2 AREA.
169 F286 90A0          LD A, $T1
170 F288 AB1C          ST A, T1ADDR        ; COPY ADDRESS OF T1 AREA.
171 F28A 90B0          LD A, $T2
172 F28C AB1E          ST A, T2ADDR        ; COPY ADDRESS OF T2 AREA.
173                ;
174                ; COPY THE A0 COEFFS. TO ON-CHIP RAM.
175                ;
176 F28E B3F238          LD X, ROMAO
177 F291 9220          LD B, $A0
178 F293 AC04CA          LD K, NSTG
179                CAOLP:
180 F296 F0             LD A, W(X+)
181 F297 E1             XS A, W(B+)
182 F298 40             NOP
183 F299 AACA          DECSZ K
184 F29B 65             JP CAOLP
185                ;
186                ; COPY THE A1 COEFFS. TO ON-CHIP RAM.
187 F29C B3F240          LD X, ROMA1
188 F29F 9230          LD B, $A1
189 F2A1 AC04CA          LD K, NSTG

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 5

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

190          CALLP:
191 F2A4 FO          LD A, W(X+)
192 F2A5 E1          XS A, W(B+)
193 F2A6 40          NOP
194 F2A7 AACA        DECSZ K
195 F2A9 65          JP CALLP
196          ;
197          ; COPY THE A2 COEFFS. TO ON-CHIP RAM.
198 F2AA B3F248      LD X, ROMA2
199 F2AD 9240        LD B, $A2
200 F2AF AC04CA      LD K, NSTG
201          CA2LP:
202 F2B2 FO          LD A, W(X+)
203 F2B3 E1          XS A, W(B+)
204 F2B4 40          NOP
205 F2B5 AACA        DECSZ K
206 F2B7 65          JP CA2LP
207          ;
208          ; COPY THE B1 COEFFS. TO ON-CHIP RAM.
209 F2BB B3F250      LD X, ROMB1
210 F2BB 9250        LD B, $B1
211 F2BD AC04CA      LD K, NSTG
212          CB1LP:
213 F2C0 FO          LD A, W(X+)
214 F2C1 E1          XS A, W(B+)
215 F2C2 40          NOP
216 F2C3 AACA        DECSZ K
217 F2C5 65          JP CB1LP
218          ;
219          ; COPY THE B2 COEFFS. TO ON-CHIP RAM.
220 F2C6 B3F258      LD X, ROMB2
221 F2C9 9260        LD B, $B2
222 F2CB AC04CA      LD K, NSTG
223          CB2LP:
224 F2CE FO          LD A, W(X+)
225 F2CF E1          XS A, W(B+)
226 F2D0 40          NOP
227 F2D1 AACA        DECSZ K
228 F2D3 65          JP CB2LP
229          ;
230          ; ZERO OUT THE REST OF USER BASE PAGE RAM.
231          ;
232 F2D4 8D70BE      LD BK, $M0, OBE
233          ZEROLP:
234 F2D7 00          CLR A
235 F2D8 E1          XS A, W(B+)
236 F2D9 62          JP ZEROLP
237          ;
238          ;
239          ; NOW INITIALIZE AND START THE CODEC.
240          ;

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 6

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

241      ;
242 F2DA B7FFB7F2      LD DIRB,OFFB7      ; SET B3 (T2IO) AND B6 (SK)
243      ;
244      ;
245 F2DE B70000E2      LD PORTB, 0      ; OUTPUT 0 ON ALL PORT B PINS.
246 F2E2 96F40B      SET BFUNL.3      ; ALT. FUN. ON B3-T2IO.
247 F2E5 96F40D      SET BFUNL.5      ; ALT. FUN. ON B5-SO.
248 F2E8 96F508      SET BFUNH.0      ; ALT. FUN. ON B8-TSO.
249 F2EB 9700D0      LD ENIR, 0      ; DISABLE INTRPTS.
250 F2EE 9700D4      LD IRCD, 0      ; SELECT SLAVE MODE FOR M-WIRE.
251 F2F1 83070188AB    LD T2TIM, 07      ; LOAD 7-DEC INTO T2 TIMER.
252 F2F6 83070186AB    LD T2REG, 07      ; LOAD 7-DEC INTO T2 REG.
253 F2FB 8300018FB8    LD DIVBYH, 0      ; SELECT EXT. CLOCK FOR T2 TIMER.
254      ;
255 F300 8ED6      X A, SIO
256 F302 8740400190AB  LD TMMD, 04040    ; START TIMER T2.
257 F308 3C      RET
258      ;
259      ;
260      ; THIS SUBROUTINE OUTPUTS THE PREVIOUS Y(K) TO THE CODEC AND READS
261      ; THE NEW INPUT VALUE. THEN THE MU-255 VALUE IS CONVERTED TO LINEAR
262      ; BY TABLE LOOK UP. THE TABLE IS ASSUMED TO START AT F000.
263      ;
264      ;
265      INPUT:
266 F309 AB02      LD A, YOFK      ; GET DATA TO BE OUTPUT.
267      NOTDN:
268 F30B 96D210      IF IRPD.0      ; IS MICROWIRE DONE?
269 F30E 41      JP MWDONE      ; YES, SO GET DATA.
270 F30F 64      JP NOTDN      ; NO, SO TRY AGAIN.
271      MWDONE:
272 F310 8ED6      X A, SIO      ; GET NEW SAMPLE, OUTPUT
273      ; COMPUTED DATA.
274 F312 01      COMP A      ; TAKE CARE OF CODEC INVERSION.
275 F313 99FF      AND A, OFF
276 F315 E7      SHL A
277 F316 BAF000     OR A, OF000      ; FORM MU-LAW TO LINEAR
278      ; TABLE ADDRESS.
279 F319 AECE      X A, X
280 F31B D0      LD A, M(X+)      ; GET LINEAR VALUE
281 F31C AECA      X A, K
282 F31E D4      LD A, M(X)      ; A BYTE AT A TIME.
283 F31F 8CC8CB     LD H(K), L(A)
284 F322 ABCA      LD A, K
285 F324 3C      RET
286      ;
287      ;
288      YCOMP:
289      ; THIS SUBROUTINE COMPUTES THE OUTPUT SAMPLE Y(K).
290      ; THE INPUT SAMPLE X(K) IS INPUT IN REG. A.
291      ; THE OUTPUT IS RETURNED IN REG. A.

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 7

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

292 ;
293 F325 AC0406 LD NCNT, NSTG ; COPY THE NUMBER OF STAGES TO
294 ; NCNT.
295 YLOOP:
296 F328 AD1CF8 ADD A, W(T1ADDR) ; A ≤ X(K) + T1.
297 F32B AD16AB ST A, W(MOADDR) ; M(K) ≤ X(K) + T1.
298 F32E AC0CCC LD B, AOADDR ; B ≤ ADDR(AO).
299 F331 3522 JSR SMULT ; A ≤ AO*M(K).
300 F333 AD1EF8 ADD A, W(T2ADDR) ; A ≤ AO*M(K) + T2.
301 ;
302 F336 AA06 DECSZ NCNT ; DONE ALL STAGES?
303 F338 941B R JMP YMORE ; NO GO DO SOME MORE.
304 ;
305 ; GET HERE MEANS ALL STAGES DONE.
306 F33A AB02 ST A, YOFK ; SAVE TEMPORARILY.
307 F33C A804 LD A, NSTG
308 F33E 05 DEC A
309 F33F E7 SHL A
310 F340 01 COMP A
311 F341 04 INC A ; A ≤ -2*(NSTG-1).
312 F342 A0C81CF8 ADD T1ADDR, A ; RESTORE T1ADDR.
313 F346 A0C816F8 ADD MOADDR, A ; RESTORE MOADDR.
314 F34A A0C80CF8 ADD AOADDR, A ; RESTORE AOADDR.
315 F34E A0C81EF8 ADD T2ADDR, A ; RESTORE T2ADDR.
316 F352 A802 LD A, YOFK ; A ≤ Y(K).
317 F354 3C RET
318 ;
319 ; PREPARE FOR NEXT STAGE ITERATION.
320 ;
321 YMORE:
322 F355 82021CF8 ADD T1ADDR, 02
323 F359 820216F8 ADD MOADDR, 02
324 F35D 82020CF8 ADD AOADDR, 02
325 F361 82021EF8 ADD T2ADDR, 02
326 F365 953D JMP YLOOP
327 ;
328 ; THIS SUBROUTINE CONVERTS THE 16 BIT OUTPUT VALUE TO
329 ; 8 BIT MU-LAW.
330 ;
331 OUTPUT:
332 F367 96D41F RESET IRCD.7
333 F36A E7 SHL A ; SIGN BIT TO C.
334 F36B 06 IFN C ; IS IT POSITIVE?
335 F36C 45 JP OPOS
336 F36D 96D40F SET IRCD.7
337 F370 01 COMP A
338 F371 04 INC A ; NEGATIVE, SO TAKE 2'S
339 ; COMPLEMENT.
340 ; OPOS:
341 F372 B80108 ADD A, 0108 ; ADD BIAS.
342 F375 9107 LD K, 07 ; SET UP COUNTER.

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 8

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

343          ALIGN:
344 F377 E7          SHL A          ; LOOP AND LOCATE MS 1 BIT.
345 F378 07          IF C
346 F379 44          JP ODONE          ; FOUND MS 1 BIT.
347 F37A AACA        DECSZ K
348 F37C 65          JP ALIGN
349 F37D E7          SHL A          ; HAS TO BE 1 IN C NOW.
350          ODONE:
351 F37E AECA        X R, K
352 F380 E7          SHL A
353 F381 E7          SHL A
354 F382 E7          SHL A
355 F383 E7          SHL A          ; COUNTER VALUE IN BITS 4-6.
356 F384 AECC        X A, B
357 F386 00          CLR A
358 F387 88CB        LD A, H(K)
359 F389 3B          SWAP A
360 F38A 990F        AND A, OF
361 F38C 96CCFA      OR A, B
362 F38F 96D417      IF IRCD.7
363 F392 96C80F      SET A.7
364 F395 01          COMP A
365 F396 8B00        ST A, YOUT
366 F398 3C          RET
367          ;
368          ; THIS SUBROUTINE UPDATES M(K-1) AND M(K-2) FOR THE NEXT SAMPLE.
369          ;
370 F399 AC1ACC        LD B, M2ADDR          ; B ≤ ADDR(M2),
371 F39C AC04CA        LD K, NSTG          ; K ≤ NSTG.
372 F39F AC18CE        LD X, M1ADDR          ; X ≤ ADDR(M1).
373          DLYLP1:
374 F3A2 F0          LD A, W(X+)          ; A ≤ M(K-1).
375 F3A3 E1          XS A, W(B+)          ; M(K-2) ≤ M(K-1).
376 F3A4 40          NOP
377 F3A5 AACA        DECSZ K
378 F3A7 65          JP DLYLP1
379          ;
380 F3A8 AC18CC        LD B, M1ADDR          ; B ≤ ADDR(M1),
381 F3AB AC04CA        LD K, NSTG          ; K ≤ NSTG.
382 F3AE AC16CE        LD X, M0ADDR          ; X ≤ ADDR(M0).
383          DLYLP2:
384 F3B1 F0          LD A, W(X+)          ; A ≤ M(K).
385 F3B2 E1          XS A, W(B+)          ; M(K-1) ≤ M(K).
386 F3B3 40          NOP
387 F3B4 AACA        DECSZ K
388 F3B6 65          JP DLYLP2
389 F3B7 3C          RET
390          ;
391          ;
392          PRECOMP:
393          ; THIS SUBROUTINE PRECOMPUTES T1 AND T2 BEFORE THE NEXT INPUT

```


APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 9

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```

394          ; SAMPLE ARRIVES.
395          ;
396 F3B8 AC0406      LD NCNT, NSTG          ; COPY NO. OF STAGES.
397          PRELP:
398 F3BB AD18A8      LD A, W(M1ADDR)        ; A ≤ M(K-1).
399 F3BE AC12CC      LD B, B1ADDR          ; B ≤ ADDR(-B1).
400 F3C1 35B2        JSR SMULT              ; A ≤ -B1*M(K-1).
401 F3C3 AB08        ST A, PTEMP
402 F3C5 AD1AA8      LD A, W(M2ADDR)        ; A ≤ M(K-2).
403 F3C8 AC14CC      LD B, B2ADDR          ; B ≤ ADDR(-B2).
404 F3CB 35BC        JSR SMULT              ; A ≤ -B2*M(K-2).
405 F3CD 9608F8      ADD A, PTEMP           ; A ≤ -B1*M(K-1) - B2*M(K-2).
406 F3D0 AD1CAB      ST A, W(T1ADDR)
407 F3D3 AD18A8      LD A, W(M1ADDR)        ; A ≤ M(K-1).
408 F3D6 ACOECC      LD B, A1ADDR          ; B ≤ ADDR(A1).
409 F3D9 35CA        JSR SMULT              ; A ≤ A1*M(K-1).
410 F3DB AB08        ST A, PTEMP
411 F3DD AD1AA8      LD A, W(M2ADDR)        ; A ≤ M(K-2).
412 F3E0 AC10CC      LD B, A2ADDR          ; B ≤ ADDR(A2).
413 F3E3 3504        JSR SMULT              ; A ≤ A2*M(K-2).
414 F3E5 9608F8      ADD A, PTEMP           ; A ≤ A1*M(K-1) + A2*M(K-2).
415          ;
416 F3E8 AA06        DECSZ NCNT             ; DONE ALL STAGES?
417 F3EA 9427        JMP FMORE              ; NO, GO DO SOME MORE.
418          ;
419          ; GET HERE MEANS DONE ALL STAGES.
420 F3EC A804        LD A, NSTG
421 F3EE 05          DEC A
422 F3EF E7          SHL A
423 F3F0 01          COMP A
424 F3F1 04          INC A                  ; A ≤ -2*(NSTG - 1).
425 F3F2 A0C818F8    ADD M1ADDR, A          ; RESTORE M1ADDR.
426 F3F6 A0C81AF8    ADD M2ADDR, A          ; RESTORE M2ADDR.
427 F3FA A0C81CF8    ADD T1ADDR, A          ; RESTORE T1ADDR.
428 F3FE A0C81EF8    ADD T2ADDR, A          ; RESTORE T2ADDR.
429 F402 A0C812F8    ADD B1ADDR, A          ; RESTORE B1ADDR.
430 F406 A0C814F8    ADD B2ADDR, A          ; RESTORE B2ADDR.
431 F40A A0C80EF8    ADD A1ADDR, A          ; RESTORE A1ADDR.
432 F40E A0C810F8    ADD A2ADDR, A          ; RESTORE A2ADDR.
433 F412 3C          RET
434          ;
435          ; PREPARE FOR NEXT STAGE ITERATION.
436          ;
437          FMORE:
438 F413 820218F8    ADD M1ADDR, 02          ; UPDATE M1ADDR.
439 F417 82021AF8    ADD M2ADDR, 02          ; UPDATE M2ADDR.
440 F41B 82021CF8    ADD T1ADDR, 02          ; UPDATE T1ADDR.
441 F41F 82021EF8    ADD T2ADDR, 02          ; UPDATE T2ADDR.
442 F423 820212F8    ADD B1ADDR, 02          ; UPDATE B1ADDR.
443 F427 820214F8    ADD B2ADDR, 02          ; UPDATE B2ADDR.
444 F42B 82020EF8    ADD A1ADDR, 02          ; UPDATE A1ADDR.

```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 10

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

```
445 F42F 820210F8          ADD A2ADDR, 02          ; UPDATE A2ADDR.
446 F433 9578             JMP PRELP
447                       ;
448                       ;
449 FFFE 00F2             .END FILTER
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 11

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

SYMBOL TABLE

A	00C8 W	A0	0020 W	A0ADDR	000C W	A1	0030 W
ALADDR	000E W	A2	0040 W	A2ADDR	0010 W	ALIGN	F377
B	00CC W	B1	0050 W	B1ADDR	0012 W	B2	0060 W
B2ADDR	0014 W	BFUN	00F4 W*	BFUNH	00F5 M	BFUNL	00F4 M
CAOLP	F296	CALLP	F2A4	CA2LP	F2B2	CB1LP	F2C0
CB2LP	F2CE	DIRB	00F2 W	DIRBH	00F3 M*	DIRBL	00F2 M*
DIVBY	018E W*	DIVBYH	018F M	DIVBYL	018E M*	DLYLP1	F3A2
DLYLP2	F3B1	ENIR	00D0 M	FILTER	F200	FLOOP	F206
INCRM	0200	INIT	F260	INPUT	F309	IRCD	00D4 M
IRPD	00D2 M	K	00CA W	MO	0070 W	MOADDR	0016 W
M1	0080 W	M1ADDR	0018 W	M2	0090 W	M2ADDR	001A W
MTEMP	000A W	MUAL	205F	MWDONE	F310	NCNT	0006 W
NOIDN	F30B	NSTG	0004 W	ODONE	F37E	OPOS	F372
OUTPUT	F367	PC	00C6 W	PMORE	F413	PORTB	00E2 W
PORTBH	00E3 M*	PORTBL	00E2 M*	PORTI	0008 M*	PRECOM	F3B8
PRELP	F3BB	PSW	00C0 M*	PTEMP	0008 W	ROMA0	F238
ROMA1	F240	ROMA2	F248	ROMB1	F250	ROMB2	F258
ROMNST	F236	RVAL	EOA1	SIO	00D6 M	SMULT	F20F
SP	00C4 W	SVAL	2100	T1	00A0 W	T1ADDR	001C W
T2	00B0 W	T2ADDR	001E W	T2REG	0186 W	T2TIM	0188 W
TMMD	0190 W	TMMDH	0191 M*	TMMDL	0190 M*	X	00CE W
YCOMP	F325	YLOOP	F328	YMORE	F355	YOFK	0002 W
YOUT	0000 M	ZEROLP	F2D7				

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

NATIONAL SEMICONDUCTOR CORPORATION PAGE: 12

HPC CROSS ASSEMBLER, REV:C, 30 JUL 86

FILTER

MACRO TABLE

MUTBL

NO WARNING LINES

NO ERROR LINES

1079 ROM BYTES USED

SOURCE CHECKSUM = 4769

OBJECT CHECKSUM = 1378

INPUT FILE C:FILTER.MAC

LISTING FILE C:FILTER.PRN

OBJECT FILE C:FILTER.LM

A Floating Point Package for the HPC

National Semiconductor
Application Note 486
Ashok Krishnamurthy



INTRODUCTION

This report describes the implementation of a Single Precision Floating Point Arithmetic package for the National Semiconductor HPC microcontroller. The package is based upon the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985). However, the package is not a conforming implementation of the standard. The differences between the HPC implementation and the standard are described later in this report.

The following single precision (SP) operations have been implemented in the package.

- (1) **FADD.** Addition of two SP floating point (FLP) numbers.
- (2) **FSUB.** Subtraction of two SP FLP numbers.
- (3) **FMULT.** Multiplication of two SP FLP numbers.
- (4) **FDIV.** Division of two SP FLP numbers.
- (5) **ATOF.** Convert an ASCII string representing a decimal FLP number to a binary SP FLP number.
- (6) **FTOA.** Convert a binary SP FLP number to a decimal FLP number and output the decimal FLP number as an ASCII string.

The report is organized as follows. The next section discusses the representation of FLP numbers. Then, the differences between the HPC implementation and the IEEE/ANSI standard are described. This is followed by a description of the algorithms used in the computations. Appendix A is a User's Manual for the package, Appendix B describes the test data for the package and Appendix C is a listing of the code.

Note that this report assumes that the reader is familiar with the IEEE/ANSI Binary Floating-Point Standard. Please refer to this document for an explanation of the terms used here.

REPRESENTATION OF FLOATING POINT NUMBERS

The specification of a binary floating point number involves two parts: a mantissa and an exponent. The mantissa is a signed fixed point number and the exponent is a signed integer. The IEEE/ANSI standard specifies that a SP FLP number shall be represented in 32 bits as shown in *Figure 1*.

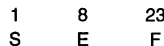


FIGURE 1

The significance of each of these fields is as follows:

1. **S**—this 1-bit field is the sign of the mantissa. $S = 0$ means that the number is positive, while $S = 1$ means that it is negative.
2. **E**—this is the 8-bit exponent field. The exponent is represented as a biased value with a bias of 127-decimal.
3. **F**—this is the 23-bit mantissa field. For normalized FLP numbers (see below), a MSB of 1 is assumed and not represented. Thus, for normalized numbers, the value of the mantissa is 1.F. This provides an effective precision of 24 bits for the mantissa.

Normalized FLP number: A binary FLP number is said to be normalized if the value of the MSB of the mantissa is 1. Normalization is important and useful because it provides maximum precision in the representation of the number. If we deal with normalized numbers only (as the HPC imple-

mentation does) then since the MSB of the mantissa is always 1, it need not be explicitly represented. This is as specified in the IEEE/ANSI standard.

Given the values of S , E and F , the value of the SP FLP number is obtained as follows.

If $0 < E < 255$, then the FLP number is $(-1)^S \cdot 1.F \cdot 2^{(E-127)}$.

If $E = 0$, then the value of the FLP number is 0.

If $E = 255$, then the FLP number is not a valid number (NaN).

The above format for binary SP FLP numbers provides for the representation of numbers in the range $-3.4 \cdot 10^{38}$ to $-1.75 \cdot 10^{-38}$, 0, and $1.75 \cdot 10^{-38}$ to $3.4 \cdot 10^{38}$. The accuracy is between 7 and 8 decimal digits.

DIFFERENCES BETWEEN THE IMPLEMENTATION AND THE IEEE/ANSI STANDARD

The IEEE/ANSI standard specifies a comprehensive list of operations and representations for FLP numbers. Since an implementation that fully conforms to this standard would lead to an excessive amount of overhead, a number of the features in the standard were dropped. This section describes the differences between the implemented package and the standard.

1. Omission of -0 . The IEEE/ANSI standard requires that both $+$ and $-$ zero be represented, and arithmetic carried out using both. The implementation does not represent -0 . Only $+0$ is represented and arithmetic is carried out with $+0$ only.
2. Omission of Infinity Arithmetic. The IEEE/ANSI standard provides for the representation of plus and minus Infinity, and requires that valid arithmetic operations be carried out on Infinity. The HPC implementation does not support this.
3. Omission of Quiet NaN. The IEEE/ANSI standard provides for both quiet and signalling NaNs. The HPC implementation provides for signalling NaNs only. A signalling NaN can be produced as the result of overflow during an arithmetic operation. If the NaN is passed as input to further floating point routines, then these routines will produce another NaN as output. The routines will also set the Invalid Operation flag, and call the user floating point error trap routine at address FPTRAP.
4. Omission of denormalized numbers. Denormalized numbers are FLP numbers with a biased exponent, E of zero and a non zero mantissa F . Such denormalized numbers are useful in providing gradual underflow to zero. Denormalized numbers are not represented or used in the HPC implementation. Instead, if the result of a computation cannot be represented as a normalized number within the allowable exponent range, then an underflow is signaled, the result is set to zero, and the user floating point error trap routine at address FPTRAP is called.
5. Omission of the Inexact Result exception. The IEEE/ANSI standard requires that an Inexact Result exception be signaled when the rounded result of an operation is not exact, or it overflows without an overflow trap. This feature is not provided in the HPC implementation.

6. **Biased Rounding to Nearest.** The IEEE/ANSI standard requires that rounding to nearest be provided as the default rounding mode. Further, the rounding is required to be unbiased. The HPC implementation provides biased rounding to nearest only. An example will help clarify this. Suppose the result of an operation is .b1b2b3XXX and needs to be rounded to 3 binary digits. Then if XXX is 0YY, the round to nearest result is .b1b2b3. If XXX is 1YY, with at least one of the Y's being 1, then the result is .b1b2b3 + 0.001. Finally if XXX is 100, it is a tie situation. In such a case, the IEEE/ANSI standard requires that the rounded result be such that its LSB is 0. The HPC implementation, on the other hand, will round the result in such a case to .b1b2b3 + 0.001.

DESCRIPTION OF ALGORITHMS

1. **General Considerations.** The HPC implementation of the SP floating point package consists of a series of subroutines. The subroutines have been designed to be compatible with the CCHPC C Cross Compiler. They have, however, not been tested with the CCHPC Cross Compiler.

The Arithmetic subroutines that compute F1 op F2 (where op is +, -, * or /) expect that F1 and F2 are input in the IEEE format. Each of F1 and F2 consists of two 16-bit words organized as follows.

F_n-HI: S EXP 7 MS bits of F

F_n-LO: 16 LS bits of F

In the above, S is the sign of the mantissa, EXP is the biased exponent, and F is the mantissa.

On input it is assumed that F1-HI is in register K, F1-LO is in the accumulator A, and F2-HI and F2-LO are on the stack just below the return address i.e., F2-HI is at W(SP-4) and F2-LO is at W(SP-6). The result, C, is also returned in IEEE format with C-HI in register K and C-LO in the accumulator A.

The two Format Conversion routines, ATOF and FTOA expect that on entry, register B contains the address of the start of the ASCII byte string representing the decimal FLP number. ATOF reads the byte string starting from this address. Note that the string must be terminated with a null byte. The binary floating point number is returned in registers K and A. FTOA, on the other hand, writes the decimal FLP string starting from the address in register B on entry. A terminating null byte is also output. Also, FTOA expects that the binary FLP number to be converted is in registers K and A on entry.

Most of the storage required by the subroutines is obtained from the stack. Two additional words of storage in the base page are also used. The first is W(0), and is referenced in the subroutines as W(TMP1). The second word of storage can be anywhere in the base page and is used to store the sticky flags used to signal floating point exceptions. This is referenced in the subroutines as W(FPERWD). Thus any user program that uses the floating point package needs to have the symbols TMP1 and FPERWD defined appropriately.

2. **Exception Handling.** The following types of exception can occur during the course of a computation.

- (i) **Invalid Operand.** This exception occurs if one of the input operands is a NaN.
- (ii) **Exponent Overflow.** This occurs if the result of a computation is such that its exponent has a biased value of 255 or more.

(iii) **Exponent Underflow.** This occurs if the result of a computation is such that its exponent is 0 or less.

(iv) **Divide-by-zero.** This exception occurs if the FDIV routine is called with F2 being zero.

The package signals exceptions in two ways. First a word at address FPERWD is maintained that records the history of these exception conditions. Bits 0-3 of this word are used for this purpose.

Bit 0—Set on Exponent Overflow.

Bit 1—Set on Exponent Underflow.

Bit 2—Set on Illegal Operand.

Bit 3—Set on Divide-by-zero.

These bits are never cleared by the floating point package, and can be examined by the user software to determine the exception conditions that occurred during the course of a computation. It is the responsibility of the user software to initialize this word before calling any of the floating point routines.

The second method that the package uses to signal exceptions is to call a user floating point exception handler subroutine whenever an exception occurs. The corresponding exception bit in FPERWD is set before calling the handler. The starting address of the handler should be defined by the symbol FPTRAP.

3. **Unpacked Floating Point Format.** The IEEE/ANSI standard floating point format described earlier is very cumbersome to deal with during computation. This is primarily because of the splitting of the mantissa between the two words. The subroutines in the package unpack the input FLP numbers into an internal representation, do the computations using this representation, and finally pack the result into the IEEE format before return to the calling program. The unpacking is done by the subroutine FUNPAK and the packing by the subroutine FPAK. The unpacked format consists of 3 words and is organized as follows.

F_n-EXP.F_n-SIGN 8 bits biased sign (extended to exponent 8 bits)

F_n-HI MS 16 bits of mantissa (implicit 1 is present as MSB)

F_n-LO LS 8 bits of Eight mantissa Zeros

Since all computations are carried out in this format, note that the result is actually known to 32 bits. This 32-bit mantissa is rounded to 24 bits before being packed to the IEEE format.

4. **Algorithm Description.** All the arithmetic algorithms first check for the easy cases when either F1 or F2 is zero or a NaN. The result in these cases is immediately available. The description of the algorithms below is for those cases when neither F1 nor F2 is zero or a NaN. Also, in order to keep the algorithm description simple, the check for underflow/overflow at the various stages is not shown. The documentation in the program, the descriptions given below, and the theory as described in the references should allow these programs to be easily maintained.

(i) **FADD.**

The processing steps are as follows:

1. Compare F1-EXP and F2-EXP. Let the difference be D. Shift right the mantissa (F_n-HI.F_n-LO, n = 1 or 2) of the FLP number with the smaller exponent D times. Let the numbers after this step be F1-EXP.F1-SIGN, F1-HI, F1-LO and F2-EXP.F2-SIGN,

F2-HI and F2-LO. This step equalizes the two exponents.

2. Take the XOR of F1-SIGN and F2-SIGN. If this is 0, then go to step 4, else go to step 3.
3. Do a true subtract of F2-LO from F1-LO. (A true subtract is when the SUBC instruction is preceded by a SET C instruction.) Then do a 1's complement subtract of F2-HI from F1-HI. If the last subtract resulted in C = 1, then go to step 3.2, else go to step 3.1.
 - 3.1. Get here means that F2 is larger than F1, and the computed result is negative. Take the 2's complement of the result to make it positive. Set the sign of the result to be the sign of F2. Go to step 3.3.
 - 3.2. Get here means F1 is larger than F2, and the result of the mantissa subtract is positive. Set the sign of the result to be the sign of F1. Go to step 3.3.
 - 3.3. The result after a subtract need not be normalized. Shift left the result mantissa until its MSB is 1. Decrement the exponent of the result by 1 for each such left shift. Go to step 5.
4. Add F2-LO to F1-LO. Next add with any carry from the previous add, F2-HI to F1-HI. If this last add results in C = 1, then go to step 4.1, else go to step 5.
 - 4.1. Rotate Right with carry C-HI. Next load C-LO in and rotate it right with carry. Increase the exponent of the result, C by 1. Go to step 5.
5. Round the result. Go to step 6.
6. Pack the result and return.

(ii) FSUB.

The processing steps are as follows:

1. Copy F2 to the stack and change its sign. Go to step 2.
2. Call FADD.
3. Remove the copy of -F2 from the stack and return.

(iii) FMULT.

The processing steps are as follows.

1. Add F1-EXP and F2-EXP to get C1-EXP. Subtract from C1-EXP 127-decimal which is the IEEE bias, to get C-EXP. Go to step 2.
2. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 3.
3. Compute F1-HI*F2-HI. Let the upper half of the product be C1-HI and the lower half C1-LO. Go to step 4.
4. Compute F1-HI*F2-LO. Let the upper half of this product be C2-HI. Add C2-HI to C1-LO to give C11-LO. If this last add results in C = 1, then increment C1-HI. Go to step 5.
5. Compute F1-LO*F2-HI. Let the upper half of this product be C3-HI. Add C3-HI to C11-LO to get C12-LO. If this last add results in C = 1, then increment C1-HI. Go to step 6.
6. Mantissa normalization. If the MSB of C1-HI is 1, then increment C-EXP, else shift left C1-HI.C12-LO. Go to step 7.
7. Round C1-HI.C12-LO to get C-HI.C-LO. Go to step 8.

8. Pack C-EXP.C-SIGN, C-HI and C-LO and return as the answer.

(iv) FDIV.

The processing steps are as follows:

1. Compare F1-HI and F2-HI. If F2-HI is greater than F1-HI then go to Step 3, else go to step 2.
2. Shift right F1-HI.F1-LO. Increase F1-EXP by 1.
3. Subtract F2-EXP from F1-EXP. Add to the result 127-decimal to get C1-EXP. Go to step 4.
4. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 5.
5. Compute F1-HI*F2-LO. Let the result be M1-HI.M1-LO. Go to step 6.
6. Divide M1-HI.M1-LO by F2-HI. Let the quotient be M2-HI. Go to step 7.
7. Do a true subtract of M2-HI from F1-LO. Let the result be M3-LO. If C = 1 as a result of this subtract, then go to step 8, else decrement F1-HI and go to step 8.
8. Divide F1-HI.M3-LO by F2-HI. Let the quotient be C1-HI and the remainder R1. Go to step 9.
9. Divide R1 .0000 by F2-HI. Let the quotient be C1-LO. Go to step 10.
10. If the MSB of C1-HI is 1 then go to step 11, else shift left C1-HI.C1-LO, decrease C1-EXP by 1 and go to step 11.
11. Round C1-HI.C1-LO to get C-HI.C-LO. go to step 12.
12. Pack C1-EXP.C-SIGN, C-HI and C-LO and return as the result.

(v) ATOF.

The processing steps in this case are as follows.

1. Set M-SIGN, the mantissa sign to 0.
Set M10-EXP, the implicit decimal exponent to 0.
Set HI-INT to 0.
Set LO-INT to 0.
Go to step 2.
2. Get a character from the input string. Let the character be C.
If C is a '+', then go to the start of step 2.
If C is a '-', then set M-SIGN to FF and go to start of step 2.
If C is a '.', then go to step 5.
If C is none of the above, then go to step 3.
3. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 3.1, else go to step 3.2.
 - 3.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Go to step 4.
 - 3.2. Increase M10-EXP by 1 and go to step 4.
4. Get a character from the input string. Let the character be C.
If C is a '.', then go to step 5.
If C is a 'E', then go to step 7.
If C is the space character, then go to the start of step 4.
If C is none of the above, then go to step 3.

5. Get a character from the input string. Let the character be C.
If C is a 'E', then go to step 7.
If C is the space character, then go to the start of step 5.
If C is none of the above, then go to step 6.
6. Subtract 30 from C to get its integer value. Let this be I. Check and see if $(HI-INT.LO-INT)*10 + 9$ can fit in 32 bits. If it can, then go to step 6.1, else go to step 5.
 - 6.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Decrement M10-EXP by 1. Go to step 5.
7. Set SEXP, the exponent sign to be 0. Go to step 8.
8. Get a character from the input string. Let the character be C.
If C is a '+', then go to start of step 8.
If C is a '-', then set SEXP to be FF and go to the start of step 8.
If C is none of the above, then go to step 9.
9. Set M20-EXP, the explicit decimal exponent to 0. Go to step 10.
10. Subtract 30 from C to get its integer value. Let this be I. Multiply M20-EXP by 10 and add I to the product. Store this sum back in M20-EXP. Go to step 11.
11. Get a character from the input string. Let this be C. If C is the null character, then go to step 12, else go to step 10.
12. Add M10-EXP and M20-EXP (with the proper sign as determined by SEXP) to get the 10's exponent M-EXP. Save in M-EXP the magnitude of the sum and in SEXP the sign of the sum. Go to step 13.
13. Check and see if HI-INT.LO-INT is 0. If it is, then set the resulting floating point number, C, to zero and return. If it is not then go to step 14.
14. Normalize HI-INT.LO-INT by left shifts such that the MSB is 1. Let the number of left shifts needed to do this be L. Set B1-EXP to 32-decimal - L. Go to step 15.
15. If SEXP is 0, then set P-HI.P-LO to the binary representation of 0.625, else set P-HI.P-LO to the binary representation of 0.8. Go to step 16.
16. Multiply HI-INT.LO-INT by P-HI.P-LO M-EXP times. After each multiplication, normalize the partial product if needed by left shifting. Accumulate the number of left shifts needed in B2-EXP. Let the final product be C-HI.C-LO. Go to step 17.
17. Subtract B2-EXP from B1-EXP. Let the result be B-EXP. Go to step 18.
18. If SEXP is 0, then multiply M-EXP by 4, else multiply M-EXP by -3. Let the result be B3-EXP. Go to step 19.
19. Add B-EXP and B3-EXP. Let the result be C1-EXP. Add 126 to C1-EXP to restore the IEEE bias, getting C-EXP. Go to step 20.
20. Round C-HI.C-LO. Go to step 21.
21. Pack C-EXP.M-SIGN, C-HI and C-LO and return.

(vi) FTOA.

The processing steps are as follows.

1. Unpack the input FLP number. Let the unpacked number be represented by C-EXP.C-SIGN, C-HI and C-LO. Go to step 2.
2. Subtract 126-decimal from C-EXP to remove the IEEE bias. Let the result be C1-EXP. Go to step 3.
3. Multiply C1-EXP by the binary representation of $\log_2(2)$. Let the product be U-HI.U-LO. Go to step 4.
4. Subtract 8 from U-HI.U-LO. Let the magnitude of the integer part of the result be V and its sign VSIGN. Go to step 5.
5. If VSIGN is 0, then set P-HI.P-LO to the binary representation of 0.8, else set P-HI.P-LO to the binary representation of 0.625. Go to step 6.
6. Multiply C-HI.C-LO by P-HI.P-LO V times. Normalize the partial product after each multiplication, if needed, by left shifting. Accumulate any left shifts needed in B1-EXP. Let the final product be HI-INT.LO-INT. Go to step 7.
7. Subtract B1-EXP from C1-EXP. Let the result be B2-EXP. Go to step 8.
8. If VSIGN is 0, then multiply V by -3, else multiply it by 4. Let the result be B3-EXP. Go to step 9.
9. Add B2-EXP and B3-EXP. Let the result be B4-EXP. Go to step 10.
10. If B4-EXP is more than 32-decimal, then increase V and go to step 6, else go to step 11.
11. If B4-EXP is less than 28-decimal, then decrease V and go to step 6, else go to step 12.
12. Subtract B4-EXP from 32. Let the result be B5-EXP. Go to step 13.
13. Shift HI-INT.LO-INT right B5-EXP number of times. Go to step 14.
14. Add 16-decimal to the address of the start of the decimal string. Output a null byte there. Go to step 15.
15. Divide V by 10-decimal. Let the quotient be Q and the remainder R. Add 30 to R and output it to the decimal string. Next add 30 to Q and output it to the decimal string. Go to step 16.
16. If VSIGN is 0, then output '+' to the output string, else output '-' to the output string. Go to step 17.
17. Output 'E' to the output string. Output '.' to the output string. Go to step 18.
18. Divide C-HI.C-LO by 10-decimal 10 times. Let the remainder in each division be R. Add 30 to each R and output it to the output string. Go to step 19.
19. If C-SIGN is 0, then output the space character to the output string, else output '-' to the output string. Then return to the calling program.

REFERENCES

1. ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, Aug. 12, 1985.
2. J.T. Coonen, "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic," IEEE Computer, Jan. 1980, pp. 68-79.
3. K. Hwang, *Computer Arithmetic*, John-Wiley and Sons, 1979.
4. M. M. Mano, *Computer System Design*, Prentice-Hall, 1980.

APPENDIX A**A USER'S MANUAL FOR THE HPC
FLOATING POINT PACKAGE**

The Single Precision Floating Point Package for the HPC implements the following functions.

ARITHMETIC FUNCTIONS

1. FADD—Add two floating point numbers.
2. FSUB—Subtract two floating point numbers.
3. FMULT—Multiply two floating point numbers.
4. FDIV—Divide two floating point numbers.

FORMAT CONVERSION FUNCTIONS

5. ATOF—Convert an ASCII string representing a decimal floating point number to a single precision floating point number.
6. FTOA—Convert a single precision floating point number to an ASCII string that represents the decimal floating point value of the number.

The entire package is in the form of a collection of subroutines and is contained in the following files.

1. FERR.MAC
2. FNACHK.MAC
3. FZCHK.MAC
4. FUNPAK.MAC
5. FPAK.MAC
6. FPTRAP.MAC
7. ROUND.MAC
8. BFMUL.MAC
9. ISIOK.MAC
10. MUL10.MAC
11. ATOF.MAC
12. FTOA.MAC
13. FADD.MAC
14. FMULT.MAC
15. FDIV.MAC

The first 7 files are general utility routines that are used by all the Arithmetic and Format Conversion subroutines. The next 3 files, BFMUL.MAC, ISIOK.MAC and MUL10.MAC are used only by the Format Conversion subroutines, ATOF and FTOA. Depending on the functions being used in the user program, only the necessary files need be included.

INTERFACE WITH USER PROGRAMS

1. All the Arithmetic routines expect the input to be in the IEEE Single Precision format. This format requires 2 words for the storage of each floating point number. If the required arithmetic operation is FlopF2, where op is +, -, *, or /, then the routines expect that F1 is available in registers K and A on entry, with the high half in K. Also, the two words of F2 are expected to be on the stack. If SP is the stack pointer on entry into one of the Arithmetic function subroutines, then the high word of F2 should be at W(SP-4) and the low word at W(SP-6). The result of the Arithmetic operation is returned in IEEE format in registers K and A, with the high word in K.

2. The Format Conversion subroutine ATOF expects that on entry, B contains the address of the ASCII string representing the decimal floating point number. This string must be of the form

Siiii.fffffEsNND

where

S is an optional sign for the mantissa. Thus S can be '+', '-' or not present at all.

iiii is the optional integer part of the mantissa. If it is present, it can be of any length, must contain only the characters '0' through '9' and must not contain any embedded blanks.

. is the optional decimal point. It need not be present if the number has no fractional part.

fffff is the optional fractional part of the mantissa. fffff is it is present must consist of a sequence of digits '0' through '9'. It can be of any length. Note that either iiiii, the integer part or .fffff the fractional part must be present.

E is the required exponent start symbol.

s is the optional sign of the exponent. If it is present, it must be '+' or '-'.

NN is the exponent and consists of at most two decimal digits. It is required to be present.

D is the null byte <00> and must be present to terminate the string.

The floating point number represented by the above string is returned by ATOF in IEEE format in registers K and A.

3. The format conversion routine FTOA expects the floating point number input to be in registers K and A in the IEEE format. Register B is expected to contain the starting address of a 17 byte portion of memory where the output string will be stored.
4. Three global symbols need to be defined in the user program before assembling the user program and any included floating point package files. These symbols are:
 - (i) TMP1 which must be set to 0. The package uses W(TMP1) for temporary storage.
 - (ii) FPERWD which must be set to an address in the base page. The package signals floating point exceptions using W(FPERWD). This is described below.
 - (iii) FPTRAP which must be set to the address of the start of a user floating point exception handler. Again this is described below.

FLOATING POINT EXCEPTS

The package maintains a history of floating point exceptions in the 4 least significant bits of the word W(FPERWD). The value of the symbol FPERWD should be defined by the user program, and should be an address in the base page. This word should also be cleared by the user program before calling any floating point routine. The word is never cleared by the floating point package, and the user program can examine this word to determine the type of exceptions that may have occurred during the course of a computation.

The following 4 types of error can occur in the course of a floating point computation.

1. **Invalid Operand.** This happens if one of the input numbers for an Arithmetic routine or the input for FTOA is not a valid floating point number. An invalid floating point number (or NaN) can be created either by an overflow in a previous computation step, or if the ASCII decimal floating point number input to ATOF is too large to be represented in the IEEE format. The result, if one of the inputs is a NaN is always set to a NaN.
2. **Overflow.** This happens if the result of a computation is too large to be represented within the exponent range available. Overflow can occur in any of the arithmetic routines or ATOF. On overflow, the result is set to a representation called NaN. An NaN is considered an illegal operand in all successive steps.
3. **Underflow.** This occurs if the result of a computation is too small to be represented with the precision and expo-

nent range available. On underflow, the result is set to zero.

4. **Divide-by-zero.** This error occurs if F2 is zero when computing $F1/F2$. The result is set to an NaN.

Each of the above errors results in a bit being set in W(FPERWD). This is done as follows:

Bit 0—Set on Overflow.

Bit 1—Set on Underflow.

Bit 2—Set on Illegal Operand.

Bit 3—Set on Divide-by-zero.

One further action is taken when a floating point exception occurs. After the result has been set to the appropriate value, and the corresponding bit in W(FPERWD) set, the package does a subroutine call to address FPTRAP. The user can provide any exception handler at this address. The file FPTRAP.MAC contains the simplest possible user exception handler. It does nothing, but merely returns back to the calling program.

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP

PAGE: 1

1		.TITLE FLP
2	LISTER:	
3	0071	.LIST 071
4	F000	. = 0F000
5	0002	FPERWD = W(2)
6	0000	TMP1 = W(0)

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
THE FLP ROUTINES

PAGE: 2

7 .FORM 'THE FLP ROUTINES'

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

```
8                               .FORM 'FERR.MAC'
9                               .INCLD FERR.MAC
1                                ; EXCEPTION HANDLING.
2                                ; DIVIDE BY ZERO.
3                                DIVBY0:
4 F000 820802FA                OR FPERWD, 08 ; SET THE DIVIDE BY 0 BIT.
5 F004 00                       CLR A
6 F005 B17F80                  LD K, 07F80
7 F008 3093                    JSR FPTRAP
8 F00R 3FCC                    POP B
9 F00C 3FCE                    POP X
10 F00E 3C                     RET
11                               ; ILLEGAL OPERAND - ONE OF F1 OR F2 IS A NAN.
12                               FNAN:
13 F00F 820402FA                OR FPERWD, 04 ; SET THE ILLEGAL OPERAND BIT.
14 F013 00                       CLR A
15 F014 B17F80                  LD K, 07F80 ; RETURN NAN IN K AND A.
16 F017 3084                    JSR FPTRAP ; GO TO USER TRAP ROUTINE.
17 F019 3FCC                    POP B
18 F01B 3FCE                    POP X
19 F01D 3C                     RET
20                               ; EXPONENT UNDERFLOW.
21                               UNDFL:
22 F01E 820202FA                OR FPERWD, 02 ; SET THE EXPONENT UNDERFLOW BIT.
23 F022 00                       CLR A
24 F023 ACC8CA                  LD K, A
25 F026 3075                    JSR FPTRAP
26 F028 3FC4                    POP SP
27 F02A 3FCC                    POP B
28 F02C 3FCE                    POP X
29 F02E 3C                     RET
30                               ; EXPONENT OVERFLOW.
31                               OVRF1:
32 F02F 820102FA                OR FPERWD, 01 ; SET THE EXPONENT OVERFLOW BIT.
33 F033 00                       CLR A
34 F034 B17F80                  LD K, 07F80
35 F037 3064                    JSR FPTRAP
36 F039 3FC4                    POP SP
37 F03B 3FCC                    POP B
38 F03D 3FCE                    POP X
39 F03F 3C                     RET
40                               ;
41                               .END
```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FLP
 FNACHK.MAC

PAGE: 4

```

10                                     .FORM 'FNACHK.MAC'
11                                     .INCLD FNACHK.MAC
1                                     .TITLE FNACHK
2                                     .LOCAL
3
4                                     ;
5                                     ; SUBROUTINE TO CHECK IF A SP FLOATING POINT NUMBER STORED IN THE
6                                     ; IEEE FLOATING POINT FORMAT IN REGS. K AND A IS NAN.
7                                     ;
8                                     ; RETURNS 0 IN C IF THE NUMBER IS NOT A NAN.
9                                     ; RETURNS 1 IN C IF THE NUMBER IS A NAN.
10                                    ; PRESERVES REGS. K, A, X AND B. DESTROYS C.
11                                    ;
12                                    FNACHK:
13 F040 AECA                            X A, K
14 F042 E7                               SHL A
15 F043 BDFEFF                           IFGT A, OFEFF
16 F046 45                               JP $ISNAN
17 F047 D7                               RRC A
18 F048 03                               RESET C
19 F049 AECA                            X A, K
20 F04B 3C                               RET
21                                    $ISNAN:
22 F04C D7                               RRC A
23 F04D 02                               SET C
24 F04E AECA                            X A, K
25 F050 3C                               RET
26
27                                     .END

```

```
12                                     .FORM 'FZCHK.MAC'  
13                                     .INCLD FZCHK.MAC  
1   .TITLE FZCHK  
2   .LOCAL  
3   ;  
4   ; SUBROUTINE THAT CHECKS IF A SP FLOATING POINT NUMBER STORED  
5   ; IN THE IEEE FORMAT IN REGS K AND A IS ZERO.  
6   ;  
7   ; RETURNS 0 IN C IF THE NUMBER IS NOT ZERO.  
8   ; RETURNS 1 IN C IF THE NUMBER IS ZERO.  
9   ; SAVES REGS. K, A, X, AND B BUT DESTROYS C.  
10  ;  
11  FZCHK:  
12 F051 AECA      X A, K  
13 F053 E7       SHL A  
14 F054 9DEF     IFGT A,OFF  
15 F056 45       JP $ANOTO  
16 F057 D7       RRC A  
17 F058 02       SET C  
18 F059 AECA     X A, K  
19 F05B 3C       RET  
20 $ANOTO:  
21 F05C D7       RRC A  
22 F05D 03       RESET C  
23 F05E AECA     X A, K  
24 F060 3C       RET  
25 ;  
26                                     .END
```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FZCHK
 FUNPAK.MAC

PAGE: 6

```

14          .FORM 'FUNPAK.MAC'
15          .INCLD FUNPAK.MAC
1          .TITLE FUNPAK
2          .LOCAL
3
4          ;
5          ; SUBROUTINE TO UNPACK A SP FLOATING POINT NUMBER STORED IN THE
6          ; IEEE FORMAT IN REGS. K AND A. THE UNPACKED FORMAT OCCUPIES 3
7          ; WORDS AND IS ORGANIZED AS FOLLOWS:
8          ;
9          ; increasing addr |      |      |      |      |      |      |      |      |      |
10         ;                |      |      |      |      |      |      |      |      |      |
11         ;                |EEEEEEEESSSSSSS| FEXP-FSIGN |      |      |      |      |      |      |
12         ;                |MMMMMMMMMMMMMMM| FHI         |      |      |      |      |      |      |
13         ;                |MMMMMMMOOOOOOOO| FLO         |      |      |      |      |      |      |
14         ;                |      |      |      |      |      |      |      |      |      |
15         ; EEEEEEE - 8 BIT EXPONENT IN EXCESS-127 FORMAT
16         ; SSSSSSS - SIGN BIT < 00 -> +, FF -> ->
17         ; M ... M - 24 BITS OF MANTISSA. NOTE THAT IMPLIED 1 IS PRESENT HERE.
18         ;
19         ; ON ENTRY TO THE SUBROUTINE X SHOULD POINT TO FLO. ON EXIT, X POINTS
20         ; TO THE WORD AFTER FSIGN.
21         ; REGS. K, A AND B ARE DESTROYED BY THIS SUBROUTINE.
22         ;
23         FUNPAK:
24         ST A,B          ; SAVE A IN B.
25         CLR A
26         X A, M(X+)      ; ZERO LOW BYTE OF FLO.
27         LD A, L(B)
28         X A, M(X+)      ; MOVE LOW BYTE OF F-RO INTO HIGH BYTE OF FLO.
29         LD A, H(B)
30         X A, M(X+)      ; MOVE MID BYTE OF MANT INTO LOW BYTE OF FHI.
31         LD A, K
32         SET A.7         ; SET IMPLIED 1 IN MANTISSA
33         X A, M(X+)      ; MOVE HIGH BYTE OF MANT INTO HIGH BYTE OF FHI.
34         LD A, K
35         SHL A           ; SIGN BIT TO CARRY.
36         AND A, OFF00    ; ZERO SIGN.
37         IF C
38         OR A, OFF       ; PUT SIGN BACK IF -.
39         X A, W(X+)      ; SAVE FEXP-FSIGN.
40         RET
41         ;
42         .END

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FUNPAK
FPAK.MAC

PAGE: 7

```

16                                     .FORM 'FPAK.MAC'
17                                     .INCLD FPAK.MAC
1   .TITLE FPAK
2   .LOCAL
3   ;
4   ; SUBROUTINE TO PACK A SF FLOATING POINT NUMBER STORED IN THE
5   ; 3 WORD FEXP-FSIGN/FHI/FLO FORMAT INTO THE IEEE FORMAT IN REGS.
6   ; K AND A.
7   ;
8   ; ON ENTRY TO THE SUBROUTINE, X POINTS TO FLO. ON EXIT, X POINTS
9   ; TO THE WORD AFTER FSIGN.
10  ;
11  ; REGS. K, A AND B ARE DESTROYED.
12  ;
13  FPAK:
14  F07C D1      X A, M(X+)      ; GET RID OF ZERO LOW BYTE OF FLO.
15  F07D D1      X A, M(X+)      ; GET HIGH BYTE OF FLO.
16  F07E ABCA    ST A, K         ; STORE IT IN K.
17  F080 D1      X A, M(X+)      ; GET LOW BYTE OF FHI.
18  F081 3B     SWAP A
19  F082 3B     SWAP A
20  F083 B9FF00  AND A, OFF00    ; SHIFT LEFT 8 TIMES.
21  F086 A0C8CAFA OR K, A      ; LOW WORD OF RESULT IS NOW IN K.
22  ;
23  F08A D1      X A, M(X+)      ; GET HIGH BYTE OF FHI.
24  F08B 96C81F  RESET A.7      ; ZERO IMPLIED MSB 1 IN MANT.
25  F08E ABCC    ST A,B         ; SAVE IN REG. B.
26  F090 D4      LD A, M(X)      ; GET SIGN BYTE FROM ASIGN.
27  F091 C7     SHR A           ; MOVE 1 SIGN BIT INTO CARRY.
28  F092 F0     LD A, W(X+)      ; GET FEXP-FSIGN.
29  F093 B9FF00  AND A, OFF00    ; ZERO SIGN.
30  F096 D7     RRC A           ; MOVE RIGHT 1 BIT. SIGN BIT FROM C
31  ;           ; ENTERS INTO THE MSB.
32  F097 96CCFA  OR A, B        ; GET MANT BITS IN FROM B.
33  F09A AECA    X A, K         ; SWAP A AND K
34  F09C 3C     RET
35  ;
36                                     .END

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPAK
FPTRAP.MAC

PAGE: 8

```

18                                     .FORM 'FPTRAP.MAC'
19                                     .INCLD FPTRAP.MAC
1   .TITLE FPTRAP
2   ; USER SUPPLIED FP TRAP ROUTINE.
3   FPTRAP:
4   F09D 3C     RET
5   ;
6                                     .END

```


NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 FPTRAP
 ROUND.MAC

PAGE: 9

```

20                                     .FORM 'ROUND.MAC'
21                                     .INCLD ROUND.MAC
1                                       .TITLE SROUND
2                                       .LOCAL
3
4   ; THIS SUBROUTINE IS USED TO ROUND THE 32 BIT MANTISSA OBTAINED
5   ; IN THE FLOATING POINT CALCULATIONS TO 24 BITS.
6
7   ; THE UNPACKED FLOATING POINT NUMBER SHOULD BE STORED IN
8   ; CONSECUTIVE WORDS OF MEMORY. ON ENTRY, X SHOULD CONTAIN
9   ; THE ADDRESS OF C-HI. C-EXP.C-SIGN IS AT W(X+2) AND
10  ; C-LO IS AT W(X-2).
11
12  ; ON EXIT X HAS THE ADDRESS OF C-EXP.C-SIGN.
13  SROUND:
14  F09E F2          LD A, W(X-)      ; REMEMBER X POINTS TO C-HI.
15  F09F F4          LDA, W(X)        ; LOAD C-LO.
16  FOA0 96C817     IF A.7           ; IF BIT 25 OF MANTISSA IS 1,
17  FOA3 43          JP $RNDUP       ; THEN NEED TO INCREASE MANTISSA.
18  FOA4 FO          LD A, W(X+)
19  FOA5 FO          LD A, W(X+)     ; X NOW POINTS TO C-EXP.C-SIGN.
20  FOA6 5F          JP $EXIT        ; DONE, SO GET OUT.
21  ; INCREASE MANTISSA.
22  $RNDUP:
23  FOA7 B80100     ADD A, 0100
24  FOAA F1          X A, W(X+)      ; INCREASE LOW BYTE BY 1.
25  FOAB 07          IF C            ; IF THERE IS A CARRY,
26  FOAC 42          JP $HIUP        ; THEN NEED TO INCREASE C-HI.
27  FOAD FO          LD A, W(X+)     ; X NOW POINTS TO C-EXP.C-SIGN.
28  FOAE 57          JP $EXIT        ; DONE, SO GET OUT.
29  ; MANTISSA INCREASE PROPAGATING TO HIGH WORD.
30  $HIUP:
31  FOAF F4          LD A, W(X)
32  FOBO B8          .BYTE 0B8,00,01 ; DO ADD A, 01 BUT WITH WORD CARRY!!
   FOB1 00
   FOB2 01
33  FOB3 07          IF C            ; IF THERE IS A CARRY,
34  FOB4 42          JP $EXIN2       ; THEN NEED TO INCREASE EXPONENT.
35  FOB5 F1          X A,W(X+)
36  FOB6 4F          JP $EXIT        ; GET OUT.
37  ; ROUND UP LEADS TO EXPONENT INCREASE.
38  $EXIN2:
39  FOB7 D7          RRC A           ; CARRY->MSB, LSB->CARRY.
40  FOB8 F3          X A,W(X-)
41  FOB9 F4          LD A,W(X)       ; LOW WORD IS NOW IN A.
42  FOBA D7          RRC A
43  FOBB F1          X A, W(X+)
44  FOBC FO          LD A, W(X+)     ; X NOW POINTS TO C-EXP.CSIGN.
45  FOBD F4          LD A, W(X)
46  FOBE B80100     ADD A, 0100
47  FOC1 07          IF C

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
ROUND.MAC

PAGE: 10

```
48 FOC2 BAFF00          OR A, OFF00      ; MAKE IT A NAN.  
49 FOC5 F6             ST A, W(X)  
50                      ;  
51                      $EXIT:  
52 FOC6 3C             RET  
53                      ;  
54                      .END
```

AN-486

5

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 SROUND
 BFMUL.MAC

PAGE: 11

```

22          .FORM 'BFMUL.MAC'
23          .INCLD BFMUL.MAC
  1          .TITLE BFMUL
  2          ;
  3          ; THIS SUBROUTINE IS USED TO MULTIPLY TWO 32 BIT FIXED POINT FRACTIONS.
  4          ; THE ASSUMED BINARY POINT IS TO THE IMMEDIATE LEFT OF THE MSB.
  5          ;
  6          ; THE FIRST FRACTION IS STORED IN REGS K AND A, WITH THE MORE
  7          ; SIGNIFICANT WORD BEING IN K.
  8          ;
  9          ; THE SECOND FRACTION IS STORED ON THE STACK. THE MORE SIGNIFICANT
10          ; WORD IS AT W(SP-4) AND THE LOWER SIGNIFICANT WORD
11          ; IS IN THE WORD BELOW IT.
12          ;
13          ; THE 32 BIT PRODUCT IS LEFT IN REGS. K AND A, WITH THE MORE
14          ; SIGNIFICANT WORD BEING IN K.
15          ;
16          ; IMPORTANT NOTE : THE FRACTIONS ARE ASSUMED TO BE UNSIGNED.
17          ;
18          ; REGS. B AND X ARE UNCHANGED.
19          ;
20          BFMUL:
21 FOC7 AFCE          PUSH X          ; SAVE X.
22 FOC9 AFC8          PUSH A          ; SAVE F1-LO
23 FOCB AFCA          PUSH K          ; SAVE F1-HI.
24 FOCB ABCA          LD A, K         ; MOVE F1-HI TO A.
25 FOCF AGFFF6C4FE    MULT A, W(SP-OA); MULTIPLY F1-HI BY F2-HI.
26 FOD4 3FCA          POP K          ; GET F1-HI.
27 FOD6 AFCE          PUSH X          ; SAVE PR-HI.
28 FOD8 AFC8          PUSH A          ; SAVE PR-LO.
29 FODA ABCA          LD A, K         ; MOVE F1-HI TO A.
30 FODC AGFFF2C4FE    MULT A, W(SP-OE); MULTIPLY F1-HI BY F2-LO.
31 FOE1 3FC8          POP A          ; GET PR-LO SAVED. NOTE THAT THE
32          ; LO WORD OF THIS PRODUCT IS DISCARDED.
33 FOE3 3FCA          POP K          ; GET PR-HI SAVED.
34 FOE5 96CEF8        ADD A, X         ; ADD TO PR-LO THE HI WORD OF THIS PRODUCT.
35 FOE8 07           IF C           ; ON CARRY,
36 FOE9 A9CA          INC K          ; PROPAGATE THRU TO PR-HI.
37 FOEB 3FCE          POP X          ; GET F1-LO.
38 FOED AFCA          PUSH K          ; SAVE PR-HI.
39 FOEF AFC8          PUSH A          ; SAVE PR-LO.
40 FOF1 ABCE          LD A, X         ; MOVE F1-LO TO A.
41 FOF3 AGFFF6C4FE    MULT A, W(SP-OA); MULTIPLY BY F2-HI.
42 FOF8 3FC8          POP A          ; GET PR-LO SAVED.
43 FOFA 3FCA          POP K          ; GET PR-HI SAVED.
44 FOFB 96CEF8        ADD A, X         ; ADD TO PR-LO THE HI-WORD OF THIS PRODUCT.
45 FOFF 07           IF C           ;
46 F100 A9CA          INC K          ; PROPAGATE ANY CARRY TO PR-HI.
47 F102 3FCE          POP X          ; RESTORE X.
48 F104 3C           RET
49          ;

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 BFMUL
 BFMUL.MAC

PAGE: 12

50 .END

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 BFMUL
 ISIOK.MAC

PAGE: 13

```

24 .FORM 'ISIOK.MAC'
25 .INCLD ISIOK.MAC
 1 .TITLE ISIOK
 2 .LOCAL
 3 ;
 4 ; THIS SUBROUTINE IS USED TO DETERMINE IF ANOTHER DECIMAL DIGIT CAN
 5 ; BE ACCUMULATED IN THE 32 BIT INTEGER STORED IN REGS. K AND A.
 6 ; THE MORE SIGNIFICANT WORD IS IN K.
 7 ; SETS THE CARRY TO 1 IF IT CAN BE ACCUMULATED; RESETS THE CARRY
 8 ; OTHERWISE. PRESERVES ALL REGS.
 9 ;
10 ISIOK:
11 F105 02 SET C
12 F106 861999CAFC IFEQ K, 01999
13 F10B 47 JP $CHKOT
14 F10C 861999CAFD IFGT K, 01999
15 F111 03 RESET C
16 F112 3C RET
17 F113 BD9998 $CHKOT: IFGT A, 09998
18 F116 03 RESET C
19 F117 3C RET
20 ;
21 .END

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 ISIOK
 MULIO.MAC

PAGE: 14

```

26                                     .FORM 'MULIO.MAC'
27                                     .INCLD MULIO.MAC
   1                                     .TITLE MULIO
   2                                     .LOCAL
   3                                     ;
   4                                     ; THIS SUBROUTINE MULTIPLIES THE 32 BIT INTEGER STORED IN REGS K AND A
   5                                     ; BY 10-DECIMAL AND ADDS TO IT THE INTEGER STORED IN X.
   6                                     ; THE RESULT IS RETURNED IN K AND A.
   7                                     ; REGS. B AND X ARE NOT CHANGED.
   8                                     ;
   9      MULIO:
10 F118 AFCE                          PUSH X          ; SAVE INTEGER.
11 F11A AFC8                          PUSH A          ; SAVE LONG INT-LO.
12 F11C A8CA                          LD A, K
13 F11E 9EOA                          MULT A, OA      ; MULT LONG INT-HI BY 10.
14 F120 AFC8                          PUSH A          ; SAVE LOW WORD OF PRODUCT.
15 F122 A6FFFCC4A8                    LD A, W(SP-4)  ; GET LONG INT-LO.
16 F127 9EOA                          MULT A, OA
17 F129 3FCA                          POP K          ; GET LO WORD OF LAST PRODUCT.
18 F12B AOCECAF8                      ADD K, X       ; ADD TO IT HI WORD OF THIS PRODUCT.
19 F12F 3FCE                          POP X          ; GET RID OF GARBAGE.
20 F131 3FCE                          POP X          ; GET INTEGER TO BE ADDED.
21 F133 96CEF8                      ADD A, X
22 F136 07                            IF C
23 F137 A9CA                          INC K
24 F139 3C                            RET
25                                     ;
26                                     .END
28                                     ;

```

```

29          .FORM 'ATOF.MAC'
30          .INCLD ATOF.MAC
1           .TITLE ATOF
2           .LOCAL
3           ;
4           ; THIS SUBROUTINE CONVERTS A DECIMAL FLOATING POINT STRING TO
5           ; AN IEEE FORMAT SINGLE PRECISION FLOATING POINT NUMBER. THE
6           ; INPUT DECIMAL STRING IS ASSUMED TO BE OF THE FORM
7           ; SMMMMMM.FFFFFFFDNN
8           ; WHERE S IS THE SIGN OF THE DECIMAL MANTISSA,
9           ; M...M IS THE INTEGER PART OF THE MANTISSA,
10          ; F...F IS THE FRACTIONAL PART OF THE MANTISSA,
11          ; D IS THE SIGN OF THE DECIMAL EXPONENT,
12          ; AND NNN IS THE DECIMAL EXPONENT.
13          ;
14          ; ON ENTRY, B SHOULD POINT TO THE ADDRESS OF THE ASCII
15          ; STRING HOLDING THE DECIMAL FLOATING POINT NUMBER. THIS STRING
16          ; MUST BE TERMINATED BY A NULL BYTE.
17          ;
18          ; THE BINARY FLOATING POINT NUMBER IS RETURNED IN
19          ; REGS. K AND A.
20          ;
21          ; REGS. B AND X ARE LEFT UNCHANGED.
22          ;
23          ;
24          ;
25          ;
26          ;
27          ATOF:
28 F13A AFCE          PUSH X
29 F13C AFCC          PUSH B
30 F13E 00            CLR A          ; ZERO A.
31 F13F AFC8          PUSH A          ; STORAGE FOR MANTISSA SIGN.
32 F141 AFC8          PUSH A          ; STORAGE FOR IMPLICIT 10'S EXPONENT.
33 F143 AFC8          PUSH A          ; STORAGE FOR HI-INT.
34 F145 AFC8          PUSH A          ; STORAGE FOR LO-INT.
35          ;
36          ; DECIMAL STRING MUST START WITH A '+', '-', '.' OR A DIGIT.
37          ; RESULTS ARE UNPREDICTABLE IF IT DOES NOT.
38          ; THE '+' MEANS THAT THE MANTISSA IS POSITIVE. IT CAN BE OMITTED.
39          ; THE '-' MEANS THAT THE MANTISSA IS NEGATIVE.
40          ; THE '.' MEANS THAT THE MANTISSA HAS NO INTEGER PART.
41          ;
42          $LOOP1:
43 F147 C0            LDS A, M(B+)
44 F148 40            NOP          ; GET THE CHARACTER.
45 F149 9C2B          IFEQ A, '+'    ; IF IT IS A '+',
46 F14B 64            JP $LOOP1    ; DO NOTHING, BUT GET 1 MORE.
47 F14C 9C2D          IFEQ A, '-'    ; IF IT IS A '-',
48 F14E 45            JP $MSIGN    ; GO AND CHANGE THE MANTISSA SIGN.
49 F14F 9C2E          IFEQ A, '.'    ; IF IT IS A '.',

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 ATOF
 ATOF.MAC

PAGE: 16

```

50 F151 9438          JMP $FRCOL          ; GO AND COLLECT THE FRACTION PART.
51                                     ; GET HERE MEANS IT IS A DIGIT.
52 F153 48            JP $INCOL          ; SO GO AND COLLECT THE INTEGER PART.
53          $MSIGN:
54 F154 90FF          LD A, OFF
55 F156 A6FFF8C4AB     ST A, W(SP-08)      ; CHANGE MANTISSA SIGN TO NEG.
56 F15B 74            JP $LOOP1         ; GO BACK FOR SOME MORE.
57                                     ;
58          $INCOL:
59          ; GET HERE MEANS COLLECTING INTEGER PART OF MANTISSA.
60                                     ;
61 F15C 02            SET C
62 F15D 8230C8EB       SUBC A, '0'          ; CONVERT DIGIT FROM ASCII TO INTEGER.
63 F161 ACC8CE        LD X, A              ; MOVE INTEGER TO X.
64 F164 3FCA          POP K              ; GET HI-INT COLLECTED SO FAR.
65 F166 3FC8          POP A              ; GET LO-INT COLLECTED SO FAR.
66 F168 3463          JSR ISIOK          ; CHECK IF THE DIGIT CAN BE ACCUMULATED.
67 F16A 07            IF C              ; LOOK AT C.
68 F16B 4B            JP $ACCM          ; YES, IT CAN BE SO GO DO IT.
69                                     ; GET HERE MEANS CAN ACCUMULATE ANY MORE.
70                                     ; SO INCREASE THE IMPLICIT 10'S EXPONENT.
71 F16C 3FCE          POP X
72 F16E A9CE          INC X              ; SO FAR AND INCREMENT IT.
73 F170 AFCE          PUSH X             ; SAVE IT BACK.
74 F172 AFC8          PUSH A             ; SAVE LO-INT.
75 F174 AFCA          PUSH K             ; SAVE HI-INT.
76 F176 46            JP $ISNXT
77                                     ;
78          $ACCM:
79          ; GET HERE MEANS THE PRESENT DIGIT CAN BE ACCUMULATED.
80 F177 345F          JSR MULLO          ; MULTIPLY BY 10 AND ADD DIGIT.
81 F179 AFC8          PUSH A             ; SAVE LO-INT.
82 F17B AFCA          PUSH K             ; SAVE HI-INT.
83          $ISNXT:
84          ; PROCESS THE NEXT CHARACTER.
85 F17D C0            LDS A, M(B+)
86 F17E 40            NOP
87 F17F 9C2E          IFEQ A, '.'          ; IF IT IS A '.'
88 F181 49            JP $FRCOL          ; GO COLLECT FRACTION PART.
89 F182 9C45          IFEQ A, 'E'          ; IF IT IS 'E',
90 F184 9434          JMP $EXCOL          ; GO COLLECT EXPONENT PART.
91 F186 9C20          IFEQ A, ' '          ; IF IT IS A SPACE,
92 F188 6B            JP $ISNXT          ; GO GET SOME MORE.
93                                     ; GET HERE MEANS IT IS A DIGIT.
94 F189 952D          JMP $INCOL
95                                     ;
96          $FRCOL:
97          ; GET HERE MEANS COLLECT THE FRACTIONAL PART OF THE MANTISSA.
98                                     ;
99 F18B C0            LDS A, M(B+)
100 F18C 40           NOP

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

```

101 F18D 9C45      IFEQ A, 'E'      ; IF IT IS A 'E',
102 F18F 9429      JMP $EXCOL       ; GO COLLECT EXPONENT.
103 F191 9C20      IFEQ A, ' '     ; IF IT IS SPACE,
104 F193 68        JP $FRCOL        ; GO GET SOME MORE.
105                ; GET HERE MEANS IT IS A DIGIT.
106 F194 D2        SET C
107 F195 8230C8EB  SUBC A, '0'     ; GET INTEGER FROM DIGIT.
108 F199 ACC8CE    LD X, A         ; SAVE IT IN A.
109 F19C 3FCA      POP K          ; GET HI-INT.
110 F19E EFC8      POP A          ; GET LO-INT.
111 F1A0 349B      JSR ISIOK      ; CHECK IF IT CAN BE ACCUMULATED.
112 F1A2 07        IF C
113 F1A3 45        JP $ACCF      ; YES, SO GO DO IT.
114                ; GET HERE MEANS CAN'T COLLECT MORE DIGITS.
115 F1A4 AFC8      PUSH A
116 F1A6 AFCA      PUSH K
117 F1A8 7D        JP $FRCOL      ; SO JUST IGNORE IT.
118                ;
119                $ACCF:
120                ; ACCUMULATE THE FRACTIONAL DIGIT.
121 F1A9 3491      JSR MUL10     ; MULTIPLY BY 10 AND ADD DIGIT.
122 F1AB 3FCE      POP X          ; GET IMPLICIT 10'S EXPONENT COLLECTED SO FAR,
123 F1AD 86FFFFCEFB ADD X, OFFFF    ; AND DECREMENT IT BY 1.
124 F1B2 AFCE      PUSH X          ; SAVE IT BACK.
125 F1B4 AFC8      PUSH A          ; SAVE LO-INT.
126 F1B6 AFCA      PUSH K          ; SAVE HI-INT.
127 F1B8 952D      JMP $FRCOL     ; GO GET SOME MORE.
128                ;
128                $EXCOL:
130                ; GET HERE MEANS THE EXPLICIT 10'S EXPONENT NEEDS TO BE
131                ; COLLECTED FROM THE STRING.
132 F1BA 03        RESET C        ; MAKE EXPONENT SIGN POST.
133                $EXCHR:
134 F1BB C0        LDS A, M(B+)
135 F1BC 40        MOP
136 F1BD 9C2B      IFEQ A, '+'     ; IF IT IS A '+',
137 F1BF 64        JP $EXCHR     ; GET SOME MORE.
138 F1C0 9C2D      IFEQ A, '-'     ; IF IT IS A '-',
139 F1C2 44        JP $ESIGN     ; GO FIX EXPONENT SIGN.
140 F1C3 9C20      IFEQ A, ' '     ; IF IT IS SPACE,
141 F1C5 6A        JP $EXCHR     ; GO GET SOME MORE.
142                ; GET HERE MEANS IT IS A DIGIT.
143 F1C6 42        JP $EXACC     ; SO GO COLLECT THE EXPONENT.
144                $ESIGN:
145 F1C7 02        SET C
146 F1C8 6D        JP $EXCHR
147                ;
148                $EXACC:
149                ; ACCUMULATE THE EXPLICIT 10'S EXPONENT.
150 F1C9 9100      LD K, 0
151 F1CB 07        IF C

```


NATIONAL SEMICONDUCTOR CORPORATION
 HFC CROSS ASSEMBLER, REV:C, 30 JUL 86
 ATOF
 ATOF.MAC

PAGE: 18

```

152 F1CC 91FF          LD K, OFF          ; GET SIGN BITS SET.
153 F1CE AFCA          PUSH K           ; SAVE EXPLICIT EXPONENTS SIGN.
154 F1D0 9300          LD X, 0
155 F1D2 AFCE          PUSH X           ; ZERO EXPLICIT EXPONENT COLLECTED SO FAR.
156                   $EXCLP:
157 F1D4 02            SET C
158 F1D5 8230C8EB      SUBC A, '0'      ; GET INTEGER FROM ASCII DIGIT.
159 F1D9 ACC8CE        LD X, A
160 F1DC 3FC8          POP A           ; GET EXPLICIT EXPONENT COLLECTED SO FAR.
161 F1DE AOC8CEFB      ADD X, A        ; X CONTAINS DIGIT + EXP.
162 F1E2 AOC8CEFB      ADD X, A        ; X CONTAINS DIGIT + 2*EXP.
163 F1E6 E7           SHL A
164 F1E7 E7           SHL A
165 F1E8 E7           SHL A          ; A CONTAINS 8*EXP.
166 F1E9 96CEFB       ADD A, X        ; A CONTAINS DIGIT + 10*EXP.
167 F1EC AFCE         PUSH A          ; SAVE BACK ON STACK.
168 F1EE CO           LDS A, M(B+)
169 F1EF 40           NOP            ; GET NEXT CHAR.
170 F1F0 9C00         IFEQ A, 0       ; IS IT A NULL ?
171 F1F2 41           JP $A10EX      ; YES SO ADD EXPLICIT AND IMPLICIT
172                   ; 10'S EXPONENT.
173                   ; GET HERE MEANS IT IS A DIGIT,
174 F1F3 7F           JP $EXCLP      ; SO GO BACK AND ACCUMULATE IT.
175                   ;
176                   $A10EX:
177                   ; DONE COLLECTING DIGITS. ADD THE EXPLICIT AND IMPLICIT
178                   ; 10'S EXPONENT COLLECTED SO FAR.
179 F1F4 3FC8         POP A           ; GET EXPLICIT 10'S EXPONENT.
180 F1F6 3FCA         POP K           ; GET ITS SIGN.
181 F1F8 8200CAFC     IFEQ K, 0       ; IS IT POSITIVE ?
182 F1FC 42           JP $ADDEX      ; YES SO ADD 'EM.
183 F1FD 01           COMP A
184 F1FE 04           INC A          ; CHANGE TO 2'S COM.
185                   $ADDEX:
186 F1FF AGFFFAC4F8   ADD A, W(SP-06) ; ADD IMPLICIT EXPONENT.
187 F204 BD7FFF       IFGT A, 07FFF   ; IS IT NEGATIVE ?
188 F207 43           JP $NEG10      ; YES, CHANGE IT.
189 F208 9300         LD X, 0        ; LOAD POST. SIGN IN X.
190 F20A 44           JP $ESAVE
191                   $NEG10:
192 F20B 93FF         LD, X, OFF     ; LOAD NEG. SIGN IN X.
193 F20D 01           COMP A
194 F20E 04           INC A          ; MAKE IT POSITIVE.
195                   $ESAVE:
196 F20F ACC8CC       LD B, A        ; SAVE 10'S EXPONENT IN A.
197 F212 3FC8         POP A          ; GET HI-INT.
198 F214 3FCA         POP K          ; GET LO-INT.
199 F216 AFCE         PUSH X         ; SAVE SIGN OF 10'S EXPONENT.
200 F218 AFCE         PUSH B         ; AND ITS VALUE.
201                   ;
202                   ; NOW CONVERT HI-INT.LO-INT TO A NORMALIZED FLOATING POINT

```

```

203 ; NUMBER. THE BINARY EXPONENT IS COLLECTED IN B.
204 ;
205 F21A 9000 IFGT A, 0 ; IF HI-INT IS NOT 0,
206 F21C 58 JP $NORM2 ; NEED TO SHIFT K AND A.
207 F21D 8200CAFD IFGT K, 0 ; IF HI-INT IS 0, BUT NOT LO-INT,
208 F221 4E JP $NORM1 ; NEED TO SHIFT ONLY K.
209 ; GET HERE MEANS MANTISSA IS 0.
210 F222 00 CLR A
211 F223 ACC8CA LD K, A
212 F226 02 SET C
213 F227 8208C4EB SUB SP, 08 ; ADJUST SP. DONE!!!
214 F22B 3FCC POP B
215 F22D 3FCE POP X
216 F22F 3C RET
217 ;
218 $NORM1:
219 ; HI-INT IS 0, SO WORK WITH LO-INT ONLY.
220 F230 AECA X A, K
221 F232 9210 LD B, 010 ; LOAD 16 INTO EXPONENT COUNTER.
222 F234 42 JP $NRLUP
223 ;
224 $NORM2:
225 ; HI-INT IS NOT 0, SO NEED TO HANDLE BOTH.
226 F235 9220 LD B, 020 ; LOAD 32 INTO LOOP COUNTER.
227 $NRLUP:
228 F237 E7 SHL A
229 F238 07 IF C ; DID A 1 COME OUT ?
230 F239 4D JP $NRDUN ; YES IT IS NORMALIZED NOW.
231 F23A AECA X A, K
232 F23C E7 SHL A
233 F23D 07 IF C
234 F23E 96CA08 SET K.0
235 F241 AECA X A, K
236 F243 AACC DECSZ B
237 F245 40 NOP ; SHOULD NEVER BE SKIPPED!!
238 F246 6F JP $NRLUP
239 $NRDUN:
240 F247 D7 RRC A ; RESTORE SHIFTED 1.
241 F248 AB00 ST A, TMP1 ; STORE IN W(0).
242 F24A 3FCE POP X ; GET 10'S EXPONENT.
243 F24C 3FC8 POP A ; GET 10'S EXPONENT SIGN.
244 F24E AE00 X A, TMP1 ; A IS HI-INT ONCE MORE.
245 F250 AFCC PUSH B ; SAVE BINARY EXPONENT.
246 F252 AFCE PUSH X ; SAVE 10'S EXPONENT.
247 F254 AECA X A, K ; HI-INT TO K, LO-INT TO A.
248 F256 960010 IF TMP1.0 ; IS 10'S EXPONENT NEGATIVE ?
249 F259 58 JP $DIV10 ; YES, GO TO DIVIDE BY 10.
250 ; GET HERE MEANS 10'S EXPONENT IS POSITIVE, SO MULTIPLY BY 10.
251 ; ACTUALLY, WHAT IS USED IS
252 ; 10^N = (0.625*(2^4))^N
253 ; SO MULTIPLY BY 0.625 NOW AND TAKE CARE OF 2^(4*N) LATER.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 ATOF
 ATOF.MAC

PAGE: 20

```

254 F25A A4F26ACCB          LD B, W($MTLO)
255 F25F AFCC                PUSH B          ; SAVE LO WORD OF 0.625 ON STACK.
256 F261 A4F26CCCB          LD B, W($MTHI)
257 F266 AFCC                PUSH B          ; SAVE HI WORD OF 0.625 ON STACK.
258 F268 57                  JP $JAMIT      ; GO TO ROUTINE THAT JAMS 0.625^N
259                          ; BY REPEATED MULTIPLICATION INTO HI-INT.LO-INT.
260                          ;
261                          ; DEFINE SOME CONSTANTS.
262 F269 40                  . EVEN ; FORCE EVEN ADDRESS.
263 F26A 0000                $MTLO: .WORD 0
264 F26C 0A00                $MTHI: .WORD 0A000
265 F26E CDCC                $DTLO: .WORD 0CCCD
266 F270 CCCC                $DTHI: .WORD 0CCCC
267                          ;
268                          $DIV10:
269                          ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO DIVIDE BY 10.
270                          ; ACTUALLY WHAT IS DONE IS
271                          ;  $10^{-N} = ((2^3)/(0.8))^{-N} = ((0.8)^N) * (2^{(-3*N)})$ 
272                          ; SO MULTIPLY BY 0.8 NOW AND TAKE CARE OF  $2^{(-3*N)}$  LATER.
273 F272 A4F26ECCB          LD B, W($DTLO)
274 F277 AFCC                PUSH B          ; SAVE LO WORD OF .8
275 F279 A4F270CCB          LD B, W($DTHI)
276 F27E AFCC                PUSH B          ; SAVE HI WORD OF .8
277                          ;
278                          $JAMIT:
279                          ; JAM IN THE MULTIPLICATION PART NEEDED TO HANDLE THE 10'S EXP.
280 F280 9200                LD B, 0         ; B IS USED TO TRACK ANY BINARY POWERS
281                          ; THAT COME UP DURING NORMALIZATION.
282 F282 8200CEFC            IFEQ X, 0       ; IS 10'S EXPONENT 0 ?
283 F286 57                  JP $JAMDN      ; YES, DONE ALREADY.
284                          $JAMPLP:
285 F287 35C0                JSR BFMUL      ; MULTIPLY USING 32 BIT UNSIGNED.
286 F289 AECA                X A, K         ; SWAP HI AND LO WORDS.
287 F28B E7                  SHL A
288 F28C 07                  IF C           ; IS THERE A CARRY ?
289 F28D 4A                  JP $ISNED     ; YES, SO IT IS ALREADY NORMALIZED.
290 F28E A9CC                INC B         ; NEED TO SHIFT LEFT TO NORMALIZE, SO
291                          ; INCREASE B BY 1.
292 F290 AECA                X A, K
293 F292 E7                  SHL A
294 F293 07                  IS C
295 F294 96CA08              SET K.0
296 F297 43                  JP $OVR1
297                          $ISNED:
298 F298 D7                  RRC A
299 F299 AECA                X A, K
300                          $OVR1:
301 F29B AACE                DECSZ X       ; DONE YET ?
302 F29D 76                  JP $JAMPLP   ; NO SO DO IT ONCE MORE.
303                          ; GET HERE MEANS MULTIPLICATIONS HAVE BEEN DONE. NOW TAKE
304                          ; CARE OF THE EXPONENTS.

```

```

305          $JAMDN:
306 F29E 3FCE          POP X
307 F2A0 3FCE          POP X          ; GET 0.625 OR 0.8 OFF THE STACK.
308 F2A2 3FCE          POP X          ; GET THE 10'S EXPONENT.
309 F2A4 AFC8          PUSH A          ; SAVE LO WORD OF FLP NUMBER.
310 F2A6 AFCA          PUSH K          ; SAVE HI WORD OF FLP NUMBER.
311 F2A8 A6FFAC4A8     LD A, W(SP-6) ; GET THE BINARY EXPONENT THAT WAS SAVED.
312 F2AD 02            SET C
313 F2AE 96CCEB        SUBC A, B          ; SUBTRACT FROM IT BINARY EXPONENT COLLECTED
314                                ; DURING THE JAMMING.
315 F2B1 ACC8CC        LD B, A          ; SAVE IT IN B.
316 F2B4 A8CE          LD A, X          ; MOVE THE 10'S EXPONENT TO A.
317 F2B6 960010        IF TMPL.0          ; IS THE 10'S EXPONENT NEGATIVE ?
318 F2B9 49            JP $NAGAS          ; YES, SO GOT TO SUBTRACT.
319                                ; GET HERE MEANS 10'S EXPONENT IS
320                                ; POSITIVE, SO MUL IT BY 4.
321 F2BA E7            SHL A          ; MULTIPLY BY 2.
322 F2BB E7            SHL A          ; MULTIPLY BY 2 AGAIN.
323 F2BC 96CCF8        ADD A, B          ; GET THE BINARY EXPONENT IN ALSO.
324 F2BF B8007E        ADD A, 07E          ; AND THE IEEE BIAS.
325 F2C2 4C            JP $EXCPT          ; GO CHECK FOR OVER/UNDERFLOW.
326                                ;
327          $NAGAS:
328          ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO GOT TO MULTIPLY
329          ; IT BY -3.
330 F2C3 E7            SHL A          ; MULTIPLY BY 2.
331 F2C4 96CEF8        ADD A, X          ; ADD TO GIVE MULTIPLY BY 3.
332 F2C7 01            COMP A
333 F2C8 04            INC A          ; MAKE IT NEGATIVE.
334 F2C9 96CCF8        ADD A, B          ; GET IN THE BINARY EXPONENT.
335 F2CC B8007E        ADD A, 07E          ; AND THE IEEE BIAS.
336          $EXCPT:
337          ; CHECK FOR OVERFLOW/UNDERFLOW.
338 F2CF ACC4CE        LD X, SP          ; FIRST DO SOME JUGGLING
339 F2D2 02            SET C          ; TO BE COMPATIBLE WITH EXCEPTION
340 F2D3 820ACEEB      SUBC X, OA          ; HANDLING IN OTHER ROUTINES.
341 F2D7 AFCE          PUSH X
342 F2D9 BD7FFF        IFGT A, 07FFF          ; IS BIASED EXPONENT NEGATIVE ?
343 F2DC B4FD3F        JMPL UNDFL
344 F2DF 9C00          IFEQ A, 0          ; IS IT 0 ?
345 F2E1 B4FD3A        JMPL UNDFL          ; YES IT IS STILL UNDERFLOW.
346 F2E4 9DFE          IFGT A, 0FE          ; IS IT GT THAN 254 ?
347 F2E6 B4FD46        JMPL OVRFL
348          ; GET HERE MEANS VALID SP FLP NUMBER.
349 F2E9 3FCE          POP X          ; X POINTS TO MANTISSA SIGN.
350 F2EB E7            SHL A
351 F2EC E7            SHL A
352 F2ED E7            SHL A
353 F2EE E7            SHL A
354 F2EF E7            SHL A
355 F2FD E7            SHL A

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

PAGE: 22

```
356 F2F1 E7          SHL A
357 F2F2 E7          SHL A          ; MOVE EXPONENT TO HIGH BYTE.
358 F2F3 8FFA        OR A, W(X)      ; GET THE MANTISSA SIGN IN.
359 F2F5 AB00        ST A, TMP1     ; SAVE IT IN TMP1.
360 F2F7 3FCA        POP K          ; FI-HI TO K.
361 F2F9 3FC8        POP A          ; FI-LO TO A.
362 F2FB F1          X A, W(X+)     ; SAVE FI-LO.
363 F2FC A8CA        LD A, K
364 F2FE F1          X A, W(X+)     ; SAVE FI-HI.
365 F2FF A800        LD A, TMP1
366 F301 F3          X A, W(X-)     ; SAVE FI-EXP.FI-SIGN, X POINTS TO FI-HI.
367 F302 3664        JSRL SROUND    ; ROUND THE RESULT.
368 F304 F2          LD A, W(X-)     ; X POINTS TO FI-HI.
369 F305 F2          LD A, W(X-)     ; X POINTS TO FI-LO.
370 F306 AFCE        PUSH X
371 F308 368C        JSR FPAK      ; PACK IT INTO IEEE FORMAT.
372 F30A 3FC4        POP SP
373 F30C 3FCC        POP B
374 F30E 3FCE        POP X
375 F310 3C          RET
376
377                      .END
```

```

31          .FORM 'FTOA.MAC'
32          .INCLD FTOA.MAC
1           .TITLE FTOA
2           .LOCAL
3           ;
4           ; THIS SUBROUTINE CONVERTS A SINGLE PRECISION, BINARY FLOATING
5           ; POINT NUMBER IN THE IEEE FORMAT TO A DECIMAL FLOATING POINT
6           ; STRING. THE DECIMAL FLOATING POINT STRING IS OBTAINED TO A
7           ; PRECISION OF 9 DECIMAL DIGITS.
8           ;
9           ; THE ALGORITHM USED IS BASED ON:
10          ; J.T. COONEN, 'AN IMPLEMENTATION GUIDE TO A PROPOSED STANDARD
11          ; FOR FLOATING POINT ARITHMETIC,' IEEE COMPUTER, JAN. 1980, PP 68-79.
12          ;
13          ; ON INPUT, THE BINARY SP FLP NUMBER IS IN REGS. K AND A.
14          ; B CONTAINS THE ADDRESS OF THE LOCATION WHERE THE DECIMAL FLOATING
15          ; POINT STRING IS TO START. NOTE THAT AT LEAST 17 BYTES ARE NEEDED
16          ; FOR THE STORAGE OF THE STRING. THE LAST BYTE IS ALWAYS NULL.
17          ;
18          ; ALL REGISTERS ARE PRESERVED BY THIS SUBROUTINE.
19          ;
20          FTOA:
21 F311 AFCE          PUSH X          ; SAVE X ON THE STACK.
22 F313 AFCC          PUSH B          ; SAVE B ON THE STACK.
23          ; CHECK AND SEE IF F1 IS A NAN.
24 F315 3605          JSR FNACHK
25 F317 07            IF C
26 F318 B401B4        JMPL $NAN          ; YET IT IS, SO GET OUT.
27          ; CHECK AND SEE IF F1 IS ZERO.
28 F31B 36CA          JSR FZCHK
29 F31D 07            IF C
30 F31E B401C8        JMPL $ZERO          ;YES IT IS, SO GET OUT.
31          ; GET HERE MEANS F1 IS A NON-ZERO, NON-NAN FLP NUMBER.
32 F321 ACC4CE        LD X, SP
33 F324 8206C4F8      ADD SP, 06          ; ADJUST SP.
34 F328 36C7          JSR FUNPAK          ; UNPACK THE NUMBER.
35          ; X POINTS ONE WORD PAST F1-EXP.F1-SIGN
36          ; ON RETURN.
37          ;
38          ; COMPUTE THE EXPONENT OF 10 FOR DECIMAL FLP NO.
39          ; THIS IS DONE AS FOLLOWS:
40          ; SUPPOSE F1 = FM * (2^M)
41          ; LET U = M*LOG(2)          NOTE: LOG IS TO BASE 10.
42          ; THEN V = INT(U+1-9)
43          ; IS USED AS THE 10'S EXPONENT.
44          ; NOTE: INT REFERS TO INTEGER PART.
45          ;
46 F32A D2            LD A, M(X-)          ; X POINTS TO F1-EXP.
47 F32B D2            LD A, M(X-)          ; LOAD F1-EXP. X POINTS TO F1-SIGN.
48 F32C B7000000      LD TMP1, 0          ; FIRST GUESS POSITIVE SIGN FOR EXP.
49 F330 B8FF82        ADD A, OFF82          ; REMOVE IEEE BIAS FROM F1-EXP.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FTOA
 FTOA.MAC

PAGE: 24

```

50 F333 AFC8          PUSH A          ; SAVE IT ON THE STACK.
51 F335 AFCE          PUSH X          ; SAVE F1-SIGN ADDRESS ALSO.
52 F337 07            IF C            ; WAS THERE A CARRY ON THE LAST ADD ?
53 F338 46            JP $MLOG2         ; YES, SO 2'S EXP IS POSITIVE.
54 F339 B700FF00      LD TMP1, OFF    ; 2'S EXPONENT IS NEGATIVE.
55 F33D 01            COMP A
56 F33E 04            INC A          ; MAKE IT POSITIVE.
57                    $MLOG2:
58                    ; MULTIPLY M BY LOG(2).
59 F33F BE4D10        MULT A, 04D10      ; LOG(2) IS 0.0100110100010000 TO 16 BITS.
60                    ; X CONTAINS INTEGER PART, AND A FRACT. PART.
61 F342 AECE          X A, X          ; SWAP THE TWO.
62 F344 960010        IF TMP1.0       ; WAS THE 2'S EXPONENT NEGATIVE ?
63 F347 41            JP $CSIGN        ; YES, SO MAKE U NEGATIVE.
64 F348 4B            JP $REMV9       ; NO, SO GO DO V = U + 1 - 9.
65                    $CSIGN:
66 F349 01            COMP A          ; COMP INTEGER PART.
67 F34A AECE          X A, X
68 F34C 01            COMP A          ; FRACTION PART.
69 F34D B80001        ADD A, 01
70 F350 AECE          X A, X
71 F352 07            IF C
72 F353 04            INC A
73                    $REMV9:
74 F354 04            INC A          ; INCREASE FRACTION PART.
75 F355 B8FFF7        ADD A, OFFF7     ; SUBTRACT 9.
76 F358 BD7FFF        IFGT A, 07FFF     ; IS IT NEGATIVE ?
77 F35B 45            JP $CHNGS        ; YES, SO CHANGE ITS SIGN.
78 F35C B7000000      LD TMP1, 0       ; REMEMBER POSITIVE SIGN.
79 F36D 4F            JP $DIV10
80                    $CHNGS:
81 F361 B700FF00      LD TMP1, OFF    ; REMEMBER NEGATIVE SIGN.
82 F365 01            COMP A          ; MAKE V POSITIVE.
83 F366 AECE          X A, X
84 F368 01            COMP A
85 F369 B80001        ADD A, 01
86 F36C AECE          X A, X
87 F36E 07            IF C
88 F36F 04            INC A
89                    $DIV10:
90                    ;
91                    ; V = INT (U+1-9) HAS BEEN COMPUTED AND IS IN A.
92                    ; NOW COMPUTE W = F1/(10^V). W SHOULD BE AN INTEGER, AND IT IS
93                    ; COMPUTED TO A 32 BIT PRECISION.
94                    ; THIS COMPUTATION IS DONE AS FOLLOWS:
95                    ;     IF V > 0, THEN F1/(10^V) = F1*(0.8^V)*(2^(-3V)).
96                    ;     IF V < 0, THEN F1/(10^V) = F1*(0.625^U)*(2^(4V)).
97                    ; SO FIRST MULTIPLY THE MANTISSA OF F1 V TIMES BY 0.8 (OR 0.625)
98                    ; AND THEN ADJUST THE EXPONENT OF F1. NOTE THAT THE PARTIAL PRODUCTS
99                    ; IN MULTIPLYING BY 0.8 (OR 0.625) ARE KEPT NORMALIZED. THIS IS
100                   ; ESSENTIAL TO PRESERVE 32 BIT ACCURACY IN THE FINAL RESULT.

```

```

101          ; SINCE THE MANTISSA OF F1 IS NORMALIZED, AND 0.8 (OR 0.625 IS ALSO
102          ; NORMALIZED, EACH PRODUCT NEEDS AT MOST 1 LEFT SHIFT FOR
103          ; RENORMALIZATION. THE SHIFTS ACCUMULATED DURING RENORMALIZATION ARE
104          ; TRACKED AND ACCOUNTED FOR IN THE CALCULATION.
105 F370 3FCE          POP X          ; X NOW POINTS TO F1-SIGN.
106 F372 AFC8          PUSH A         ; SAVE U ON THE STACK.
107 F374 ACC8CC        LD B, A         ; MOVE V TO B ALSO.
108 F377 F2            LD A, W(X-)     ; X POINTS TO F1-HI.
109 F378 F2            LD A, W(X-)     ; LOAD F1-HI. X POINTS TO F1-L0.
110 F379 ACC8CA        LD K, A
111 F37C F4            LD A, W(X)      ; LOAD F1-L0.
112 F370 960010        IF TEMPL.0     ; IS V NEGATIVE?
113 F380 57            JP $MUL10      ; YES, SO MULTIPLY V TIMES BY .625.
114                                     ; GET HERE MEANS MULTIPLY V TIMES BY .8.
115 F381 A4F390CEAB    LD X, W($DTLO)
116 F386 AFCE          PUSH X         ; LO WORD OF 0.8 TO STACK.
117 F388 A4F392CEAB    LD X, W($DTHI)
118 F38D AFCE          PUSH X         ; HI WORD OF 0.8 TO STACK.
119 F38F 56            JP $JAMIT      ; GO DO MULTIPLICATION.
120          ;
121          .EVEN                     ; FORCE EVEN ADDRESS.
122 F390 CDCC          $DTLO: .WORD OCCCD
123 F392 CCCC          $DTHI: .WORD OCCCD
124 F394 0000          $MILO: .WORD 0
125 F396 00A0          $MTHI: .WORD 0A000
126          ;
127          $MUL10:
128 F398 A4F394CEAB    LD X, W($MTLO)
129 F39D AFCE          PUSH X         ; LO WORD OF 0.625 TO STACK.
130 F39F A4F396CEAB    LD X, W($MTHI)
131 F3A4 AFCE          PUSH X         ; HI WORD OF 0.625 TO STACK.
132          ;
133          $JAMIT:
134 F3A6 9300          LD X, 0         ; INIT X TO TRACK ANY POWERS OF
135                                     ; 2 GENERATED DURING NORMALIZATION
136                                     ; OF PARTIAL PRODUCTS.
137 F3A8 8200CCFC      IFEQ B, 0     ; IS B ALREADY 0 ?
138 F3AC 57            JP $JAMON      ; YES, SO SKIP MULTIPLY LOOP.
139          $JAMLP:
140 F3AD 36E6          JSR BFMUL      ; MULTIPLY.
141 F3AF AECA          X A, K         ; SWAP HI AND LO WORDS OF PART. PROD.
142 F3B1 E7            SHL A
143 F3B2 07            IF C          ; IS THERE A CARRY ?
144 F3B3 4A            JP $ISNED     ; YES, SO SKIP OVER RENORMALIZATION.
145                                     ; GET HERE MEANS NEED TO RENORM.
146 F3B4 A9CE          INC X         ; UPDATE RENORM COUNT.
147 F3B6 AECA          X A, K         ; SWAP HI AND LO PART. PROD.
148 F3B8 E7            SHL A
149 F3B9 07            IF C
150 F3BA 96CA08        SET K.0       ; SET BIT SHIFTED OUT FROM LO WORD.
151 F3BD 43            JP $OVRL

```


NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FTOA
 FTOA.MAC

PAGE: 26

```

152          $ISNED:
153 F3BE D7          RRC A          ; PUT BACK SHIFTED BIT.
154 F3BF AECA          X A, K
155          $OVR1:
156 F3C1 AAC          DECSZ B          ; IS B 0 YET ?
157 F3C3 76          JP $JMLP          ; NO, SO DO IT AGAIN.
158          $JAMON:
159          ; GET HERE MEANS MULTIPLICATION HAS BEEN DONE, SO TAKE CARE
160          ; OF EXPONENT.
161 F3C4 3FCC          POP B
162 F3C6 3FCC          POP B          ; GET RID OF 0.8 (OR 0.625) FROM STACK.
163 F3C8 AFCA          PUSH A          ; SAVE LO WORD OF PROD.
164 F3CA AFCA          PUSH K          ; SAVE HI WORD OF PRODUCT.
165 F3CC A6FFF8C4A8    LD A, W(SP-08) ; GET F1'S BINARY EXPONENT.
166 F3D1 02          SET C
167 F3D2 96CEEB        SUBC A, X          ; SUBTRACT FROM IT ANYTHING COLLECTED
168          ; DURING RENORM.
169 F3D5 ACC8CE          LD X, A          ; AND SAVE IT IN X.
170 F3D8 A6FFFAC4A8    LD A, W(SP-06) ; GET V FROM THE STACK.
171 F3DD 960010          IF Tmpl.0          ; IS V NEGATIVE ?
172 F3E0 49          JP $ML4          ; YES, SO MULTIPLY V BY 4.
173          ; GET HERE MEANS MULTIPLY V BY -3.
174 F3E1 E7          SHL A          ; NOW A CONTAINS 2*V.
175 F3E2 A6FFFAC4F8    ADD A, W(SP-06) ; NOW A CONTAINS 3*V.
176 F3E7 01          COMP A
177 F3E8 04          INC A          ; NOW A CONTAINS -3*V.
178 F3E9 42          JP $ADEM          ; GO FIGURE FINAL EXPONENT.
179          $ML4:
180 F3EA E7          SHL A
181 F3EB E7          SHL A          ; NOW A CONTAINS 4*V.
182          $ADEM:
183 F3EC 96CEF8          ADD A, X          ; A SHOULD NOW BE A POSITIVE INTEGER
184          ; IN THE RANGE 0 TO 32.
185          ; NOW CHECK AND SEE IF A HAS ENOUGH PRECISION.
186 F3EF 9020          IFGT A, 020          ; NEED MORE THAN 32 BITS ?
187 F3F1 5A          JP $INCRV          ; YES, SO GO INCREASE V.
188 F3F2 9D1B          IFGT A, 01B          ; NEED AT LEAST 28 BITS ?
189 F3F4 9435          JMP $GOON          ; YES, SO ALL IS OK. GO ON.
190          ; GET HERE MEANS NEED MORE
191          ; PRECISION, SO DECREASE V.
192 F3F6 3FC8          POP A          ; GET HI-PROD OFF STACK.
193 F3F8 3FC8          POP A          ; GET LO WORD OFF STACK.
194 F3FA 3FC8          POP A          ; GET MAGN. OF V.
195 F3FC 96D010          IF Tmpl.0          ; IS V NEG. ?
196 F3FF 56          JP $VUP          ; YES, SO GO INCR. MAGN. OF V.
197          ; GET HERE MEANS V IS POSITIVE,
198          ; AND NEED TO DECREMENT IT.
199 F400 B8FFFF          ADD A, OFFF          ; SUBTRACT 1 FROM A.
200 F403 07          IF C          ; GOT A CARRY ?
201 F404 5A          JP $GOBAK          ; THEN OK.
202 F405 01          COMP A

```

```

203 F406 04          INC A
204 F407 B700FF00   LD Tmpl, OFF ; U CHANGES SIGN.
205 F40B 53         JP $GOBAK
206                $INCRV:
207 F40C 3FC8       POP A ; GET HI PROD. OFF STACK.
208 F40E 3FC8       POP A ; GET LO PROD. OFF STACK.
209 F410 3FC8       POP A ; GET MAGN. OF V.
210 F412 960010     IF Tmpl.0 ; IS V NEGATIVE ?
211 F415 42         JP $VDOWN ; YES.
212                $VUP:
213 F416 04         INC A
214 F417 47         JP $GOBAK
215                $VDOWN:
216 F418 AAC8       DECSZ A
217 F41A 44         JP $GOBAK
218 F41B B7000000   LD Tmpl, 0 ; V CHANGES SIGN.
219                $GOBAK:
220 F41F ACC4CE     LD X, SP
221 F422 02         SET C
222 F423 8204CEEB   SUBC X, 04
223 F427 AFCE      PUSH X
224 F429 95B9      JMP $DIV10
225                $GOON:
226 F42B 01         COMP A
227 F42C 04         INC A ; NEGATE A.
228 F42D B80020     ADD A, 020 ; SUBTRACT IT FROM 32.
229 F430 ACC8CE     LD X, A ; AND MOVE IT TO X.
230 F433 3FCA      POP X ; GET HI WORD OF PRODUCT.
231 F435 3FC8       POP A ; GET LO WORD OF PROD.
232 F437 8200CEFC   IFEQ X, 0 ; IS X 0 ?
233 F43B 49         JP $INDUN ; YES, SO ALREADY A 32 BIT INTEGER.
234                $INTFY:
235                ; NOW ADJUST THE PRODUCT TO FORM A 32 BIT INTEGER.
236 F43C AECA       X A, K ; SWAP HI AND LO WORDS.
237 F43E C7         SHR A
238 F43F AECA       X A, K
239 F441 D7         RRC A ; SHIFT IT RIGHT ONCE.
240 F442 AAEE       DECSZ X ; X 0 YET ?
241 F444 68         JP $INTFY ; NO SO GO DO SOME MORE.
242                $INDUN:
243                ; GET HERE MEANS K.A CONTAIN THE 32 BIT INTEGER THAT IS THE
244                ; MANTISSA OF THE DECIMAL FLP NUMBER.
245 F445 AFC8       PUSH A ; SAVE LO-INT.
246 F447 AFCA       PUSH K ; SAVE HI INT.
247 F449 A6FFFC4A8 LD A, W(SP-010) ; GET STARTING ADDRESS OF DECIMAL STRING.
248 F44E B80010     ADD A, 010 ; ADD 16 TO IT.
249 F451 ACC8CC     LD B, A ; AND MOVE IT B.
250 F454 00         CLR A
251 F455 C3         XS A, M(B-) ; OUTPUT TERMINATING NULL BYTE.
252 F456 40         NOP
253 F457 A6FFFAC4A8 LD A, W(SP-06) ; GET V.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FTOA
 FTOA.MAC

PAGE: 28

```

254 F45C 9FOA          DIV A, OA          ; DIVIDE IT BY 10. QUOT. IN A,
255                    ; REM. IN X.
256 F45E AECE          X A, X              ; REM TO A.
257 F460 B80030        ADD A, O30         ; MAKE IT INTO ASCII BYTE.
258 F463 C3            XS A, M(B-)       ; OUTPUT IT.
259 F464 40            NOP
260 F465 A8CE          LD A, X
261 F467 B80030        ADD A, O30
262 F46A C3            XS A, M(B-)
263 F46B 40            NOP          ; FINISHED OUTPUTING EXPONENT.
264 F46C 902B          LD A, O28         ; SAY EXP SIGN IS '+'.
265 F46E 960010        IF Tmpl.0
266 F471 902D          LD A, O2D         ; NOPE, IT IS '-'.
267 F473 C3            XS A, M(B-)       ; OUTPUT IT.
268 F474 40            NOP
269 F475 9045          LD A, O45
270 F477 C3            XS A, M(B-)       ; OUTPUT 'E'.
271 F478 40            NOP
272 F479 902E          LD A, O2E
273 F47B C3            XS A, M(B-)       ; OUTPUT '.'.
274 F47C 40            NOP
275                    ; NOW NEED TO OUTPUT 10 DECIMAL DIGITS.
276 F47D B7000A00      LD Tmpl, OA        ; LOAD 10 INTO Tmpl AS LOOP COUNTER.
277                    $DOLUP:
278 F481 3FC8          POP A              ; A CONTAINS HI INT.
279 F483 9FOA          DIV A, OA          ; DIVIDE IT BY 10. QUOT. IN A,
280                    ; REM. IN X.
281 F485 ACC8CA        LD K, A
282 F488 3FC8          POP A              ; A CONTAINS LO INT.
283 F48A AFCC          PUSH B           ; SAVE DEC. STR. ADDR.
284 F48C ACCACC        LD B, K              ; B CONTAINS HI-QUOT.
285 F48F 82            .BYTE 082,0A,0C8,0EF
      F490 OA
      F491 C8
      F492 EF

286                    ; THE ABOVE 4 BYTES REPRESENT THE INSTRUCTION DIVD A, OA.
287                    ; BECAUSE THE ASSEMBLER DOES NOT KNOW ABOUT IT YET, WE HAVE TO
288                    ; KLUDGE IT THIS WAY.
289                    ; AFTER THE DIVD, A CONTAINS THE LO-QUOT. AND X THE REM.
290 F493 ACCCCA        LD K, B              ; MOVE HI-QUOT TO K.
291 F496 3FCC          POP B              ; B CONTAINS DEC. STR. ADDR.
292 F498 AFCC          PUSH A           ; SAVE LO INT.
293 F49A AFCA          PUSH K           ; SAVE HI INT.
294 F49C ABCE          LD A, X              ; MOVE REM TO A.
295 F49E B80030        ADD A, O30         ; ASCII-FY IT.
296 F4A1 C3            XS A, M(B-)       ; AND OUTPUT IT.
297 F4A2 40            NOP
298 F4A3 AA00          DECSZ Tmpl        ; IS Tmpl 0 YET ?
299 F4A5 9524          JMP $DOLUP       ; NO, GO GET SOME MORE.
300                    ; GET HERE MEANS DONE WITH OUTPUTING MANTISSA.
301 F4A7 3FC8          POP A

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FTOA
 FTOA.MAC

PAGE: 29

```

302 F4A9 3FC8          POP A
303 F4AB 3FC8          POP A
304 F4AD 3FC8          POP A          ; GET SOME GARBAGE OFF THE STACK.
305 F4AF 3FCA          POP K          ; GET F1-EXP.F1-SIGN TO K.
306 F4B1 9020          LD A, 02D        ; LOAD SP INTO A.
307 F4B3 96CA10        IF K.0
308 F4B6 902D          LD A, 02D        ; IF MAINT. IS NEG. LOAD '-'.
309 F4B8 06            ST A, M(B)      ; OUTPUT SIGN.
310 F4B9 AFCA          PUSH K          ; F1-EXP.F1-SIGN BACK ON STACK.
311 F4BB ACC4CE        LD X, SP
312 F4BE 02            SET C
313 F4BF 8206CEEB      SUBC X, 06       ; X POINTS TO F1-L0.
314 F4C3 AFCE          PUSH X
315 F4C5 B5FBB4        JSR FPAK        ; PACK IT, SO RESTORING K AND A.
316 F4C8 3FC4          POP SP
317 F4CA 3FCC          POP B          ; RESTORE B.
318 F4CC 3FCE          POP X          ; RESTORE X.
319 F4CE 3C            RET
320                    ;
321                    $NAN:
322                    ; GET HERE MEANS F1 IS A NAN.
323 F4CF AFC8          PUSH A
324 F4D1 ACC0CE        LD X, B
325 F4D4 8210CEF8      ADD X, 010
326 F4D8 00            CLR A
327 F4D9 03            X A, M(X-)
328 F4DA 9210          LD B, 010
329                    $NANLP:
330 F4DC 90FF          LD A, OFF
331 F4DE D3            X A, M(X-)
332 F4DF AAC0          DECSZ B
333 F4E1 65            JP $NANLP
334 F4E2 3FC8          POP A
335 F4E4 3FCC          POP B
336 F4E6 3FCE          POP X
337 F4E8 3C            RET
338                    ;
339                    $ZERO:
340                    ; GET HERE MEANS F1 IS ZERO.
341 F4E9 AFC8          PUSH A
342 F4EB ACC0CE        LD X, B          ; X CONTAINS DECIMAL STRING ADDR.
343 F4EE 8210CEF8      ADD X, 010
344 F4F2 00            CLR A
345 F4F3 D3            X A, M(X-)      ; OUTPUT TERMINATING NULL BYTE.
346 F4F4 9030          LD A, 030        ; LOAD 0 INTO A.
347 F4F6 D3            X A, M(X-)
348 F4F7 9030          LD A, 030
349 F4F9 D3            X A, M(X-)      ; OUTPUT 00 FOR EXPONENT.
350 F4FA 902B          LD A, 02B        ; LOAD '+' SIGN.
351 F4FC D3            X A, M(X-)
352 F4FD 9045          LD A, 045        ; LOAD 'E'

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FTOA
FTOA.MAC

PAGE: 30

```
353 F4FF D3          X A, M(X-)
354 F500 902E        LD A, 02E          ; LOAD '.'.
355 F502 D3          X A, M(X-)
356 F503 920A        LD B, 0A
357                  $ZERLP:
358 F505 9030        LD A, 030
359 F507 D3          X A, M(X-)
360 F508 AAC3        DECSZ B
361 F50A 65          JP $ZERLP
362 F508 9020        LD A, 020          ; LOAD SP.
363 F50D D5          X A, M(X)
364 F50E 3FC8        POP A
365 F510 3FCC        POP B
366 F512 3FCE        POP X
367 F514 3C          RET
368                  ;
369                  .END
```

```

33          .FORM, 'FADD.MAX'
34          .INCLD FADD.MAC
1           .TITLE FADD
2           .LOCAL
3
4           ;
5           ; SUBROUTINE TO ADD/SUBTRACT TWO SP FLOATING POINT NUMBERS.
6           ;           C = F1 + F2 OR C = F1 - F2
7           ;
8           ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
9           ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
10          ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-RO AND IS IN A.
11          ;
12          ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
13          ; STACK POINTER ON ENTRY, THEN
14          ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
15          ; THE LOW WORD OF F2, REFERRED TO AS F2-RO IS AT SP - 6.
16          ;
17          ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
18          ;
19          FSUB:
20          ST A, TMP1
21          LD A, W(SP-06) ; LOAD F2-RO.
22          PUSH A ; AND SAVE ON STACK.
23          LD A, W(SP-06) ; LOAD F2-R1.
24          XOR A, 08000 ; CHANGE THE SIGN.
25          PUSH A ; AND SAVE ON THE STACK.
26          LD A, TMP1 ; RESTORE A.
27          JSR FADD ; CALL THE ADD ROUTINE.
28          ST A, TMP1 ; SAVE A.
29          POP A ; GET RID OF JUNK
30          POP A ; FROM THE STACK.
31          LD A, TMP1 ; RESTORE A.
32          RET
33          ;
34          FADD:
35          ; SAVE ADDRESS OF F2-RO IN TMP1.
36          PUSH X ; SAVE X ON ENTRY.
37          PUSH B ; AND B ON ENTRY.
38          LD X, SP
39          ADD X, OFFF6 ; SUBTRACT 10.
40          LD TMP1, X ; AND SAVE IN TMP1.
41          ; CHECK AND SEE IF F1 IS A NAN.
42          JSR FNACHK
43          IF C
44          JMPL FNAN ; F1 IS A NAN.
45          ; CHECK AND SEE IF F2 IS A NAN.
46          LD B, K
47          LD X, A
48          LD A, W(TMP1+2)
49          LD K, A
50          X A, X

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FADD
 FADD.MAC

PAGE: 32

```

50 F55A B5FAE3 + JSR FNACHK
51 F55D 07 IF C
52 F55E B4FAAE JMPL FNAN ; F2 IS NAN.
53 ; CHECK AND SEE IF F2 IS ZERO.
54 F561 B5FAED + JSR FZCHK
55 F564 06 IFN C
56 F565 48 JP $F1CHK ; F2 IS NOT ZERO. CHECK F1.
57 F566 ACCCCA LD K, B ; F2 IS ZERO, SO ANSWER IS F1.
58 F569 3FCC POP B
59 F56B 3FCE POP X
60 F56D 3C RET
61 ; CHECK AND SEE IF F1 IS ZERO.
62 $F1CHK:
63 F56E ACCCCA LD K, B ; RESTORE F1-R1 FROM B.
64 F571 B5FADD + JSR FZCHK
65 F574 06 IFN C
66 F575 4F JP $NTZERO ; JUMP SINCE F1 IS ALSO NOT ZERO.
67 F567 A20200AB LD A, W(TMP1+2) ; GET HERE MEANS F1 IS ZERO,
68 F57A ACC8CA LD K, A ; SO ANSWER IS F2.
69 F57D ADO0A8 LD A, W(TMP1)
70 F580 3FCC POP B
71 F582 3FCE POP X
72 F584 3C RET
73 ; GET HERE MEANS NORMAL ADDITION.
74 ; UNPACK F1 AND F2.
75 $NTZERO:
76 F585 ACC4CE LD X, SP ; X POINTS TO F1-L0.
77 F588 8210C4F8 ADD SP, 010 ; MOVE SP PAST LOCAL STORAGE.
78 F58C AFCE PUSH X ; SAVE SP ON STACK FOR QUICK RETURN.
79 F58E B5FAD0 + JSR FUNPAK ; UNPACK F1.
80 F591 ACO0CC LD B, TMP1 ; B NOW POINTS TO F2-RO.
81 F594 ACCE00 LD TMP1, X ; TMP1 NOW POINTS TO F2-L0.
82 F597 E0 LDS A, W(B+) ; LOAD F2-RO INTO A.
83 F598 40 NOP
84 F599 AECA X A, K
85 F59B E4 LD A, W(B)
86 F59C AECA X A, K ; LOAD F2-R1 INTO K.
87 F59E B5FAC0 + JSR FUNPAK ; UNPACK F2.
88 ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
89 F5A1 F2 LD A, W(X-)
90 F5A2 ACO0CC LD B, TMP1
91 F5A5 E2 LDS A, W(B-)
92 F5A6 40 NOP
93 ; COMPARE F1-EXP AND F2-EXP.
94 F5A7 F2 LD A, W(X-) ; LOAD F2-EXP.F2-SIGN INTO A.
95 F5A8 B9FF00 AND A, OFF00 ; MASK OUT SIGN.
96 F5AB A6FFFCC4AB ST A, W(SP-4) ; SAVE IN C-SIGN.C-EXP.
97 F5B0 E2 LDS A, W(B-)
98 F5B1 40 NOP ; LOAD F1-EXP.F1-SIGN INTO A.
99 F5B2 B9FF00 AND A, OFF00 ; CHANGE TO F1-EXP.00000000.
100 F5B5 02 SET C

```

```

101 F5B6 A6FFFCC4EB      SUBC A, W(SP-4) ; SUBTRACT F2-EXP.00000000.
102 F5BB 06              IFN C
103 F5BC 942D            JMP $F2GTR       ; F2-EXP IS BIGGER THAN F1-EXP.
104                      ; GET HERE MEANS F1-EXP IS BIGGER THAN F2-EXP.
105 F5BE 80C9CAAB        LD K, H(A)       ; SAVE DIFF. IN K TO BE USED AS LOOP COUNTER.
106 F5C2 A6FFFCC4FB      ADD A, W(SP-4)
107 F5C7 A6FFFCC4AB      ST A, W(SP-4)   ; RESTORE F1-EXP AND STORE IN C-SIGN.
108 F5CC 8217CAFD        IFGT K, 017
109 F5D0 51              JP $ZROF2       ; K GT 23-DEC MEANS F2 GETS ZEROED IN SHIFTS.
110                      ; LOOP TO SHIFT F2 INTO ALIGNMENT.
111 F5D1 8200CAFC        IFEQ K, 0
112 F5D5 943B            JMP $ADDMN      ; K = 0 MEANS DONE SHIFTING.
113                      $LOOP2:
114 F5D7 F4              LD A, W(X)
115 F5D8 C7              SHR A
116 F5D9 F3              X A, W(X-)
117 F5DA F4              LD A, W(X)
118 F5DB D7              RRC A
119 F5DC F1              X A, W(X+)
120 F5DD AACA            DECSZ K
121 F5DF 68              JP $LOOP2
122 F5E0 9430            JMP $ADDMN
123                      $ZROF2:
124                      ; SET F2 MANTISSA TO 0.
125 F5E2 F0              LD A, W(X+)     ; X POINTS TO F2-EXP.F2-SIGN.
126 F5E3 00              CLR A
127 F5E4 F3              X A, W(X-)     ; AND STORE IT BACK.
128 F5E5 00              CLR A
129 F5E6 F3              X A, W(X-)
130 F5E7 00              CLR A
131 F5E8 F1              X A, W(X+)
132 F5E9 9427            JMP $ADDMN
133                      ; F2 EXPONENT IS GREATER THAN F1 EXPONENT.
134                      $F26TR:
135 F5EB 01              COMP A
136 F5EC 04              INC A           ; CHANGE DIFF IN EXP TO POSITIVE.
137 F5ED 80C9CAAB        LD K, H(A)     ; LOAD K WITH LOOP COUNTER.
138 F5F1 8217CAFD        IFGT K, 017
139 F5F5 51              JP $ZROF1     ; F1 MANT. REDUCED TO 0 IN SHIFTS.
140                      ; LOOP TO SHIFT F1 MANT INTO ALIGNMENT.
141 F5F6 8200CAFC        IFEQ K, 0
142 F5FA 57              JP $ADDMN     ; K=0 MEANS DONE SHIFTING.
143                      $LOOP1:
144 F5FB E4              LD A, W(B)
145 F5FC C7              SHR A
146 F5FD E3              XS A, W(B-)
147 F5FE 40              NOP
148 F5FF E4              LD A, W(B)
149 F600 D7              RRC A
150 F601 E1              XS A, W(B+)
151 F602 40              NOP

```


NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 FADD
 FADD.MAC

PAGE: 34

```

152 F603 AACA          DECSZ K          ;
153 F605 6A          JP $LOOP1
154 F606 4B          JP $ADDMN
155          $ZROF1:          ; SET F1 MANT TO 0.
156 F607 E0          LOS A, W(B+)      ; B POINTS TO F1-EXP.F1-SIGN.
157 F608 40          NOP
158 F609 00          CLR A
159 F60A E3          XS A, W(B-)      ; STORE IT BACK.
160 F60B 40          NOP
161 F60C 00          CLR A
162 F60D E3          XS A, W(B-)
163 F60E 40          NOP
164 F60F 00          CLR A
165 F610 E1          XS A, W(B+)
166 F611 40          NOP
167          ; DETERMINE IF MANTISSAS ARE TO BE ADDED OR SUBTRACTED.
168          $ADDMN:          ; B POINTS TO F1-HI, X TO F2-HI.
169 F612 E0          LDS A, W(B+)
170 F613 40          NOP
171 F614 F0          LD A, W(X+)
172 F615 D4          LD A, M(X)      ; LOAD F2-SIGN.
173 F616 D8          XOR A, M(B)      ; XOR WITH F1-SIGN.
174 F617 9C00        IFEQ A, 0
175 F619 9451        JMP $TRADD      ; SAME SIGN SO GO TO ADD MANTISSA.
176          ; GET HERE MEANS TRUE SUBTRACT OF MANTISSA.
177 F61B F2          LD A, W(X-)
178 F61C F2          LD A, W(X-)      ; X POINTS TO F2-LO.
179 F61D E2          LDS A, W(B-)
180 F61E 40          NOP
181 F61F E2          LDS A, W(B-)
182 F620 40          NOP          ; B NOW POINTS TO F1-LO.
183 F621 E0          LDS A, W(B+)
184 F622 40          NOP          ; A NOW CONTAINS F1-LO.
185 F623 02          SET C
186 F624 8FEB        SUBC A, W(X)      ; SUBTRACT F2-LO.
187 F626 F1          X A, W(X+)
188 F627 E0          LDS A, W(B+)      ; A CONTAINS F1-HI.
189 F628 40          NOP
190 F629 8FEB        SUBC A, W(X)      ; SUBTRACT F2-HI.
191 F62B F1          X A, W(X+)
192 F62C 07          IF C
193 F62D 55          JP $F1SIN      ; F1 GE F2, SO SIGN IS F1-SIGN.
194          ; GET HERE MEANS F1 LT F2, SO SIGN IS F2-SIGN.
195 F62E A6FFFC4A8  LD A, W(SP-4)
196 F633 8FDA        OR A, M(X)
197 F635 F3          X A, W(X-)      ; C-EXP.C-SIGN HAS BEEN DETERMINED.
198 F636 F4          LD A, W(X)
199 F637 D1          COMP A
200 F638 F3          X A, W(X-)
201 F639 F4          LD A, W(X)
202 F63A 01          COMP A

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FADD
 FADD.MAC

PAGE: 35

```

203 F63B B80001          ADD A, 01
204 F63E F1             X A, W(X+)
205 F63F 07            IF C
206 F640 8FA9          INC W(X)
207 F642 47            JP $ANORM
208                    ; GET HERE MEANS F1 GE F2.
209                    $FLSIN:
210 F643 A6FFFCC4A8     LD A, W(SP-4)
211 F648 DA             OR A, M(B)
212 F649 F3            X A, W(X-)
213                    ; NORMALIZE THE MANTISSA.
214                    $ANORM:
215 F64A ACCECC         LD B, X          ; B POINTS TO C-HI.
216 F64D E0            LDS A, W(B+)
217 F64E 40            NOP
218 F64F C0            LDS A, M(B+)
219 F650 40            NOP          ; B NOW POINTS TO C-EXP BYTE.
220 F651 9118          LD K, 018       ; SET UP LOOP LIMIT OF 24-DEC IN K.
221                    $NLOOP:
222 F653 F4            LD A, W(X)
223 F654 E7            SHL A
224 F655 07            IF C          ; CARRY MEANS NORMALIZED.
225 F656 9448          JMP $ROUND     ; SO JUMP TO ROUNDING CODE.
226 F658 F3            X A, W(X-)
227 F659 F4            LD A, W(X)
228 F65A E7            SHL A
229 F65B F1            X A, W(X+)
230 F65C 07            IF C
231 F65D 8F08          SET W(X).0
232 F65F ADCC8A        DECSZ M(B)     ; ADJUST EXPONENT.
233 F662 43            JP $OV1
234 F663 B4F9B8        JMPL UNDFL    ; C-EXP ZERO MEANS UNDERFLOW.
235                    $OV1:
236 F666 AACA          DECSZ K       ; DECREMENT LOOP COUNTER.
237 F668 75            JP $NLOOP     ; GO BACK TO LOOP.
238 F669 B4F9B2        JMPL UNDFL    ; UNDERFLOW
239                    ;GET HERE MEANS TRUE ADDITION OF MANTISSA.
240                    $TRADD:
241 F66C E2            LDS A, W(B-)
242 F66D 40            NOP
243 F66E E2            LDS A, W(B-)
244 F66F 40            NOP          ; B NOW POINTS TO F1-HI.
245 F670 F2            LD A, W(X-)
246 F671 F2            LD A, W(X-)
247 F672 F4            LD A, W(X)     ; LOAD F2-LO INTO A.
248 F673 F8            ADD A, W(B)     ; ADD F1-LO.
249 F674 F1            X A, W(X+)     ; STORE IN F2-LO.
250 F675 E0            LDS A, W(B+)
251 F676 40            NOP          ; B NOW POINTS TO F1-HI.
252 F677 F4            LD A, W(X)     ; LOAD F2-HI INTO A.
253 F678 E8            ADC A, W(B)     ; ADD F1-HI WITH CARRY FROM LO ADD.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 FADD
 FADD.MAC

PAGE: 36

```

254 F679 07          IF C
255 F67A 4A          JP $ADJEX          ; IF CARRY, NEED TO INCREASE EXP.
256 F67B F1          X A, W(X+)          ; STORE RESULT IN F2-HI.
257 F67C A6FFFCC4A8 LD A, W(SP-4)        ; GET C-EXP.00000000.
258 F681 8FDA        OR A, M(X)          ; INTRODUCE SIGN.
259 F683 F3          K A, W(X-)          ; STORE IN F2-EXP.F2-SIGN.
260 F684 5B          JP $ROUND
261                  ; GET HERE MEANS NEED TO INCREASE EXP BY 1.
262                  $ADJEX:
263 F685 D7          RRC A
264 F686 F3          X A, W(X-)
265 F687 F4          LD A, W(X)
266 F688 D7          RRC A
267 F689 F1          X A, W(X+)
268 F68A F0          LD A, W(X+)          ; X NOW POINTS TO F2-EXP.F2-SIGN.
269 F68B A6FFFCC4A8 LD A, W(SP-4)        ; GET C-EXP.00000000.
270 F690 B80100      ADD A, 0100          ; INCREASE EXP BY 1.
271 F693 07          IF C
272 F694 B4F998      JMPL OVRF
273 F697 BDFEFF      IFGT A, OFEFF          ; IS BIASED EXPONENT 255-DEC ?
274 F69A B4F992      JMPL OVRF
275 F69D 8FDA        OR A, M(X)
276 F69F F3          X A, W(X-)
277                  ; NEED TO ROUND THE RESULT. X POINTS TO C-HI.
278                  $ROUND:
279 F6A0 B5F9FB      JSRL $ROUND
280                  ; FINAL CHECK OF EXPONENT.
281 F6A3 D0          LD A, M(X+)          ; X NOW POINTS TO C-EXP.
282 F6A4 D2          LD A, M(X-)
283 F6A5 9C00        IFEQ A, 0
284 F6A7 B4F974      JMPL UNDF
285 F6AA 90FE        IFGT A, OFE
286 F6AC B4F980      JMPL OVRF
287 F6AF F2          LD A, W(X-)
288 F6B0 F2          LD A, W(X-)          ; X NOW POINTS TO C-LO.
289 F6B1 B5F9C8      JSR FPAK          ; PACK C.
290 F6B4 3FC4        POP SP          ; SET UP SP FOR RETURN.
291 F6B6 3FCC        POP B
292 F6B8 3FCE        POP X
293 F6BA 3C          RET
294                  ;
295                  .END

```

```

35          .FORM 'FMULT.MAC'
36          .INCLD FMULT.MAC
1           .TITLE FMULT
2           .LOCAL
3           ;
4           ; SUBROUTINE TO MULTIPLY TWO SP FLOATING POINT NUMBERS.
5           ;     C = F1*F2
6           ;
7           ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
8           ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
9           ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-RO AND IS IN A.
10          ;
11          ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
12          ; STACK POINTER ON ENTRY, THEN
13          ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
14          ; THE LOW WORD OF F2, REFERRED TO AS F2-RO IS AT SP - 6.
15          ;
16          ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
17          ; REGS. X AND B ARE PRESERVED.
18          ;
19          FMULT:
20          F6BB AFCE          PUSH X          ; SAVE X ON ENTRY.
21          F6BD AFCC          PUSH B          ; SAVE B ON ENTRY.
22          ; SAVE ADDRESS OF F2-RO IN TMP1.
23          F6BF ACC4CE          LD X, SP
24          F6C2 86FFF6CEF8      ADD X, OFFF6      ; SUBTRACT 10.
25          F6C7 ACCE00          LD TMP1, X      ; SAVE IN TMP1.
26          ; CHECK AND SEE IF F1 IS A NAN.
27          F6CA B5F973          + JSR FNACHK
28          F6CD 07              IF C
29          F6CE B4F93E          JMPL FNAN      ; F1 IS A NAN.
30          ; CHECK AND SEE IF F2 IS A NAN.
31          F6D1 ACCACC          LD B, K
32          F6D4 ACC8CE          LD X, A
33          F6D7 A20200A8        LD A, W(TMP1+2)
34          F6DB ACC8CA          LD K, A
35          F6DE AECE            X A, X
36          F6E0 B5F95D          + JSR FNACHK
37          F6E3 07              IF C
38          F6E4 B4F928          JMPL FNAN      ; F2 IS NAN.
39          ; CHECK AND SEE IF F2 IS ZERO.
40          F6E7 B5F967          + JSR FZCHK
41          F6EA 07              IF C
42          F6EB 94DC            JMP %CZERO      ; F2 IS ZERO.
43          ; CHECK AN SEE IF F1 IS ZERO.
44          F6ED ACCCCA          LD K, B          ; RESTORE F1-R1 FROM B.
45          F6FD B5F95E          + JSR FZCHK
46          F6F3 07              IF C
47          F6F4 94D3            JMP %CZERO      ; F1 IS ZERO.
48          ; GET HERE MEANS NORMAL MULTIPLICATION.
49          ; UNPACK F1 AND F2.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FMULT
 FMULT.MAC

PAGE: 38

```

50 F6F6 ACC4CE          LD X, SP          ; X POINTS TO F1-L0.
51 F6F9 8210C4F8       ADD SF, 010       ; MOVE SF PAST LOCAL STORAGE.
52 F6FD AFCE           PUSH X           ; SAVE SP ON STACK FOR QUICK RETURN.
53 F6FF B5F95F         + JSR FUNPAK       ; UNPACK F1.
54 F702 ACO0CC         LD B, TMP1       ; B NOW POINTS TO F2-R0.
55 F705 ACCE00         LD TMP1, X       ; TMP1 NOW POINTS TO F2-L0.
56 F708 E0            LOS A, W(B+)      ; LOAD F2-R0 INTO A.
57 F709 40            NOP
58 F70A AECA          X A, K
59 F70C E4            LD A, W(B)
60 F70D AECA          X A, K          ; LOAD F2-R1 INTO K.
61 F70F B5F94F         + JSR FUNPAK       ; UNPACK F2.
62                   ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
63 F712 F2            LD A, W(X-)
64 F713 ACO0CC         LD B, TMP1
65 F716 E2            LDS A, W(B-)
66 F717 40            NOP
67                   ; COMPUTE C-EXP AND C-SIGN AND STORE IN F2-EXP AND F2-SIGN.
68 F718 F4            LD A, W(X)       ; A IS (EEEEEEEE-F2).(SSSSSSS-F2)
69 F719 C7            SHR A           ; SHR SINCE SUM OF EXPS CAN BE 9 BITS.
70 F71A ACC8CA        LD K, A          ; K IS (EEEEEEEE-F2).(SSSSSSS-F2)
71 F71D E4            LD A, W(B)       ; A IS (EEEEEEEE-F1).(SSSSSSS-F1)
72 F71E B9FF00        AND A, OFF00    ; MASK OUT SIGN BITS.
73 F721 C7            SHR A           ; A IS (DEEEEEEE-F1).(0000000)
74 F722 96CAF8        ADD A, K         ; A IS (EEEEEEEE-C).(SSSSSSS-F2)
75 F725 F6            ST A, W(X)      ; STORE IN F2-SIGN.
76 F726 E2            LOS A, W(B-)    ; A IS (EEEEEEEE-F1).(SSSSSSS-F1)
77 F727 40            NOP
78 F728 99FF         AND A, OFF      ; MASK OUT EXP BITS.
79 F72A C7            SHR A           ; A IS (00000000SSSSSSS-F1)
80 F72B 8FFB         XOR A, W(X)     ; A IS (EEEEEEEESSSSSSS-C)
81 F72D B8C080        ADD A, 0C080    ; REMOVE EXCESS BIAS OF 127-DEC FROM EXP.
82 F730 07            IF C
83 F731 46            JP $EXCH2      ; IF CARRY, THEN NO UNDERFLOW NOW
84                   ; CHECK TO SEE IF EXP IS ZERO. IF NOT, UNDERFLOW FOR SURE.
85 F732 E7            SHL A
86 F733 07            IF C
87 F734 B4F8E7        JMPL UNDFL     ; UNDERFLOW, SO JUMP.
88 F737 C7            SHR A           ; RESTORE BIT SHIFTED OUT (0).
89                   ; CHECK FOR EXPONENT OVERFLOW.
90 $EXCH2:
91 F738 E7            SHL A
92 F739 07            IF C           ; IF C IS 1,
93 F73A B4F8F2        JMPL OVRFL     ; THEN OVERFLOW FOR SURE.
94 F73D 96C817        IF A.7
95 F740 96C808        SET A.0       ; RESTORE LAST BIT OF SIGN.
96 F743 F3            X A, W(X-)    ; STORE C-EXP. C-SIGN IN F2-EXP.F2-SIGN.
97                   ;
98                   ; MULTIPLY THE MANTISSA.
99                   ; FIRST COMPUTE F1-HI*F2-HI.
100 F744 F2           LD A, W(X-)

```

```

101 F745 ACCE00          LD TMP1, X          ; TMP1 NOW POINTS TO F2-L0.
102 F748 FE             MULT A, W(B)
103 F749 A6FFFAC4AB     ST A, W(SP-6)      ; STORE LOW WORD OF PRODUCT ON STACK.
104 F74E AECE           X A, X
105 F750 A6FFFC4AB     ST A, W(SP-4)      ; STORE HIGH WORD OF PRODUCT ON STACK.
106                     ; NOW COMPUTE F1-HI*F2-L0.
107 F755 AC00CE         LD X, TMP1
108 F758 F0             LD A, W(X+)
109 F759 ACCE00         LD TMP1, X          ; TMP1 NOW POINTS TO F2-HI.
110 F75C FE             MULT A, W(B)
111 F75D AECE           X A, X
112 F75F A6FFFAC4F8     ADD A, W(SP-6)     ; ADD LOW WORD OF LAST PROD. TO HIGH WORD.
113 F764 A6FFFAC4AB     ST A, W(SP-6)
114 F769 07             IF C
115 F76A A6FFFC4A9     INC W(SP-4)        ; IF CARRY, INCREASE HIGH WORD BY 1.
116                     ; FINALLY COMPUTE F1-L0*F2-HI.
117 F76F E2            LDS A, W(B-)      ; ADJUST B TO POINT TO F1-L0.
118 F770 40             NOP
119 F771 AC00CE         LD X, TMP1
120 F774 F4             LD A, W(X)
121 F775 FE             MULT A, W(B)
122 F776 AECE           X A, X
123 F778 A6FFFAC4F8     ADD A, W(SP-6)     ; ADD LOW WORD ACCUMULATED SO FAR.
124 F77D A6FFFAC4AB     ST A, W(SP-6)
125 F782 A6FFFC4AB     LD A, W(SP-4)     ; A CONTAINS HIGH WORD OF PRODUCT.
126 F787 07             IF C
127 F788 04             INC A          ; THEN INCREASE HIGH WORD.
128                     ;
129                     ; MANTISSA MULTIPLICATION DONE. NOW CHECK FOR NORMALIZATION.
130 F789 AC00CE         LD X, TMP1
131 F78C BD7FFF         IFGT A, 07FFF      ; IS MSB OF PRODUCT 1 ?
132 F78F 4D             JP $EXINC      ; YES, INCREASE MANTISSA.
133                     ; NEED TO SHIFT MANTISSA LEFT BY 1 BIT.
134 F790 E7             SHL A
135 F791 F3             X A, W(X-)
136 F792 A6FFFAC4AB     LD A, W(SP-6)
137 F797 E7             SHL A
138 F798 F1             X A, W(X+)
139 F799 07             IF C          ; DID SHIFT OF LOW WORD PUSH OUT A 1 ?
140 F79A 8F08          SET W(X).0      ; YES SO SET LSB OF HIGH WORD.
141 F79C 51             JP $ROUND      ; GO TO ROUNDING CODE.
142                     ;
143                     ; $EXINC:
144                     ; NEED TO INCREASE EXPONENT BY 1. REMEMBER X POINTS TO F2-HI.
144 F79D F3             X A, W(X-)      ; A CONTAINS HIGH WORD, X POINTS TO F2-L0.
145 F79E A6FFFAC4AB     LD A, W(SP-6)     ; GET LOW WORD.
146 F7A3 F1             X A, W(X+)      ; STORE LOW WORD.
147 F7A4 F0             LD A, W(X+)
148 F7A5 F4             LD A, W(X)      ; GET C-EXP.C-SIGN
149 F7A6 B80100         ADD A, 0100      ; INCREASE C-EXP.
150 F7A9 07             IF C
151 F7AA B4F882         JMPL OVRFL      ; EXPONENT OVERFLOW.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER,REV:C,30 JUL 86
 FMULT
 FMULT.MAC

PAGE: 40

```

152 F7AD F3          X A, W(X-)      ; NO OVERFLOW, SO SAVE C-EXP.C-SIGN.
153                ; ROUNDING CODE.
154                $ROUND:
155 F7AE B5F8ED      JSRL SROUND
156                ; FINAL CHECK OF EXPONENT.
157 F7B1 D0          LD A, M(X+)      ; X NOW POINTS TO C-EXP.
158 F7B2 D2          LD A, M(X-)
159 F7B3 9C00        IFEQ A, 0
160 F7B5 B4F866      JMPL UNDFL
161 F7B8 9DFE        IFGT A, OFE
162 F7BA B4F872      JMPL OVRFL
163 F7BD F2          LD A, W(X-)
164 F7BE F2          LD A, W(X-)      ; X NOW POINTS TO C-L0.
165 F7BF B5F8BA      JSR FPAK        ; PACK C.
166 F7C2 3FC4        POP SP          ; SET UP SP FOR RETURN.
167 F7C4 3FCC        POP B
168 F7C6 3FCE        POP X
169 F7C8 3C          RET
170                ; EXCEPTION HANDLING.
171                ; C IS ZERO B'COS ONE OF F1 OR F2 IS ZERO.
172                $CZERO:
173 F7C9 00          CLR A
174 F7CA ACC8CA      LD K, A
175 F7CD 3FCC        POP B
176 F7CF 3FCE        POP X
177 F7D1 3C          RET
178                ;
179                .END

```

```
37          .FORM 'FDIV.MAC'
38          .INCLD FDIV.MAC
1           .TITLE FDIV
2           .LOCAL
3
4           ;
5           ; SUBROUTINE TO DIVIDE TWO SP FLOATING POINT NUMBERS.
6           ; C = F1/F2
7           ;
8           ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
9           ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
10          ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-RO AND IS IN A.
11          ;
12          ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
13          ; STACK POINTER ON ENTRY, THEN
14          ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
15          ; THE LOW WORD OF F2, REFERRED TO AS F2-RO IS AT SP - 6.
16          ;
17          ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
18          ;
19          ; FDIV:
20          ;
21          ;
22          ;
23          ;
24          ;
25          ;
26          ;
27          ;
28          ;
29          ;
30          ;
31          ;
32          ;
33          ;
34          ;
35          ;
36          ;
37          ;
38          ;
39          ;
40          ;
41          ;
42          ;
43          ;
44          ;
45          ;
46          ;
47          ;
48          ;
49          ;
```

19 F7D2 AFCE PUSH X
20 F7D4 AFCC PUSH B
21 ; SAVE ADDRESS OF F2-RO IN TMP1.
22 F7D6 ACC4CE LD X, SP
23 F7D9 86FFF6CEFB ADD X, OFFF6 ; SUBTRACT 10.
24 F7DE ACCB00 LD TMP1, X ; AND SAVE IN TMP1.
25 ; CHECK AND SEE IF F1 IS A NAN.
26 F7E1 85F85C + JSR FNACHK
27 F7E4 07 IF C
28 F7E5 B4F827 JMPL FNAN ; F1 IS A NAN.
29 ; CHECK AND SEE IF F2 IS A NAN.
30 F7E8 ACCACC LD B, K
31 F7EB ACC8CE LD X, A
32 F7EE A20200A8 LD A, W(TMP1+2)
33 F7F2 ACC8CA LD K, A
34 F7F5 AECE X A. X
35 F7F7 B5F846 + JSR FNACHK
36 F7FA 07 IF C
37 F7FB B4F811 JMPL FNAN ; F2 IS NAN.
38 ; CHECK AND SEE IF F2 IS ZERO.
39 F7FE B5F850 + JSR FZCHK
40 F801 07 IF C
41 F802 B4F7FB JMPL DIVBYO ; F2 IS ZERO.
42 ; CHECK AND SEE IF F1 IS ZERO.
43 F805 ACCCCA LD K, B ; RESTORE F1-R1 FROM B.
44 F808 B5F846 + JSR FZCHK
45 F80B 07 IF C
46 F80C 94F1 JMP \$CZERO ; F1 IS ZERO.
47 ; GET HERE MEANS NORMAL DIVISION.
48 ; UNPACK F1 AND F2.
49 F80E ACC4CE LD X, SP ; X POINTS TO F1-L0.

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FDIV
 FDIV.MAC

PAGE: 42

```

50 F811 8210C4F8      ADD SP, 010      ; MOVE SP PAST LOCAL STORAGE.
51 F815 AFCE          PUSH X          ; SAVE SP ON STACK FOR QUICK RETURN.
52 F817 B5F847      + JSR FUNPAK      ; UNPACK F1.
53 F81A AC00CC          LD B, TMP1      ; B NOW POINTS TO F2-RO
54 F81D ACCE00          LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
55 F820 E0           LDS A, W(B+)     ; LOAD F2-RO INTO A.
56 F821 40           NOP
57 F822 AECA          X A, K
58 F824 E4           LD A, W(B)
59 F825 AECA          X A, K      ; LOAD F2-R1 INTO K.
60 F827 B5F837      + JSR FUNPAK      ; UNPAK F2.
61                   ;
62                   ; ENSURE THAT F1-HI IS LESS THAN F2-HI.
63                   ;
64 F82A F2           LD A, W(X-)     ; X POINTS TO F2-EXP.F2-SIGN.
65 F82B F2           LD A, W(X-)     ; X POINTS TO F2-HI.
66 F82C AC00CC          LD B, TMP1      ; B POINTS TO F2-LO.
67 F82F E2           LDS A, W(B-)     ; B POINTS TO F1-EXP.F1-SIGN.
68 F830 40           NOP
69 F831 E2           LDS A, W(B-)     ; LOAD F1-EXP.F1-SIGN.
70 F832 40           NOP      ; B POINTS TO F1-HI.
71 F833 ACC8CA        LD K, A      ; SAVE F1-EXP.F1-SIGN IN K.
72 F836 F4           LD A, W(X)     ; LOAD F2-HI.
73 F837 FD           IFGT A, W(B)    ; IS F2-HI > F1-HI ?
74 F838 51           JP $FEXSN     ; YES, SO ALL IS WELL.
75                   ; GET HERE MEANS NEED TO SHR F1,
76                   ; AND INCREASE ITS EXPONENT.
77 F839 E0           LOS A, W(B+)    ; GET F1-HI.
78 F83A 40           NOP      ; B POINTS TO F1-EXP.F1-SIGN.
79 F83B AECA          X A, K      ; SWAP F1-EXP.F1-SIGN AND F1-HI.
80 F83D B80100        ADD A, 0100     ; INCREASE F1-EXP BY 1.
81 F840 E3           XS A, W(B-)     ; STORE BACK IN F1-EXP.F1-SIGN.
82 F841 40           NOP      ; B POINTS TO F1-HI.
83 F842 E4           LD A, W(B)     ; LOAD F1-HI.
84 F843 C7           SHR A
85 F844 E3           XS A, W(B-)     ; STORE BACK IN F1-HI.
86 F845 40           NOP      ; B POINTS TO F1-LO.
87 F846 E4           LD A, W(B)     ; LOAD F1-LO.
88 F847 D7           RRC A
89 F848 E1           XS A, W(B+)    ; PUT IT BACK IN F1-LO.
90 F849 40           NOP      ; B POINTS TO F1-HI.
91                   ;
92                   ; $FEXSN:
93                   ; DETERMINE C-EXP AND C-SIGN.
94 F84A F0           LD A, W(X+)     ; X POINTS TO F2-EXP.F2-SIGN.
95 F84B E0           LDS A, W(B+)    ; B POINTS TO F1-EXP.F1-SIGN.
96 F84C 40           NOP
97 F84D F4           LD A, W(X)     ; LOAD F2-EXP.F2-SIGN.
98 F84E B9FF00        AND A, OFF00    ; MASK OUT THE SIGN.
99 F851 C7           SHR A      ; ALLOW 9 BITS FOR EXP CALCULATIONS.
100 F852 ACC8CA       LD K, A      ; SAVE IT IN K.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FDIV
 FDIV.MAC

PAGE: 43

```

101 F855 E4          LD A, W(B)          ; LOAD F1-EXP.F1-SIGN.
102 F856 B9FF00     AND A, OFF00        ; MASK OUT SIGN.
103 F859 C7         SHR A
104 F85A 02         SET C
105 F85B 96CAEB     SUBC A, K          ; SUBTRACT THE EXPONENTS.
106                ; NOTE THAT NOW THE MS 9 BITS
107                ; OF A CONTAIN A 2'S COMP. INTEGER.
108 F85E B7000000   LD TMP1, 0
109 F862 E7         SHL A
110 F863 07         IF C
111 F864 B700FF00   LD TMP1, OFF
112 F868 D7         RRC A          ; SAVE SIGN OF NUMBER IN TMP1.
113 F869 B83F00     ADD A, 03F00        ; RESTORE IEEE BIAS.
114 F86C E7         SHL A          ; MAKE EXPONENT 8 BITS.
115 F86D 06         IFN C          ; NO CARRY ?
116 F86E 49         JP $FSIGN      ; THEN ALL IS WELL.
117 F86F 960010     IF TMP1.0      ; WAS EXP NEGATIVE BEFORE ?
118 F872 B4F7A9     JMPL UNDFL     ; YES, SO UNDERFLOW.
119 F875 B4F7B7     JMPL OVRFL     ; OTHERWISE OVERFLOW.
120                ;
121                ; $FSIGN:
122                ; C-EXP HAS BEEN COMPUTED. NOW FIND C-SIGN.
123                ; BUT FIRST TAKE CARE OF SPECIAL OVER/UNDERFLOW CASES.
124 F878 BCFF00     IFEQ A, OFF00
125 F87B B4F7B1     JMPL OVRFL
126 F87E 9C00       IFEQ A, 0
127 F880 B4F79B     JMPL UNDFL
128 F883 AB00       ST A, TMP1      ; SAVE C-EXP.00000000 IN TMP1.
129 F885 F4         LD A, W(X)      ; LOAD F2-EXP.F2-SIGN.
130 F886 99FF       AND A, OFF      ; MASK OUT F2-EXP.
131 F888 FB         XOR A, W(B)     ; A NOW HAS F1-EXP.C-SIGN.
132 F889 99FF       AND A, OFF      ; MASK OUT F1-EXP.
133 F88B 9600FA     OR A, TMP1      ; BRING IN C-EXP.
134 F88E F3         X A, W(X-)     ; STORE IN F2-EXP.F2-SIGN.
135                ; X POINTS TO F2-HI.
136 F88F F2         LD A, W(X-)     ; X POINTS TO F2-LO.
137 F890 E2         LDS A, W(B-)    ; B POINTS TO F1-HI.
138 F891 40         NOP
139                ;
140                ; NOW DO THE MANTISSA DIVISION.
141                ;
142 F892 FO         LD A, W(X+)     ; LOAD F2-LO. X POINTS TO F2-HI.
143 F893 ACCE00     LD TMP1, X      ; SAVE ADDRESS OF F2-HI IN TMP1.
144 F896 FE         MULT A, W(B)    ; COMPUTE F2-LO*F1-HI.
145                ; X CONTAINS MS WORD AND A IS LS WORD.
146                ;
147 F897 AECC       X A, B          ; A POINTS TO F1-HI, B CONTAINS LS WORD.
148 F899 AE00       X A, TMP1      ; A POINTS TO F2-HI, TMP1 POINTS TO F1-HI.
149 F89B AECC       X A, B          ; A CONTAINS LS WORD, B POINTS TO F2-HI.
150                ;
151 F890 EF         .BYTE OEF       ; DIVD A, W(B) - KLUDGED !!

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86
 FDIV
 FDIV.MAC

PAGE: 44

```

152
153 F89E ACC8CA      ; LD K, A      ; SAVE QUOTIENT IN K.
154 F8A1 A8CC        LD A, B      ; A POINTS TO F2-HI.
155 F8A3 AE00        X A, TMP1   ; A POINTS TO F1-HI, TMP1 POINTS TO F2-HI.
156 F8A5 AECC        X A, B      ; B POINTS TO F1-HI.
157 F8A7 E2         LDS A, W(B-) ; B POINTS TO F1-LO.
158 F8A8 40         NOP
159 F8A9 E0         LOS A, W(B+) ; LOAD F1-LO.
160 F8AA 40         NOP        ; B POINTS TO F1-HI.
161 F8AB 02         SET C
162 F8AC 96CAEB      SUBC A, K    ; SUBTRACT QUOTIENT SAVED IN K.
163 F8AF ACC8CE      LD X, A     ; AND SAVE IN X.
164 F8B2 E4         LD A, W(B)  ; LOAD F1-HI.
165 F8B3 06         IFN C      ; IF C WAS NOT SET IN THE LAST SUBTRACT,
166 F8B4 05         DEC A      ; ADJUST THE BORROW.
167 F8B5 AECE       X A, X
168 F8B7 ACOOCC     LD B, TMP1  ; B POINTS TO F2-HI.
169
170 F8BA EF        ; .BYTE OEF  ; DIVD A, W(B) - KLUDGED AGAIN !
171                ; QUOTIENT IN A, REM IN X.
172
173 F8BB AB00      ; ST A, TMP1  ; SAVE QUOTIENT IN TMP1.
174 F8BD 00        CLR A      ; ZERO A.
175
176 F8BE EF        ; .BYTE OEF  ; DIVD A, W(B) - KLUDGED YET AGAIN !
177
178 F8BF AE00      ; X A, TMP1  ; SWAP OLD AND NEW QUOTIENTS.
179
180                ; CHECK FOR NORMALIZATION. CAN BE OFF BY AT MOST 1 BIT.
181 F8C1 E7        SHL A
182 F8C2 07        IF C
183 F8C3 56        JP $NMED  ; IT IS NORMALIZED.
184                ; GET HERE MEANS NEED TO SHIFT LEFT ONCE.
185 F8C4 AE00      X A, TMP1  ; SWAP HI AND LO WORDS.
186 F8C6 E7        SHL A
187 F8C7 AE00      X A, TMP1  ; HI WORD IS IN A, LO WORD IN TMP1.
188 F8C9 07        IF C      ; WAS 1 SHIFTED OUT OF LO WORD?
189 F8CA 96C808    SET A.0    ; YES, THEN SET LSB OF HI WORD.
190 F8CD ABCA      ST A, K    ; SAVE HI WORD IN K.
191 F8CF E1        XS A, W(B+)
192 F8D0 40        NOP        ; B POINTS TO F2-EXP.F2-SIGN.
193 F8D1 E4        LD A, W(B)  ; LOAD F2-EXP.F2-SIGN.
194 F8D2 B8FF00    ADD A, OFF00 ; SUBTRACT 1 FROM EXPONENT.
195 F8D5 E3        XS A, W(B-) ; STORE BACK IN F2-EXP.F2-SIGN.
196 F8D6 40        NOP        ; B POINTS TO F2-HI.
197 F8D7 A8CA     LD A, K    ; HI WORD TO A.
198 F8D9 E7        SHL A
199
200 F8DA D7        $NMED: RRC A      ; RESTORE BIT OUT.
201 F8DB E3        XS A, W(B-) ; SAVE HI-WORD IN F2-HI.
202 F8DC 40        NOP        ; B POINTS TO F2-LO.

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FDIV
 FDIV.MAC

PAGE: 45

```

203 F8DD A800          LD A, TMP1
204 F8DF E1           XS A, W(B+)      ; SAVE C-LO.
205 F8E0 40           NOP                ; B POINTS TO F2-HI.
206 F8E1 ACCCCE       LD X, B            ; MOVE ADDRESS OF F2-HI TO X.
207                   ;
208                   ; ROUNDING CODE.
209 F8E4 B5F7B7       JSRL SROUND
210                   ; FINAL CHECK OF EXPONENT.
211 F8E7 D0           LD A, M(X+)      ; X NOW POINTS TO C-EXP.
212 F8E8 D2           LD A, M(X-)
213 F8E9 9C00         IFEQ A, 0
214 F8EB B4F730       JMPL UNDFL
215 F8EE 9DFE         IFGT A, OFE
216 F8F0 B4F73C       JMPL OVRFL
217 F8F3 F2           LD A, W(X-)
218 F8F4 F2           LD A, W(X-)      ; X NOW POINTS TO C-LO.
219 F8F5 B5F784       JSR FPAK          ; PACK C.
220 F8F8 3FC4         POP SP           ; SET UP SP FOR RETURN.
221 F8FA 3FCC         POP B
222 F8FC 3FCE         POP X
223 F8FE 3C           RET
224                   ; C IS ZERO B'COS F1 IS ZERO.
225 $CZERO:
226 F8FF 00           CLR A
227 F900 ACC8CA       LD K, A
228 F903 3FCC         POP B
229 F905 3FCE         POP X
230 F907 3C           RET
231                   ;
232                   .END

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 FDIW
 FSINX.MAC

PAGE: 46

```

39                                     .FORM 'FSINX.MAC'
40                                     .INCLD FSINX.MAC
1   ;
2   .TITLE SINX
3   .LOCAL
4   ; A VERY DIRTY APPROXIMATION TO SIN(X).
5   ; X SHOULD BE IN RADIANS.
6   ;
7   ; ON INPUT X SHOULD BE IN IEEE FLP FORMAT IN REGS. K AND A.
8   ; ON RETURN SIN(X) IS IN IEEE FLP FORMAT IN REGS. K AND A.
9   ;
10  SINX:
11  F908 AFCE          PUSH X          ; SAVE X.
12  F90A AFC8          PUSH A
13  F90C AFCA          PUSH K          ; X TO THE STACK.
14  F90E 3653          - JSRL FMULT    ; COMPUTE X^2.
15  F910 AFC8          PUSH A
16  F912 AFCA          PUSH K          ; X^2 TO THE STACK.
17  F914 B6F994A8      LD A, W($A5L0)
18  F918 A4F996CAAB    LD K, W($A5HI) ; LOAD A5.
19  F91D 3662          - JSRL FMULT    ; COMPUTE A5*X^2.
20  F91F AFC8          PUSH A
21  F921 AFCA          PUSH K
22  F923 B6F998A8      LD A, W($A4L0)
23  F927 A4F99ACAAB    LD K, W($A4HI) ; LOAD A4.
24  F92C B5FBE6        JSRL FSUB     ; COMPUTE A4-A5*X^2.
25  F92F 3FCE          POP X
26  F931 3FCE          POP X
27  F933 3678          - JSRL FMULT    ; COMPUTE
28                                     ; X^2(A4 - A5*X^2).
29  F935 AFC8          PUSH A
30  F937 AFCA          PUSH K
31  F939 B6F99CAB      LD A, W($A3L0)
32  F93D A4F99ECAAB    LD K, W($A3HI) ; LOAD A3.
33  F942 B5FBDO        JSRL FSUB     ; COMPUTE
34                                     ; A3 - X^2(A4 - A5*X^2).
35  F945 3FCE          POP X
36  F947 3FCE          POP X
37  F949 368E          - JSRL FMULT    ; COMPUTE
38                                     ; X^2(A3 - X^2(A4 - A5*X^2)).
39  F94B AFC8          PUSH A
40  F94D AFCA          PUSH K
41  F94F B6F9A0A8      LD A, W($A2L0)
42  F953 A4F9A2CAAB    LD K, W($A2HI) ; LOAD A2.
43  F958 B5FBBA        JSRL FSUB     ; COMPUTE
44                                     ; A2 - X^2(A3 - X^2(A4 - A5*X^2)).
45  F95B 3FCE          POP X
46  F95D 3FCE          POP X
47  F95F 36A4          - JSRL FMULT    ; COMPUTE
48                                     ; X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
49  F961 AFC8          PUSH A

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
FSINX.MAC

PAGE: 47

```

50 F963 AFCA          PUSH K
51 F965 B6F9A4AB     LD A, W($ALLO)
52 F969 A4F9A6CAAB   LD K, W($ALHI) ; LOAD A1.
53 F96E B5FBA4       JSRL FSUB        ; COMPUTE
54                   ; A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
55 F971 3FCE         POP X
56 F973 3FCE         POP X
57 F975 36BA         - JSRL FMULT        ; COMPUTE
58                   ; X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
59 F977 AFC8         PUSH A
60 F979 AFCA         PUSH K
61 F97B B13F80       LD K, 03F80
62 F97E 00           CLR A           ; LOAD 1.0 INTO K-A.
63 F97F B5FB93       JSRL FSUB        ; COMPUTE
64                   ; 1 - ALL THE JUNK ABOVE.
65 F982 3FCE         POP X
66 F984 3FCE         POP X
67 F986 3FCE         POP X
68 F988 3FCE         POP X           ; NOW X IS AT THE TOP OF STACK.
69 F98A 36CF         - JSRL FMULT        ; COMPUTE
70                   ; X(1 - X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
71 F98C 3FCE         POP X
72 F98E 3FCE         POP X
73 F990 3FCE         POP X
74 F992 3C           RET
75                   ;
76 F993 40           .EVEN
77                   ;
78 F994 2B32         $A5LO: .WORD 0322B
79 F996 D732         $A5HI: .WORD 032D7
80 F998 1DEF         $A4LO: .WORD 0EF1D
81 F99A 3836         $A4HI: .WORD 03638
82 F99C 010D         $A3LO: .WORD 00D01
83 F99E 5039         $A3HI: .WORD 03950
84 F9A0 8988         $A2LO: .WORD 08889
85 F9A2 083C         $A2HI: .WORD 03C08
86 F9A4 ADA4         $A1LO: .WORD 0AAAD
87 F9A6 2A3E         $A1HI: .WORD 03E2A
88                   ;
89                   ; A DIRTY APPROXIMATION TO COS(X) USING SIN(X).
90                   ;
91 COSX:
92 F9A8 AFCE         PUSH X
93 F9AA ACC8CE       LD X, A
94 F9AD B6F9C8A8     LD A, W($PI2LO)
95 F9B1 AFC8         PUSH A
96 F9B3 B6F9CAA8     LD A, W($PI2HI)
97 F9B7 AFC8         PUSH A
98 F9B9 ABCE         LD A, X
99 F9BB B5FB77       JSRL FADD        ; COMPUTE X + PI/2.
100 F9BE 3FCE        POP X

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 SINX
 FSINX.MAC

PAGE: 48

```

101 F9C0 3FCE          POP X
102 F9C2 34BA          - JSRL SINX          ; COMPUTE SIN(X+PI/2).
103 F9C4 3FCE          POP X
104 F9C6 3C            RET
105                    ;
106 F9C7 40            .EVEN
107 F9C8 DBOF          $PI2LO: .WORD 00FDB
108 F9CA C93F          $PI2HI: .WORD 03FC9
109                    ;
110                    ; A DIRTY APPROXIMATION TO TAN(X) USING SINX AND COSX.
111                    ;
112 TANX:
113 F9CC AFCE          PUSH X
114 F9CE AFCC          PUSH B
115 F9D0 AFC8          PUSH A
116 F9D2 AFCA          PUSH K
117 F9D4 342C          JSR COSX          ; COMPUTE COS(X)
118 F9D6 ACC8CE        LD X, A
119 F9D9 ACCACC        LD B, K
120 F9DC 3FCA          POP K
121 F9DE 3FC8          POP A
122 F9E0 AFCE          PUSH X
123 F9E2 AFCC          PUSH B
124 F9E4 34DC          JSR SINX          ; COMPUTE SIN(X).
125 F9E6 3614          JSR FDIV          ; COMPUTE TAN(X) = SIN(X)/COS(X).
126 F9E8 3FCC          POP B
127 F9EA 3FCC          POP B
128 F9EC 3FCC          POP B
129 F9EE 3FCE          POP X
130 F9F0 3C            RET
131                    ;
132                    .END
41                    ;
42                    ;
43 FFFE 00F0          .END LISTER

```

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
 SINX
 SYMBOL TABLE

PAGE: 49

A	00C8 W	ATOF	F13A *	B	00CC W	BFMUL	FOC7
COSX	F9A8	DIVBYO	F000	FADD	F535	FDIV	F7D2
FMULT	F6BB	FNACHK	F040	FNAN	F00F	FPAK	F07C
FFERWD	0002 W	FPTRAP	F09D	FSUB	F515	FTOA	F311 *
FUNPAK	F061	FZCHK	F051	ISIOK	F105	K	00CA W
LISTER	F000	MULLO	F118	OVRF1	F02F	PC	00C6 W
SINX	F908	SP	00C4 W	SROUND	F09E	TANX	F9CC *
TMP1	0000 W	UNDFL	F01E	X	00CE W	\$A10EX	FLF4
\$A1HI	F9A6	\$A1LO	F9A4	\$A2HI	F9A2	\$A2LO	F9A0
\$A3HI	F99E	\$A3LO	F99C	\$A4HI	F99A	\$A4LO	F998
\$A5HI	F996	\$A5LO	F994	\$ACCF	F1A9	\$ACCM	F177
\$ADDEX	F1FF	\$ADDMN	F612	\$ADEM	F3EC	\$ADJEX	F685
\$ANORM	F64A	\$ANOTO	F05C	\$CHKOT	F113	\$CHNGS	F361
\$CSIGN	F349	\$CZERO	F7C9	\$CZERO	F8FF	\$DIV10	F272
\$DIV10	F370	\$DOLUP	F481	\$DTHI	F270	\$DTHI	F392
\$DTLO	F26E	\$DTLO	F390	\$ESAVE	F20F	\$ESIGN	F1C7
\$EXACC	F1C9	\$EXCH2	F738	\$EXCHR	F1BB	\$EXCLP	FLD4
\$EXCOL	F1BA	\$EXCPT	F2CF	\$EXIN2	F0B7	\$EXINC	F79D
\$EXIT	FOC6	\$FLCHK	F56E	\$F1SIN	F643	\$F2GTR	F5EB
\$TEXSN	F84A	\$FRCOL	F18B	\$FSIGN	F878	\$GOBAK	F41F
\$GOON	F42B	\$HIUP	FOAF	\$INCOL	F15C	\$INCRV	F40C
\$INDUN	F445	\$INTFY	F43C	\$ISNAN	F04C	\$ISNED	F298
\$ISNED	F3BE	\$ISNXT	F17D	\$JAMDN	F29E	\$JAMDN	F3C4
\$JAMIT	F280	\$JAMIT	F3A6	\$JAMLP	F287	\$JAMLP	F3AD
\$LOOP1	F147	\$LOOP1	F5FB	\$LOOP2	F5D7	\$ML4	F3EA
\$MLOG2	F33F	\$MSIGN	F154	\$MTHI	F26C	\$MTHI	F396
\$MTLO	F26A	\$MTLO	F394	\$MULLO	F398	\$NAGAS	F2C3
\$NAN	F4CF	\$NANLP	F4DC	\$NEG10	F20B	\$NLOOP	F653
\$NMED	F8DA	\$NORM1	F230	\$NORM2	F235	\$NRDUN	F247
\$NRLUP	F237	\$NTZER	F585	\$OV1	F666	\$OVR1	F29B
\$OVR1	F3C1	\$PI2HI	F9CA	\$PI2LO	F9C8	\$REMV9	F354
\$RNDUP	FOA7	\$ROUND	F6A0	\$ROUND	F7AE	\$TRADD	F66C
\$VDOWN	F418	\$VUP	F416	\$ZERLP	F505	\$ZERO	F4E9
\$ZROF1	F607	\$ZROF2	F5E2				

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
MACRO TABLE

PAGE: 50

NO WARNING LINES

NO ERROR LINES

2547 ROM BYTES USED

SOURCE CHECKSUM = A31F
OBJECT CHECKSUM = 2AC3

INPUT FILE C:LISTER.MAC
LISTING FILE C:LISTER.PRN
OBJECT FILE C:LISTER.LM

A Radix 2 FFT Program for the HPC

National Semiconductor
Application Note 487
Ashok Krishnamurthy



INTRODUCTION

This report describes the implementation of a radix-2, Decimation-in-time FFT algorithm on the HPC. The program, as presently set up can do FFTs of length 2, 4, 8, 16, 32, 64, 128 and 256. The program can be easily modified to work with higher FFT lengths by increasing the Twiddle Factor table.

FFT FUNDAMENTALS

If $x(n)$, $n = 0, 1, \dots, N-1$ are N samples of a time domain signal, its Discrete Fourier Transform (DFT) is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk}, \quad k = 0, 1, \dots, N-1$$

where $W = e^{-j2\pi/N}$

The straight evaluation of the above equation requires on the order of N^2 complex multiplies. The FFT is nothing but a fast algorithm to compute the DFT that uses only on the order of $N \log(N)$ complex multiplies. Many different FFT algorithms exist (please see references 1, 2 and 3). The algorithm implemented for the HPC is the most common type of FFT — a radix-2, Decimation-in-time algorithm. This class of algorithms requires that the number of input samples, N , be a power of 2. This is usually not a problem, since the input data can be zero padded to achieve this. The development of this algorithm is described in references 1 and 2; the discussion here is brief and based on reference 1.

Separating the DFT summation above into the even-numbered points and odd-numbered points of $x(n)$, we can rewrite the above sum as:

$$X(k) = \sum_{n \text{ even}} x(n) W^{nk} + \sum_{n \text{ odd}} x(n) W^{nk}$$

Using $n = 2r$ for n even and $n = 2r + 1$ for n odd, we can further rewrite the above as:

$$X(k) = \sum_{r=0}^{N/2-1} x(2r) W^{2rk} + W^k \sum_{r=0}^{N/2-1} x(2r+1) W^{2rk}$$

If $G(k)$ is the $N/2$ point DFT of $x(2r)$ and $H(k)$ is the $N/2$ point DFT of $x(2r+1)$, the above equation can be written as:

$$X(k) = G(k) + W^k H(k)$$

This equation shows that a N point DFT can be written as the sum of two $N/2$ point DFTs. The $N/2$ point DFTs can be computed as the sum two $N/4$ point DFTs and so on until we are left with two point DFTs. The two point DFTs can be trivially evaluated by direct computation.

Figure 1, taken from reference 1, shows the decomposition for the case $N = 8$. With reference to this figure, we can note the following points.

1. If N is the number of points in the original sequence, where $N = 2^L$, then there are L stages in the DFT decomposition.

2. The basic computation unit is the so-called Butterfly, shown in Figure 2. Each stage involves the computation of $N/2$ butterflies.
3. The results from the computation in one stage are fed to the next stage after multiplication by some power of W . These powers of W are the so-called Twiddle Factors. Note that each power of W is really a complex number that can be represented by its real and imaginary parts. The real part of W^k is $\cos(2\pi k/N)$ and the imaginary part is $-\sin(2\pi k/N)$.
4. The number of distinct Twiddle Factors used in the first stage is 1, in the second stage is 2 etc., until the L^{th} stage that involves $2^{L-1} = N/2$ twiddle factors. Each twiddle factor in the first stage is involved in $N/2$ Butterflies, in the second stage with $N/4$ butterflies etc., until in the L^{th} stage each twiddle factor is involved with $N/(2^L) = 1$ butterfly.
5. The input data sequence needs to be suitably scrambled if the output sequence is to be in the proper order. This scrambling is easily accomplished by using the so-called Bit-Reverse counter as outlined in reference 2.
6. The outputs from each stage can be stored back again in the same storage area as the input sequence. This gives the algorithm the in-place property. Thus the final DFT results overwrite the initial data.

THE INVERSE FFT

If $X(k)$ $k = 0, 1, \dots, N-1$ is the DFT of a sequence, then its inverse DFT, $x(n)$, is defined as follows:

$$x(n) = \left(\frac{1}{N}\right) \sum_{k=0}^{N-1} X(k) W^{-nk} \quad n = 0, 1, \dots, N-1.$$

Thus the Inverse FFT is the same as the forward FFT except for the following: 1. Negative powers of W are used instead of positive powers; and 2. The final sequence is scaled by $1/N$. The basic FFT program can therefore be used to compute the inverse FFT with these two changes. This is the approach used in the HPC implementation.

TWIDDLE FACTOR TABLE

The brief description of the FFT in the previous section shows that the algorithm needs to use the Twiddle Factors W^k in the computation. The twiddle factors can either be computed as required, they can be computed using a recursive relation, or they can be obtained by looking up in a table (Ref. 2). The approach used in the HPC implementation is to construct a table containing the needed twiddle factors. This table is stored in ROM and values needed are looked up from this table. The length of the table needed is determined by the maximum FFT length that you want to use. The HPC FFT implementation is presently limited to a maximum length of 256. This requires that the twiddle factors W^0, W^1, \dots, W^{255} be available, where

$W = e^{-j2\pi/256}$. Since $e^{ix} = \cos(x) + j\sin(x)$, the values stored in this table are $\cos(0)$, $\sin(0)$, $\cos(2\pi/256)$, $\sin(2\pi/256)$ etc., up to $\cos(2\pi \times 255/256)$, $\sin(2\pi \times 255/256)$. The table used in the implementation is organized as follows:

```
.WORD cos(0) × 214
.WORD sin(0) × 214
.WORD cos(2π/256) × 214
.WORD sin(2π/256) × 214
.
.
.
.WORD cos(2π255/256) × 214
.WORD sin(2π255/256) × 214
```

This table is available in the file TWDTBL.MAC and occupies 1024 bytes of storage.

DATA STORAGE

The data to be transformed, $x(0), \dots, x(N-1)$ are also regarded as complex numbers with a real and an imaginary part. Let $x_r(i)$ be the real part of $x(i)$ and $x_i(i)$ the imaginary part of $x(i)$. Then the data needs to be stored as follows:

```
.WORD xr(0)
.WORD xi(0)
.WORD xr(1)
.WORD xi(1)
.
.
.
.WORD xr(N-1)
.WORD xi(N-1)
```

The length of this storage area obviously depends on the number of data points to be transformed. Note that the FFT program itself does not use any base page user RAM. Also, only 8 words of stack are needed. Thus the base page user RAM can be used to store the data to be transformed. Since 192 bytes are available in this area, transforms of up to 32 point in length can be in the single chip mode with no external RAM.

USING THE FFT PROGRAM

The FFT program along with test data to test the program is provided in the files FFT.MAC, TSTDAT.MAC and TWDTBL.MAC. TSTDAT.MAC contains the test data, and the output from the FFT routines. TWDTBL.MAC contains the Twiddle Factors. The FFT computation involves the use

of 4 different subroutines: FFT, IFFT, BRNCNTR and SMULT. FFT does the forward FFT calculation, IFFT the Inverse FFT calculation, BRNCNTR implements the bit reversed counter, and SMULT does signed multiplication.

Two global symbols need to be defined by the user to use the FFT routines. The first, called TWSTAD should be set to the address of the start of the twiddle factor table. The second, called DTSTAD, should be set to the address of the start of the data area to be transformed. For details on the organization of these storage areas, see the preceding sections.

The actual number of data points to be transformed needs to be passed to the FFT routines. This is done as follows.

Two symbols that refer to words of on-chip RAM have been defined. The first is NUMB = W(01C0) and the second is L1 = W(01C2). Before calling the FFT routine, the user should load NUMB with N, the number of data points to be transformed, and L1 with L, $N = 2L$.

To do a forward FFT, call FFT; to do an inverse FFT, call IFFT. In both cases, the output of the transform overwrites the input data.

INCREASING THE MAXIMUM TRANSFORM LENGTH

The maximum transform length for the FFT program is primarily limited by the size of the Twiddle Factor table. To increase the transform length, the following needs to be done.

1. Increase the Twiddle Factor table. Thus, if the maximum transform length required is 1024, the table needs to store the cosine and sine of the angles

$$0, 2\pi/1024, 2\pi \times 2/1024, \dots, 2\pi \times 1023/1024$$

2. Change the global symbol LMAX such that the maximum transform length is 2^{LMAX} .

FFT/IFFT TEST PROGRAM

The data in the file TSTDAT.MAC can be used to test the FFT program. The data and its transform value is from reference 3. The program in reference 3 is for a Floating point FORTRAN FFT program. Since the HPC FFT program is a fixed point one, the input data needs to be suitably scaled. The scale factor chosen is 2^{10} . The file TSTDAT.MAC contains the scaled input data, and the expected transform. The input data is stored in memory words 200/27E and the expected transform is stored in memory words 280/2FE. To run the test program, do the following.

Set up the MOLE Development System with Blocks 0, 13, 14 and 15 mapped ON. Download the program to the MOLE. Set up a Breakpoint at F410. Run the program starting at F400. When the program is breakpointed, list memory words 200/27F and compare them with memory words 280/2FE.

Note that any difference between the expected DFT values and the DFT values actually computed is due to the fixed point computations in the FFT program.

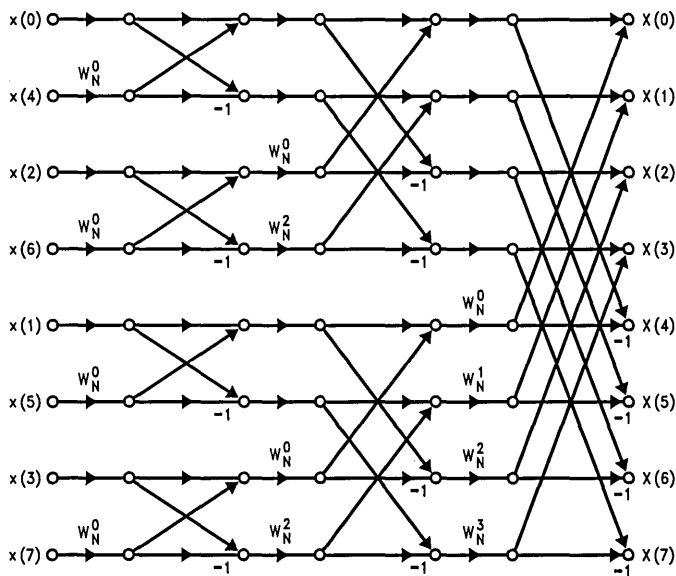
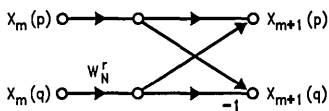


FIGURE 1. FFT Flow Graph for N = 8 Points

TL/DD/9259-1



TL/DD/9259-2

FIGURE 2. The Butterfly—The Basic Computation Unit in the FFT

REFERENCES

1. A.V. Oppenheim and R.W. Schaffer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
2. L.R. Rabiner and B. Gold, *Theory and Applications of Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
3. IEEE ASSP Society Digital Signal Processing Committee, *Programs for Digital Signal Processing*, IEEE Press, New York, 1979.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162

Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

Listing of FFT Program Code

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER, REV: C, 30 JUL 86

PAGE: 1

```

1          ;
2          ; THIS PROGRAM IMPLEMENTS A RADIX-2, DECIMATION IN TIME FFT ALGORITHM.
3          ;
4          ;
5          0008          LMAX = 08          ; MAXIMUM FFT LENGTH IS 2*LMAX.
6          ;
7          F000          TWSTAD = 0F000      ; TWIDDLE FACTOR TABLE START
8          ; ADDRESS.
9          0200          DSTAD = 0200        ; DATA STORAGE AREA START
10         ; ADDRESS.
11         ;
12         01C0          NUMB = W(01C0)       ; NUMBER OF DATA POINTS TO BE
13         ; TRANSFORMED.
14         01C2          L1 = W(01C2)        ; NUMB IS 2*L1.
15         01C4          LSHIFT = W(01C4)    ; LSHIFT = LMAX - L1. IT IS A SHIFT
16         ; FACTOR NEEDED TO COMPUTE THE
17         ; ADDRESS REQUIRED FOR TWIDDLE
18         ; FACTOR LOCKUP.
19         01C6          NBFLY = W(01C6)     ; NUMBER OF BUTTERFLIES PER
20         ; TWIDDLE FACTOR PER STAGE.
21         01C8          ISTEP = W(01C8)     ; IF X(1) AND X(J) ARE INVOLVED
22         ; IN A BUTTERFLY, THEN J=I+ISTEP.
23         ; IT IS ALSO THE NUMBER OF TWIDDLE
24         ; FACTORS IN A STAGE.
25         01CA          ILEAP = W(01CA)     ; IF X(I) IS THE FIRST DATA VALUE
26         ; FOR THE FIRST BUTTERFLY FOR A
27         ; GIVEN TWIDDLE FACTOR, THEN THE
28         ; SUBSEQUENT BUTTERFLIES FOR THAT
29         ; TWIDDLE FACTOR HAVE AS THE FIRST
30         ; DATA VALUE X(I+N*ILEAP).
31         01CC          WESTEP = W(01CC)    ; TWIDDLE FACTOR EXPONENT STEP.
32         ; THE TWIDDLE FACTORS FOR A GIVEN
33         ; STAGE ARE W*(I*WESTEP).
34         01CE          NSTG = W(01CE)      ; FFT STAGE BEING EVALUATED.
35         ;
36         01D0          ISTART = W(01D0)    ; INDEX OF THE FIRST DATA VALUE
37         ; FOR THE FIRST BUTTERFLY FOR A
38         ; GIVEN TWIDDLE FACTOR.
39         01D2          WEXP = W(01D2)     ; EXPONENT VALUE FOR A GIVEN
40         ; TWIDDLE FACTOR.
41         01D4          NIWD = W(01D4)     ; TWIDDLE FACTOR BEING EVALUATED.
42         ;
43         01D6          COSTH = W(01D6)     ; COSINE PART OF TWIDDLE FACTOR.
44         ;
45         01D8          SINTH = W(01D8)     ; SINE PART OF TWIDDLE FACTOR.
46         ;
47         01DA          M1 = W(01DA)        ; INDEX OF FIRST DATA VALUE FOR
48         ; A BUTTERFLY.
49         01DC          NBCNT = W(01DC)     ; BUTTERFLY BEING EVALUATED.
50         ;
51         01DE          RLADDR = W(01DE)   ; ADDRESS OF REAL PART OF FIRST

```

```

52                                     ; DATA VALUE INVOLVED IN A BUTTERFLY.
53     01E0                           R2ADDR = W(01E0)   ; ADDRESS OF REAL PART OF SECOND
54                                     ; DATA VALUE INVOLVED IN A BUTTERFLY.
55     01E2                           XR1 = W(01E2)     ; REAL PART OF FIRST DATA VALUE
56                                     ; INVOLVED IN A BUTTERFLY.
57     01E4                           XI1= W(01E4)     ; IMAGINARY PART OF ABOVE.
58                                     ;
59     01E6                           XR2 = W(01E6)     ; REAL PART OF SECOND DATA VALUE
60                                     ; INVOLVED IN A BUTTERFLY.
61     01E8                           XI2 = W(01E8)     ; IMAGINARY PART OF ABOVE.
62                                     ;
63     01EA                           TEMPR = W(01EA)   ; TEMPORARY STORAGE USED IN
64                                     ; A BUTTERFLY.
65     01EC                           TEMPI = W(01EC)   ; SAME AS ABOVE.
66                                     ;
67     01EE                           MTEMP = W(01EE)   ; TEMPORARY STORAGE USED IN SMULT.
68                                     ;
69                                     .INCLD TSTDAT.MAC
70                                     .INCLD TWDTBL.MAC
71                                     ;
72                                     ;
73     F400                             . = OF400
74     TSTFFT:
75     F400 B701FOC4                     LD SP, 01F0
76     F404 832001COAB                   LD NUMB, 020           ; 32 POINT FFT.
77     F409 830501C2AB                   LD L1, 05             ; 32 = 2^5.
78     F40E 3049                         JSR FFT              ; COMPUTE FFT.
79     F410 40                            NOP
80     F411 31C4                         JSR IFFT
81     F413 40                            NOP
82     F414 61                            JP .-1
83                                     ;
84                                     ;
85                                     ; THIS SUBROUTINE IMPLEMENTS A BIT REVERSED COUNTER AS NEEDED FOR
86                                     ; DATA SHUFFLING IN THE FFT ROUTINE. THE ALGORITHM IS BASED ON
87                                     ; THE DESCRIPTION IN:
88                                     ;   RABINER AND GOLD,
89                                     ;   THEORY AND APPLICATIONS OF DIGITAL SIGNAL PROCESSING,
90                                     ;   PRENTICE-HALL, 1975.
91                                     ;
92                                     ; ON INPUT, X CONTAINS THE PREVIOUS BIT REVERSED COUNTER VALUE.
93                                     ; THE NEXT BIT REVERSED OUTPUT IS RETURNED IN X.
94                                     ; A IS LOST, B AND K ARE PRESERVED.
95                                     ;
96                                     .LOCAL
97     BRCNTR:
98     F415 B601COA8                       LD A, NUMB           ; GET NUMBER OF DATA SAMPLES
99                                     ; TO BE TRANSFORMED.
100    F419 C7                             SHR A               ; DIVIDE BY 2.
101    $REPEAT:
102    F41A 96CEFD                         IFGT A, X           ; IS BIT BEING TESTED A 0 ?

```

```

103 F41D 47          JP $FOUND          ; YES, SO STOP CHECKING.
104                  ; GET HERE MEANS BIT BEING
105                  ; CHECKED IS 1.
106 F41E 02          SET C
107 F41F A0C8CEE8    SUBC X, A          ; ZERO OUT THE BIT.
108 F423 C7          SHR A              ; UPDATE BIT LOCATOR.
109 F424 6A          JP $REPEAT
110                  $FOUND:
111 F425 A0C8CEF8     ADD X, A
112 F429 3C          RET
113                  .LOCAL
114                  ;
115                  ;
116                  ;
117                  ; THIS SUBROUTINE MULTIPLIES TWO 16-BIT 2'S COMPLEMENT INTEGERS AND RETURNS
118                  ; THE UPPER HALF OF THE RESULT. THE MULTIPLICAND IS IN A, AND THE MULTIPLIER
119                  ; IN W(B). THE RESULT IS RETURNED IN A. ONE TEMPORARY WORD OF STORAGE,
120                  ; ADDRESSED AS MTEMP IS USED.
121                  ;
122                  SMULT:
123 F42A 830001EEAB   LD MTEMP, 0        ; CLEAR TEMPORARY STORAGE.
124 F42F A9CC         INC B              ; B NOW POINTS TO UPPER BYTE
125                  ; OF MULTIPLIER.
126 F431 17          IF M(B).7          ; IS IT NEGATIVE ?
127 F432 B601EEAB    ST A, MTEMP       ; THEN SAVE MULTIPLICAND IN MTEMP.
128 F436 AACCC       DECSZ B          ; B INTO WORD POINTER.
129 F438 40          NOP
130 F439 B601EEAE    X A, MTEMP        ; SWAP A AND MTEMP.
131 F430 B601EFL7    IF M(($MTEMP)+1).7 ; IS MULTIPLICAND NEGATIVE ?
132 F441 F8          ADD A, W(B)        ; THEN ACCUMULATE MULTIPLIER.
133 F442 B601EEAE    X A, MTEMP
134 F446 FE          MULT A, W(B)       ; UNSIGNED MULTIPLY.
135 F447 AECE        X A, X          ; UPPER HALF IN A.
136 F449 02          SET C
137 F44A B601EEEB    SUBC A, MTEMP
138 F44E E7          SHL A
139 F44F 96CF17     IF H(X).7
140 F452 04          INC A
141 F453 E7          SHL A
142 F454 96CF16     IF H(X).6
143 F457 04          INC A
144 F458 3C          RET
145                  ;
146                  ;
147                  ;
148                  ; THIS SUBROUTINE IMPLEMENTS THE FIXED POINT RADIX-2 DECIMATION-IN-TIME
149                  ; FFT ALGORITHM. THE DATA IS INITIALLY PUT IN THE BIT REVERSED ORDER, AND
150                  ; THEN THE FFT IS COMPUTED. FOR THE THEORY BEHIND THE ALGORITHM, CONSULT:
151                  ;
152                  ; 1. OPPENHEIM AND SCHAFFER, DIGITAL SIGNAL PROCESSING,
153                  ; PRENTICE-HALL.

```

```

154      ;
155      ;           2. RABINER AND GOLD, THEORY AND APPLICATIONS OF DIGITAL SIGNAL
156      ;           PROCESSING, PRENTICE-HALL, 1975.
157      ;
158      ; THE ALGORITHM USED CLOSELY FOLLOWS THE FORTRAN PROGRAM FOREA IN
159      ;
160      ;           3. PROGRAMS FOR DIGITAL SIGNAL PROCESSING, IEEE.
161      ;
162      ;
163      FFT:
164      ;
165      ; FIRST PUT THE DATA IN BIT REVERSED ORDER.
166      ;
167 F459 00          CLR A
168 F45A ABCC       ST A, B           ; SET UP NORMAL COUNTER.
169 F45C ABCE       ST A, X           ; SET UP BIT REVERSED COUNTER.
170 F45E A401COCAAB LD K, NUMB       ; K HAS NUMBER OF DATA POINTS.
171
172 F463 A0CCCEFD   REVL P:          IFGT X, B           ; IS BIT REV CNTR → NORM CNTR ?
173 F467 42         JP SWAP          ; YES, SO SWAP DATA.
174 F468 9421       JMP COUNT        ; NO SO INCREMENT COUNT.
175
176 F46A AFCC       SWAP:           PUSH B
177 F46C AFCE       PUSH X
178 F46E A8CC       LD A, B         ; INDEX VALUE I IS IN A.
179 F470 E7        SHL A
180 F471 E7        SHL A
181 F472 B80200    ADD A, DISTAD     ; GET ADDR. OF XR(I).
182 F475 ABCC       ST A, B         ; SAVE IT IN B.
183 F477 A8CE       LD A, X         ; INDEX VALUE J IS IN A.
184 F479 E7        SHL A
185 F47A E7        SHL A
186 F47B B80200    ADD A, DISTAD     ; GET ADDR. OF XR(J).
187 F47E ABCE       ST A, X         ; SAVE IT IN X.
188 F480 E4        LD A, W(B)       ; A ← XR(I).
189 F481 F1        X A, W(X+)       ; A ← XR(J), XR(J) ← XR(I).
190 F482 E1        XS A, W(B+)     ; A ← XR(I), XR(I) ← XR(J).
191 F483 40        NOP
192 F484 E4        LD A, W(B)       ; A ← XI(I).
193 F485 F5        X A, W(X)       ; A ← XI(J), XI(J) ← XI(I).
194 F486 E6        ST A, W(B)     ; XI(I) ← XI(J).
195 F487 3FCE      POP X
196 F489 3FCC      POP B
197
198      ;
199      ; COUNT:
200      ;
201 F48B AACA       DECSZ K         ; DONE ?
202 F48D 41        JP UPIT         ; NO GO DO SOME MORE.
203 F48E 46        JP DOFFT
204
205      UPIT:
206 F48F 347A      JSR BRCNTR       ; COUNT UP ON BIT REV CNTR.

```



```

205 F491 A9CC          INC B          ; COUNT UP ON NORMAL CNTR.
206 F493 9530         JMP REVL P
207                   DOFFT:
208                   ;
209                   ; DATA IS NOW STORED IN THE BIT REVERSED ORDER. COMPUTE THE FFT.
210                   ;
211 F495 9008         LD A, LMAX        ; A HAS MAX FFT EXPONENT.
212 F497 04          INC A
213 F498 04          INC A
214 F499 02          SET C
215 F49A B601C2EB     SUBC A, L1        ; COMPUTE LSHIFT.
216 F49E B601C4AB     ST A, LSHIFT
217 F4A2 B601COA8     LD A, NUMB
218 F4A6 C7          SHR A
219 F4A7 B601C8AB     ST A, NBFY        ; INITIALIZE NBFY.
220 F4AB B601CCAB     ST A, WESTEP     ; INITIALIZE WESTEP.
221 F4AF 830101C8AB   LD ISTEP, 01     ; INITIALIZE ISTEP.
222 F4B4 830201CAAB   LD ILEAP, 02     ; INITIALIZE ILEAP.
223                   ;
224                   ; SET UP L1 STAGES OF BUTTERFLIES.
225                   ;
226 F4B9 B601C2AB     LD A, L1
227 F4BD B601CEAB     ST A, NSTG        ; LOOP L1 TIMES.
228                   LOOP1:
229                   ;
230 F4C1 00          CLR A
231 F4C2 B601DOAB     ST A, ISTART     ; INITIALIZE ISTART FOR EACH STAGE.
232 F4C6 B601D2AB     ST A, WEXP        ; INITIALIZE WEXP.
233                   ;
234                   ; SET UP ISTEP LOOPS OF TWIDDLE FACTORS.
235                   ;
236 F4CA B601C8AB     LD A, ISTEP
237 F4CE B601D4AB     ST A, NTWD        ; LOOP ISTEP TIMES.
238                   LOOP2:
239                   ;
240                   ; LOOK UP THE TWIDDLE FACTOR.
241                   ;
242 F4D2 A401C4CAAB   LD K, LSHIFT     ; SHIFT LEFT LSHIFT TIMES.
243 F4D7 B601D2AB     LD A, WEXP
244                   GADLP:
245 F4DB E7          SHL A
246 F4DC AACA        DECSZ K          ; DONE SHIFTING ?
247 F4DE 63          JP GADLP         ; NO SO DO MORE.
248 F4DF B8F000      ADD A, TWSTAD     ; ADD STARTING ADDR OF TWIDDLE
249                   ; FACTOR TABLE.
250 F4E2 ABCE        ST A, X          ; TWIDDLE FACTOR ADDR IN X.
251 F4E4 F0          LD A, W(X+)      ; GET COS(THETA).
252 F4E5 B601D6AB     ST A, COSTH
253 F4E9 F4          LD A, W(X)      ; GET SIN(THETA).
254 F4EA 01          COMP A
255 F4EB 04          INC A          ; MAKE IT NEGATIVE.

```

```

256 F4EC B601D8AB          ST A, SINTH
257                          ;
258 F4FO A501D001DAAB      LD M1, ISTART          ; INITIALIZE M1.
259                          ;
260                          ; SET UP NBFLY BUTTERFLIES FOR THIS TWIDDLE FACTOR.
261                          ;
262 F4F6 A501C601DCAB      LD NBCNT, NBFLY          ; LOOP NBFLY TIMES.
263 LOOP3:
264 F4FC B601DAAB          LD A, M1              ; GET INDEX OF X(I).
265 F500 E7                SHL A
266 F501 E7                SHL A
267 F502 B80200           ADD A, DISTAD        ; ADDR. OD XR(I).
268 F505 B601DEAB        ST A, R1ADDR
269 F509 ABCE             ST A, X
270 F50B F0              LD A, W(X+)          ; A ← XR(I).
271 F50C B601E2AB        ST A, XR1            ; STORE IN XR1.
272 F510 F4              LD A, W(X)          ; A ← XI(I).
273 F511 B601E4AB        ST A, XI1            ; STORE IN XI1.
274 F515 B601DAA8        LD A, M1
275 F519 B601CBF8        ADD A, ISTEP        ; GET INDEX OF X(J).
276 F51D E7              SHL A
277 F51E E7              SHL A
278 F51F B80200           ADD A, DISTAD        ; ADDR. OF XR(J).
279 F522 B601E0AB        ST A, R2ADDR
280 F526 ABCE             ST A, X
281 F528 F0              LD A, W(X+)          ; A ← XR(J).
282 F529 B601E6AB        ST A, XR2            ; STORE IN XR2.
283 F520 F4              LD A, W(X)          ; A ← XI(J).
284 F52E B601E8AB        ST A, XI2            ; STORE IN XI2.
285
286 F532 B201E6          LD B, #XR2          ; B ← ADDR(XR2).
287 F535 B601D6A8        LD A, COSTH          ; A ← COS(THETA).
288 F539 350F           JSR SMULT          ; COMPUTE XR(J)*COS(THETA).
289 F53B B601EAAB        ST A, TEMPR          ; SAVE IN TEMPR.
290 F53F B601D8A8        LD A, SINTH          ; A ← SIN(THETA).
291 F543 3519           JSR SMULT          ; COMPUTE XR(J)*SIN(THETA).
292 F545 B601ECAB        ST A, TEMPI          ; SAVE IN TEMPI.
293 F549 B201E8          LD B, #XI2          ; B ← ADDR(XI2).
294 F54C B601D8A8        LD A, SINTH          ; A ← SIN(THETA).
295 F550 3526           JSR SMULT          ; COMPUTE XI(J)*SIN(THETA).
296 F552 01              COMP A
297 F553 04              INC A
298 F554 B601EAF8        ADD A, TEMPR          ; COMPUTE XR(J)*COS(THETA) -
299                          ; XI(J)*SIN(THETA).
300 F558 B601EAAB        ST A, TEMPR
301 F55C B601D6A8        LD A, COSIN          ; A ← COS(THETA).
302 F560 3536           JSR SMULT          ; COMPUTE XI(J)*COS(THETA).
303 F562 B601ECF8        ADD A, TEMPI          ; COMPUTE XR(J)*SIN(THETA) +
304                          ; XI(J)*COS(THETA).
305 F566 B601ECAB        ST A, TEMPI
306

```

```

307 ;
308 F56A A401DECEAB ; LD X, RIADDR ; X ← ADDR(XR(I)).
309 F56F A401EOCCAB ; LD B, R2ADDR ; B ← ADDR(XR(J)).
310 F574 F0 ; LD A, W(X+) ; A ← XR(I).
311 F575 02 ; SET C
312 F576 B601EAEB ; SUBC A, TEMPR ; A ← XR(I) - TEMPR.
313 F57A E1 ; XS A, W(B+)
314 F57B 40 ; NOP
315 F57C F2 ; LD A, W(X-) ; A ← XI(I).
316 F57D 02 ; SET C
317 F57E B601ECEB ; SUBC A, TEMPI ; A ← XI(J) - TEMPI.
318 F582 E6 ; ST A, W(B)
319 F583 F4 ; LD A, W(X) ; A ← XR(I).
320 F584 B601EAF8 ; ADD A, TEMPR ; A ← XR(I) + TEMPR.
321 F588 F1 ; X A, W(X+)
322 F589 F4 ; LD A, W(X) ; A ← XI(I).
323 F58A B601ECF8 ; ADD A, TEMPI ; A ← XI(I) + TEMPI.
324 F58E F6 ; ST A, W(X)
325 ;
326 F58F A501CA01DAF8 ; ADD M1, ILEAP ; UPDATE M1 FOR NEXT LOOP.
327 ;
328 F595 B601DCAA ; DECSZ NBCNT ; DONE WITH ALL BUTTERFLIES
329 ; ; FOR THIS TWIDDLE FACTOR ?
330 F599 959D ; JMP LOOP3 ; NO, SO GO DO SOME MORE.
331 ;
332 ;
333 F59B B601DOA9 ; INC ISTART ; SET UP STARTING INDEX FOR
334 ; ; NEXT TWIDDLE FACTOR.
335 F59F A501CC01D2F8 ; ADD WEXP, WESTEP ; UPDATE TWIDDLE FACTOR
336 ; ; EXPONENT VALUE.
337 ;
338 F5A5 B601D4AA ; DECSZ NIWD ; DONE WITH ALL TWIDDLES
339 ; ; FOR THIS STAGE ?
340 F5A9 95D7 ; JMP LOOP2 ; NO, SO GO DO SOME MORE.
341 ;
342 ;
343 F5AB B601CAA8 ; LD A, ILEAP
344 F5AF E7 ; SHL A
345 F5B0 B601CAAB ; ST A, ILEAP ; UPDATE ILEAP FOR NEXT STAGE.
346 F5B4 B601C8A8 ; LD A, ISTEP
347 F5B8 E7 ; SHL A
348 F5B9 B601C8AB ; ST A, ISTEP ; UPDATE ISTEP FOR NEXT STAGE.
349 F5BD B601C6A8 ; LD A, NBFLY
350 F5C1 C7 ; SHR A
351 F5C2 B601C6AB ; ST A, NBFLY ; UPDATE NBFLY FOR NEXT STAGE.
352 F5C6 B601CCA8 ; LD A, WESTEP
353 F5CA C7 ; SHR A
354 F5CB B601CCAB ; ST A, WESTEP ; UPDATE WESTEP FOR NEXT STAGE.
355 ;
356 F5CF B601CEAA ; DECSZ NSTG ; DONE WITH ALL STAGES ?
357 F5D3 B4FEEB + ; JMP LOOP1 ; NO SO GO DO SOME MORE.

```

```

358      ;
359 F5D6 3C      RET      ; ALL OVER.
360      ;
361      ; THE CODE BELOW IS FOR THE INVERSE FFT. THE ONLY DIFFERENCE IS THAT
362      ; THE TWIDDLE FACTORS ARE USED A LITTLE DIFFERENTLY, AND A FINAL SCALING BY
363      ; 1/NUMB IS DONE.
364      IFFT:
365      ;
366      ; FIRST PUT THE DATA IN BIT REVERSED ORDER.
367      ;
368 F5D7 00      CLR A
369 F5D8 ABCC    ST A, B      ; SET UP NORMAL COUNTER.
370 F5DA ABCE    ST A, X      ; SET UP BIT REVERSED COUNTER.
371 F5DC A401COCAAB LD K, NUMB  ; K HAS NUMBER OF DATA POINTS.
372      IREVLV:
373 F5E1 A0CCCEFD IFGT X, B      ; IS BIT REV CNTR → NORM CNTR ?
374 F5E5 42      JP ISWAP   ; YES, SO SWAP DATA.
375 F5E6 9421    JMP ICOUNT  ; NO SO INCREMENT COUNT.
376      ISWAP:
377 F5E8 AFCC    PUSH B
378 F5EA AFCE    PUSH X
379 F5EC ABCC    LD A, B      ; INDEX VALUE I IS IN A.
380 F5EE E7      SHL A
381 F5EF E7      SHL A
382 F5F0 B80200  ADD A, DTSTAD  ; GET ADDR. OF XR(I).
383 F5F3 ABCC    ST A, B      ; SAVE IT IN B.
384 F5F5 ABCE    LD A, X      ; INDEX VALUE J IS IN A.
385 F5F7 E7      SHL A
386 F5F8 E7      SHL A
387 F5F9 B80200  ADD A, DTSTAD  ; GET ADDR. OF XR(J).
388 F5FC ABCE    ST A, X      ; SAVE IT IN X.
389 F5FE E4      LD A, W(B)    ; A ← XR(I).
390 F5FF F1      X A, W(X+)   ; A ← XR(J), XR(J) ← XR(I).
391 F600 E1      XS A, W(B+) ; A ← XR(I), XR(I) ← XR(J).
392 F601 40      NOP
393 F602 E4      LD A, W(B)    ; A ← XI(I).
394 F603 F5      X A, W(X)    ; A ← XI(J), XI(J) ← XI(I).
395 F604 E6      ST A, W(B)    ; XI(I) ← XI(J).
396 F605 3FCC    POP X
397 F607 3FCC    POP B
398      ;
399      ICOUNT:
400      ;
401 F609 AACA    DECSZ K      ; DONE ?
402 F60B 41      JP IUPIT   ; NO GO DO SOME MORE.
403 F60C 46      JP DOIFFT
404      IUPIT:
405 F60D 35F8    JSR BRCNTR  ; COUNT UP ON BIT REV CNTR.
406 F60F A9CC    INC B      ; COUNT UP ON NORMAL CNTR.
407 F611 9530    JMP IREVLV
408      DOIFFT:

```

```

409      ;
410      ; DATA IS NOW STORED IN THE BIT REVERSED ORDER. COMPUTE THE FFT.
411      ;
412 F613 9008      LD A, LMAX      ; A HAS MAX FFT EXPONENT.
413 F615 04      INC A
414 F616 04      INC A
415 F617 02      SET C
416 F618 B601C2EB SUBC A, L1      ; COMPUTE LSHIFT.
417 F61C B601C4AB ST A, LSHIFT
418 F620 B601C0A8 LD A, NUMB
419 F624 C7      SHR A
420 F625 B601C6AB ST A, NBFLY    ; INITIALIZE NBFLY.
421 F629 B601CCAB ST A, WESTEP   ; INITIALIZE WESTEP.
422 F62D 830101C8AB LD ISTEP, 01   ; INITIALIZE ISTEP.
423 F632 830201CAAB LD ILEAP, 02   ; INITIALIZE ILEAP.
424      ;
425      ; SET UP L1 STAGES OF BUTTERFLIES.
426      ;
427 F637 B601C2AB LD A, L1
428 F63B B601CEAB ST A, NSTG      ; LOOP L1 TIMES.
429      ILOOP1:
430      ;
431 F63F 00      CLR A
432 F640 B601D0AB ST A, ISTART   ; INITIALIZE ISTART FOR EACH STAGE.
433 F644 B601D2AB ST A, WEXP     ; INITIALIZE WEXP.
434      ;
435      ; SET UP ISTEP LOOPS OF TWIDDLE FACTORS.
436      ;
437 F648 B601C8A8 LD A, ISTEP
438 F64C B601D4AB ST A, NIWD      ; LOOP ISTEP TIMES.
439      ILOOP2:
440      ;
441      ; LOOK UP THE TWIDDLE FACTOR.
442      ;
443 F650 A401C4CAAB LD K, LSHIFT   ; SHIFT LEFT LSHIFT TIMES.
444 F655 B601D2AB LD A, WEXP
445      IGADLP:
446 F659 E7      SHL A
447 F65A AACA    DECSZ K      ; DONE SHIFTING ?
448 F65C 63      JP IGADLP    ; NO DO SOME MORE.
449 F65D B8F000  ADD A, TWSTAD   ; ADD STARTING ADDR OF TWIDDLE
450      ;          ; FACTOR TABLE.
451 F660 ABCE    ST A, X      ; TWIDDLE FACTOR ADDR IN X.
452 F662 F0      LD A, W(X+)   ; GET COS(THETA).
453 F663 B601D6AB ST A, COSTH
454 F667 F4      LD A, W(X)   ; GET SIN(THETA).
455 F668 B601D8AB ST A, SINTH
456      ;
457 F66C A501D001DAAB LD M1, ISTART   ; INITIALIZE M1.
458      ;
459      ; SET UP NBFLY BUTTERFLIES FOR THIS TWIDDLE FACTOR.

```

```

460 ;
461 F672 A501C601DCAB LD NBCNT, NBFly ; LOOP NBFly TIMES.
462 ILOOP3:
463 F678 B601DAA8 LD A, M1 ; GET INDEX OF X(I).
464 F67C E7 SHL A
465 F67D E7 SHL A
466 F67E B80200 ADD A, DTSTAD ; ADDR. OD XR(I).
467 F691 B601DEA8 ST A, R1ADDR
468 F685 ABCE ST A, X
469 F687 F0 LD A, W(X+) ; A ← XR(I).
470 F688 B601E2AB ST A, XR1 ; STORE IN XR1.
471 F68C F4 LD A, W(X) ; A ← XI(I).
472 F68D B601E4AB ST A, XI1 ; STORE IN XI1.
473 F691 B601DAA8 LD A, M1
474 F695 B601CBF8 ADD A, ISTEP ; GET INDEX OF X(J).
475 F699 E7 SHL A
476 F69A E7 SHL A
477 F69B B80200 ADD A, DTSTAD ; ADDR. OF XR(J).
478 F69E B601E0AB ST A, R2ADDR
479 F6A2 ABCE ST A, X
480 F6A4 F0 LD A, W(X+) ; A ← XR(J).
481 F6A5 B601E6AB ST A, XR2 ; STORE IN XR2.
482 F6A9 F4 LD A, W(X) ; A ← XI(J).
483 F6AA B601E8AB ST A, XI2 ; STORE IN XI2.
484 ;
485 F6A8 B201E6 LD B, #XR2 ; B ← ADDR(XR2).
486 F6B1 B601D8A8 LD A, COSTH ; A ← COS(THETA).
487 F6B5 368B JSR SMULT ; COMPUTE XR(J)*COS(THETA).
488 F6B7 B601EAAB ST A, TEMPR ; SAVE IN TEMPR.
489 F6BB B601D8A8 LD A, SINTH ; A ← SIN(THETA).
490 F6BF 3695 JSR SMULT ; COMPUTE XR(J)*SIN(THETA).
491 F6C1 B601ECAB ST A, TEMPI ; SAVE IN TEMPI.
492 F6C5 B201E8 LD B, #XI2 ; B ← ADDR(XI2).
493 F6C8 B601D8A8 LD A, SINTH ; A ← SIN(THETA).
494 F6CC 36A2 JSR SMULT ; COMPUTE XI(J)*SIN(THETA).
495 F6CE 01 COMP A
496 F6CF 04 INC A
497 F6D0 B601EAF8 ADD A, TEMPR ; COMPUTE XR(J)*COS(THETA) -
498 ; XI(J)*SIN(THETA).
499 F6D4 B601EAAB ST A, TEMPR
500 F6D8 B601D8A8 LD A, COSTH ; A ← COS(THETA).
501 F6DC 36B2 JSR SMULT ; COMPUTE XI(J)*COS(THETA).
502 F6DE B601ECF8 ADD A, TEMPI ; COMPUTE XR(J)*SIN(THETA) +
503 ; XI(J)*COS(THETA).
504 F6E2 B601ECAB ST A, TEMPI
505 ;
506 ;
507 F6E6 A401DECEAB LD X, R1ADDR ; X ← ADDR(XR(I)).
508 F6EB A401EOCCAB LD B, R2ADDR ; B ← ADDR(XR(J)).
509 F6F0 F0 LD A, W(X+) ; A ← XR(I).
510 F6F1 02 SET C

```

```

511 F6F2 B601EAE8      SUBC A, TEMPR          ; A ← XR(I) - TEMPR.
512 F6F6 E1            XS A, W(B+)
513 F6F7 40           NOP
514 F6F8 F2          LD A, W(X-)          ; A ← XI(I).
515 F6F9 02          SET C
516 F6FA B601ECEB     SUBC A, TEMPI          ; A ← XI(J) - TEMPI.
517 F6FE E6          ST A, W(B)
518 F6FF F4          LD A, W(X)          ; A ← XR(I).
519 F700 B601EAF8     ADD A, TEMPR          ; A ← XR(I) + TEMPR.
520 F704 F1          X A, W(X+)
521 F705 F4          LD A, W(X)          ; A ← XI(I).
522 F706 B601ECF8     ADD A, TEMPI          ; A ← XI(I) + TEMPI.
523 F70A F6          ST A, W(X)
524
525 F70B A501CA01DAF8  ADD M1, ILEAP         ; UPDATE M1 FOR NEXT LOOP.
526
527 F711 B601DCAA     DECSZ NBCNT          ; DONE WITH ALL BUTTERFLIES
528                                     ; FOR THIS TWIDDLE FACTOR ?
529 F715 959D        JMP ILOOP3           ; NO, SO GO DO SOME MORE.
530
531
532 F717 B601D0A9     INC ISTART          ; SET UP STARTING INDEX FOR
533                                     ; NEXT TWIDDLE FACTOR.
534 F71B A501CC01D2F8  ADD WEXP, WESTEP     ; UPDATE TWIDDLE FACTOR
535                                     ; EXPONENT VALUE.
536
537 F721 B601D4AA     DECSZ NTWD          ; DONE WITH ALL TWIDDLES
538                                     ; FOR THIS STAGE ?
539 F725 95D5        JMP ILOOP2           ; NO, SO GO DO SOME MORE.
540
541
542 F727 B601CAA8     LD A, ILEAP
543 F72B E7           SHL A
544 F72C B601CAAB     ST A, ILEAP         ; UPDATE ILEAP FOR NEXT STAGE.
545 F730 B601C8A8     LD A, ISTEP
546 F734 E7           SHL A
547 F735 B601C8AB     ST A, ISTEP         ; UPDATE ISTEP FOR NEXT STAGE.
548 F739 B601C6A8     LD A, NBFLY
549 F73D C7           SHR A
550 F73E B601C6AB     ST A, NBFLY        ; UPDATE NBFLY FOR NEXT STAGE.
551 F742 B601CCAB     LD A, WESTEP
552 F746 C7           SHR A
553 F747 B601CCAB     ST A, WESTEP       ; UPDATE WESTEP FOR NEXT STAGE.
554
555 F748 B601CEAA     DECSZ NSTG          ; DONE WITH ALL STAGES ?
556 F74F B4FEED      JMP ILOOP1         ; NO SO GO DO SOME MORE.
557
558
559                                     ; DO THE FINAL SCALING OF THE DATA BY 1/NUMB.
560
561 F752 B04000      LD A, 04000        ; A ← 1.0

```

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

PAGE: 12

```

562 F755 A401C2CAAB      LD K, L1          ; K ← L1.
563                      SCALLP:
564 F75A C7              SHR A            ; DIVIDE BY 2.
565 F75B AACA            DECSZ K
566 F75D 63              JP SCALLP
567                      ;
568                      ; GET HERE MEANS A IS 1/(2*L1).
569 F75E B601EAAB        ST A, TEMPR      ; SAVE IT IN TEMPR.
570 F762 A501C001DCAB    LD NBCNT, NUMB   ; LOOP COUNTER.
571 F768 B20200          LD B, DTSTAD     ; B ← ADDR(XR(0)).
572                      SCALIT:
573 F76B B601EAAB        LD A, TEMPR
574 F76F 3745            JSR SMULT        ; A ← XR(I)*(1/NUMB).
575 F771 E1              XS A, W(B+)      ; XR(I) ← XR(I)*(1/NUMB).
576 F772 40              NOP
577 F773 B601EAAB        LD A, TEMPR      ; A ← 1/NUMB.
578 F777 374D            JSR SMULT        ; A ← X1(I)*(1/NUMB).
579 F779 E1              XS A, W(B+)      ; X1(I) ← X1(I)*(1/NUMB).
580 F77A 40              NOP
581 F77B B601DCAA        DECSZ NBCNT     ; DONE ?
582 F77F 74              JP SCALIT        ; NO DO SOME MORE.
583 F780 3C              RET              ; ALL OVER.
584                      ;
585 FFFE 00F4            .END TSIFFT

```

PAGE: 13

SYMBOL TABLE

A	00C8 W	B	00CC W	BRCNTR F415	COSTN 01D6 W
COUNT	F48B	DOFFT	F495	DOIFFT F613	DTSTAD 0200
FFT	F459	GADLP	F40B	ICOUNT F609	IFFT F507
IGADLP	F659	ILEAP	01CA W	ILOOP1 F63F	ILOOP2 F650
ILOOP3	F678	IREVLP	F5E1	ISTART 01D0 W	ISTEP 01C8 W
ISWAP	F5E8	IUPIT	F600	K 00CA W	L1 01C2 W
LMAX	0008	LOOP1	F4C1	LOOP2 F4D2	LOOP3 F4FC
LSHIFT	01C4 W	M1	01DA W	MTEMP 01EE W	NBCNT 01DC W
NBFLY	01C6 W	NSTG	01CE W	NTWD 01D4 W	NUMB 01C0 W
PC	00C6 W	RLADDR	01DE W	R2ADDR 01E0 W	REVLP F463
SCALIT	F76B	SCALLP	F75A	SINTH 01D8 W	SMULT F42A
SP	00C4 W	SWAP	F46A	TEMPI 01EC W	TEMPR 01EA W
TSIFFT	F400	TWSTAD	F000	UPIT F48F	WESTEP 01CC W
WEXP	01D2 W	X	00CE W	XI1 01E4 W	XI2 01E8 W
XR1	01E2 W	XR2	01E6 W	\$FOUND F425	\$REPEA F41A

MACRO TABLE

NO WARNING LINES

NO ERROR LINES

2307 ROM BYTES USED

SOURCE CHECKSUM = E9FC

OBJECT CHECKSUM = 28FC

INPUT FILE C:FFT.MAC

LISTING FILE C:FFT.PRN

OBJECT FILE C:FFT.LM

APPENDIX B

Twiddle Factor Table

```

;
; TWIDDLE FACTOR TABLE FOR USE IN THE FFT ROUTINES.
;
; TABLE SET FOR MAX FFT LENGTH OF 256.
;
; TABLE STARTS AT F000 AND OCCUPIES 1024 BYTES OF STORAGE.
;
      . = OF000
      .WORD 16384, 0
      .WORD 16379, 402
      .WORD 16364, 804
      .WORD 16340, 1205
      .WORD 16305, 1606
      .WORD 16261, 2006
      .WORD 16207, 2404
      .WORD 16143, 2801
      .WORD 16069, 3196
      .WORD 15986, 3590
      .WORD 15893, 3981
      .WORD 15791, 4370
      .WORD 15679, 4756
      .WORD 15557, 5139
      .WORD 15426, 5520
      .WORD 15286, 5897
      .WORD 15137, 6270
      .WORD 14978, 6639
      .WORD 14811, 7005
      .WORD 14635, 7366
      .WORD 14449, 7723
      .WORD 14256, 8076
      .WORD 14053, 8423
      .WORD 13842, 8765
      .WORD 13623, 9102
      .WORD 13395, 9434
      .WORD 13160, 9760
      .WORD 12916, 10080
      .WORD 12665, 10394
      .WORD 12406, 10702
      .WORD 12140, 11003
      .WORD 11866, 11297
      .WORD 11585, 11585
      .WORD 11297, 11866
      .WORD 11003, 12140
      .WORD 10702, 12406
      .WORD 10394, 12665
      .WORD 10080, 12916
      .WORD 9760, 13160
      .WORD 9434, 13395
      .WORD 9102, 13623
      .WORD 8765, 13842
      .WORD 8423, 14053
      .WORD 8076, 14256
      .WORD 7723, 14449
      .WORD 7366, 14635
      .WORD 7005, 14811
      .WORD 6639, 14978
      .WORD 6270, 15137
      .WORD 5897, 15286
      .WORD 5520, 15426
      .WORD 5139, 15557
      .WORD 4756, 15679
      .WORD 4370, 15791
      .WORD 3981, 15893
      .WORD 3590, 15986
      .WORD 3196, 16069
      .WORD 2801, 16143
      .WORD 2404, 16207
      .WORD 2006, 16261
      .WORD 1606, 16305
      .WORD 1205, 16340
      .WORD 804, 16364
      .WORD 402, 16379
      .WORD 0, 16384
      .WORD -402, 16379
      .WORD -804, 16364
      .WORD -1205, 16340
      .WORD -1606, 16305
      .WORD -2006, 16261
      .WORD -2404, 16207
      .WORD -2801, 16143
      .WORD -3196, 16069
      .WORD -3590, 15986
      .WORD -3981, 15893
      .WORD -4370, 15791
      .WORD -4756, 15679
      .WORD -5139, 15557
      .WORD -5520, 15426
      .WORD -5897, 15286
      .WORD -6270, 15137
      .WORD -6639, 14978
      .WORD -7005, 14811
      .WORD -7366, 14635
      .WORD -7723, 14449
      .WORD -8076, 14256
      .WORD -8423, 14053
      .WORD -8765, 13842
      .WORD -9102, 13623
      .WORD -9434, 13395
      .WORD -9760, 13160
      .WORD -10080, 12916
      .WORD -10394, 12665
      .WORD -10702, 12406
      .WORD -11003, 12140
      .WORD -11297, 11866
      .WORD -11585, 11585
      .WORD -11866, 11297
      .WORD -12140, 11003
      .WORD -12406, 10702
      .WORD -12665, 10394
      .WORD -12916, 10080

```

.WORD -13160, 9760	.WORD -12916, -10080
.WORD -13395, 9434	.WORD -12665, -10394
.WORD -13623, 9102	.WORD -12406, -10702
.WORD -13842, 8765	.WORD -12140, -11003
.WORD -14053, 8423	.WORD -11866, -11297
.WORD -14256, 8076	.WORD -11585, -11585
.WORD -14449, 7723	.WORD -11297, -11866
.WORD -14635, 7366	.WORD -11003, -12140
.WORD -14811, 7005	.WORD -10702, -12406
.WORD -14978, 6639	.WORD -10394, -12665
.WORD -15137, 6270	.WORD -10080, -12916
.WORD -15286, 5897	.WORD -9760, -13160
.WORD -15426, 5520	.WORD -9434, -13395
.WORD -15557, 5139	.WORD -9102, -13623
.WORD -15679, 4756	.WORD -8765, -13842
.WORD -15791, 4370	.WORD -8423, -14053
.WORD -15893, 3981	.WORD -8076, -14256
.WORD -15986, 3590	.WORD -7723, -14449
.WORD -16069, 3196	.WORD -7366, -14635
.WORD -16143, 2801	.WORD -7005, -14811
.WORD -16207, 2404	.WORD -6639, -14978
.WORD -16261, 2006	.WORD -6270, -15137
.WORD -16305, 1606	.WORD -5897, -15286
.WORD -16340, 1205	.WORD -5520, -15426
.WORD -16364, 804	.WORD -5139, -15557
.WORD -16379, 402	.WORD -4756, -15679
.WORD -16384, 0	.WORD -4370, -15791
	.WORD -3981, -15893
	.WORD -3590, -15986
.WORD -16379, -402	.WORD -3196, -16069
.WORD -16364, -804	.WORD -2801, -16143
.WORD -16340, -1205	.WORD -2404, -16207
.WORD -16305, -1606	.WORD -2006, -16261
.WORD -16261, -2006	.WORD -1606, -16305
.WORD -16207, -2404	.WORD -1205, -16340
.WORD -16143, -2801	.WORD -804, -16364
.WORD -16069, -3196	.WORD -402, -16379
.WORD -15986, -3590	.WORD 0, -16384
.WORD -15893, -3981	.WORD 402, -16379
.WORD -15791, -4370	.WORD 804, -16364
.WORD -15679, -4756	.WORD 1205, -16340
.WORD -15557, -5139	.WORD 1606, -16305
.WORD -15426, -5520	.WORD 2006, -16261
.WORD -15286, -5897	.WORD 2404, -16207
.WORD -15137, -6270	.WORD 2801, -16143
.WORD -14978, -6639	.WORD 3196, -16069
.WORD -14811, -7005	.WORD 3590, -15986
.WORD -14635, -7366	.WORD 3981, -15893
.WORD -14449, -7723	.WORD 4370, -15791
.WORD -14256, -8076	.WORD 4756, -15679
.WORD -14053, -8423	.WORD 5139, -15557
.WORD -13842, -8765	.WORD 5520, -15426
.WORD -13623, -9102	.WORD 5897, -15286
.WORD -13395, -9434	.WORD 6270, -15137
.WORD -13160, -9760	.WORD 6639, -14978

.WORD 7005, -14811
.WORD 7366, -14635
.WORD 7723, -14449
.WORD 8076, -14256
.WORD 8423, -14053
.WORD 8765, -13842
.WORD 9102, -13623
.WORD 9434, -13395
.WORD 9760, -13160
.WORD 10080, -12916
.WORD 10394, -12665
.WORD 10702, -12406
.WORD 11003, -12140
.WORD 11297, -11866
.WORD 11585, -11585
.WORD 11866, -11297
.WORD 12140, -11003
.WORD 12406, -10702
.WORD 12665, -10394
.WORD 12916, -10080
.WORD 13160, -9760
.WORD 13395, -9434
.WORD 13623, -9102
.WORD 13842, -8765
.WORD 14053, -8423
.WORD 14256, -8076
.WORD 14449, -7723
.WORD 14635, -7366
.WORD 14811, -7005
.WORD 14978, -6639
.WORD 15137, -6270
.WORD 15286, -5897
.WORD 15426, -5520
.WORD 15557, -5139
.WORD 15679, -4756
.WORD 15791, -4370
.WORD 15893, -3981
.WORD 15986, -3590
.WORD 16069, -3196
.WORD 16143, -2801
.WORD 16207, -2404
.WORD 16261, -2006
.WORD 16305, -1606
.WORD 16340, -1205
.WORD 16364, -804
.WORD 16379, -402

.END

APPENDIX C

Test Data and Expected Results

NATIONAL SEMICONDUCTOR CORPORATION
 HPC CROSS ASSEMBLER, REV: C, 30 JUL 86

PAGE: 1

```

1          ;
2          ;
3          ; TEST DATA FOR FFT ROUTINES.
4          ; OBTAINED FROM : PROGRAMS FOR DIGITAL SIGNAL PROCESSING, IEEE PRESS,
5          ; CHAPTER 1 BY GOLD.
6          ;
7          ;
8          0200          . = 0200
9          0200 0004          .WORD 1024, 0
           0202 0000
10         0204 9A03          .WORD 922, 307
           0206 3301
11         0208 E102          .WORD 737, 553
           020A 2902
12         020C F201          .WORD 498, 719
           020E CF02
13         0210 E800          .WORD 232, 796
           0212 1C03
14         0214 E2FF          .WORD -30, 786
           0216 1203
15         0218 F9FE          .WORD -263, 699
           021A BB02
16         021C 42FE          .WORD -446, 550
           021E 2602
17         0220 C9FD          .WORD -567, 361
           0222 6901
18         0224 96FD          .WORD -618, 155
           0226 9800
19         0228 A5FD          .WORD -603, -46
           022A D2FF
20         022C EFFD          .WORD -529, -222
           022E 22FF
21         0230 67FE          .WORD -409, -359
           0232 99FE
22         0234 FBFE          .WORD -261, -446
           0236 42FE
23         0238 9BFF          .WORD -101, -479
           023A 21FE
24         023C 3500          .WORD 53, -462
           023E 32FE
25         0240 BA00          .WORD 186, -400
           0242 70FE
26         0244 1F01          .WORD 287, -304
           0246 D0FE
27         0248 5E01          .WORD 350, -187
           024A 45FF
28         024C 7301          .WORD 371, -64
           024E C0FF
29         0250 6101          .WORD 353, 54
           0252 3600
30         0254 2001          .WORD 301, 154

```

0256 9A00	
31 0258 E100	.WORD 225, 229
025A E500	
32 025C 8600	.WORD 134, 274
025E 1201	
33 0260 2600	.WORD 38, 287
0262 1F01	
34 0264 CCFF	.WORD -52, 269
0266 0D01	
35 0268 81FF	.WORD -127, 227
026A E300	
36 026C 49FF	.WORD -183, 166
026E A600	
37 0270 2AFF	.WORD -214, 95
0272 5F00	
38 0274 23FF	.WORD -221, 21
0276 1500	
39 0278 33FF	.WORD -205, -48
027A DDFD	
40 027C 55FF	.WORD -171, -104
027E 98FF	
41	;
42	; THESE ARE THE EXPECTED DFT RESULTS.
43	;
44 0280 C702	.WORD 711, 3584
0282 000E	
45 0284 2B0B	.WORD 2859, 8244
0286 3420	
46 0288 9D25	.WORD 9629, -9354
028A 76DB	
47 028C 7707	.WORD 1911, -3926
028E AAFO	
48 0290 8704	.WORD 1159, -2288
0292 10F7	
49 0294 9F03	.WORD 927, -1571
0296 DDF9	
50 0298 3303	.WORD 819, -1167
029A 71FB	
51 029C F502	.WORD 757, -903
029E 79FC	
52 02A0 CE02	.WORD 718, -715
02A2 35FD	
53 02A4 B202	.WORD 690, -572
02A6 C4FD	
54 02A8 9D02	.WORD 669, -457
02AA 37FE	
55 02AC 8C02	.WORD 652, -361
02AE 97FE	
56 02B0 7F02	.WORD 639, -279
02B2 E9FE	
57 02B4 7302	.WORD 627, -205

```
0286 33FF
58 02B8 6902          .WORD 617, -139
   02BA 75FF
59 02BC 6002          .WORD 608, -77
   02BE B3FF
60 02C0 5802          .WORD 600, -18
   02C2 EEFF
61 02C4 5102          .WORD 593, 39
   02C6 2700
62 02C8 4A02          .WORD 586, 95
   02CA 5F00
63 02CC 4302          .WORD 579, 152
   02CE 9800
64 02D0 3C02          .WORD 572, 210
   02D2 0200
65 02D4 3502          .WORD 565, 271
   02D6 0F01
66 02D8 2E02          .WORD 558, 336
   02DA 5001
67 02DC 2702          .WORD 551, 408
   02DE 9801
68 02E0 2002          .WORD 544, 488
   02E2 EB01
69 02E4 1802          .WORD 536, 581
   02E6 4502
70 02E8 1002          .WORD 528, 691
   02EA B302
71 02EC 0702          .WORD 519, 827
   02EE 3B03
72 02F0 FE01          .WORD 510, 1004
   02F2 EC03
73 02F4 F701          .WORD 503, 1248
   02F6 E004
74 02F8 F701          .WORD 503, 1615
   02FA 4F06
75 02FC 1202          .WORD 530, 2241
   02FE C108
76                ;
77                ;
78                ; TEST DATA FOR FFT ROUTINES.
79                ; OBTAINED FROM :
80                ;
81                ;
82 0300 0004          .WORD 1024, 0
   0302 0000
83 0304 9A03          .WORD 922, 307
   0306 3301
84 0308 E102          .WORD 737, 553
   030A 2902
85 030C F201          .WORD 498, 719
   030E CF02
```

86	0310 E800	.WORD 232, 796
	0312 1C03	
87	0314 E2FF	.WORD -30, 786
	0316 1203	
88	0318 F9FE	.WORD -263, 699
	031A BB02	
89	031C 42FE	.WORD -446, 550
	031E 2602	
90	0320 C9FD	.WORD -567, 361
	0322 6901	
91	0324 96FD	.WORD -618, 155
	0326 9B00	
92	0328 A5FD	.WORD -603, -46
	032A D2FF	
93	032C EFFD	.WORD -529, -222
	032E 22FF	
94	0330 67FE	.WORD -409, -359
	0332 99FE	
95	0334 FBFE	.WORD -261, -446
	0336 42FE	
96	0338 9BFF	.WORD -101, -479
	033A 21FE	
97	033C 3500	.WORD 53, -462
	033E 32FE	
98	0340 BA00	.WORD 186, -400
	0342 70FE	
99	0344 1F01	.WORD 287, -304
	0346 D0FE	
100	0348 5E01	.WORD 350, -187
	034A 45FF	
101	034C 7301	.WORD 371, -64
	034E C0FF	
102	0350 6101	.WORD 353, 54
	0352 3600	
103	0354 2D01	.WORD 301, 154
	0356 9A00	
104	0358 E100	.WORD 225, 229
	035A E500	
105	035C 8600	.WORD 134, 274
	035E 1201	
106	0360 2600	.WORD 38, 287
	0362 1F01	
107	0364 CCFF	.WORD -52, 269
	0366 0D01	
108	0368 81FF	.WORD -127, 227
	036A E300	
109	036C 49FF	.WORD -183, 166
	036E A600	
110	0370 2AFF	.WORD -214, 95
	0372 5F00	
111	0374 23FF	.WORD -221, 21

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

PAGE: 5

0376 1500
112 0378 33FF .WORD -205, -48
037A DOFF
113 037C 55FF .WORD -171, -104
037E 98FF
114 ;
115 .END

@
ERROR, OPERAND MUST BE SINGLE VALID SYMBOL NAME

NATIONAL SEMICONDUCTOR CORPORATION
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

PAGE: 6

SYMBOL TABLE

A	00C8 W	B	00CC W	K	00CA W	PC	00C6 W
SP	00C4 W	X	00CE W				

MACRO TABLE

NO WARNING LINES

1 ERROR LINES

384 ROM BYTES USED

SOURCE CHECKSUM = 7A03

OBJECT CHECKSUM = 0705

INPUT FILE C:TSTDAT.MAC

LISTING FILE C:TSTDAT.PRN

Expanding the HPC Address Space

National Semiconductor
Application Note 497
Joe Cocovich



INTRODUCTION

The maximum address range of the HPC family of 16-bit High Performance microControllers is 64k bytes using the external address/data bus to interface with external memory. This application note describes a method to increase the amount of memory in a system to 544k bytes utilizing bank switching techniques. Block diagrams are presented to aid in circuit design. Software examples are given for memory and bank management.

HPC ADDRESSING

Program memory addressing is accomplished by the 16-bit Program Counter on a byte basis (instructions are always fetched a byte at a time). Memory can be addressed as words or bytes directly by instructions or indirectly through the B, X and SP registers. Words are always addressed on even-byte boundaries. The HPC uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The external address/data bus of the HPC is 16 bits wide. This means the maximum address that the bus can hold is FFFF for a maximum address range of 64K bytes (65,536). Keep in mind, this uses the external address/data bus (A0:A15 for Address/Data and B10, 11, 12, 15) for Control.

BANK SWITCHING

If more than 64k of addressing is needed in the HPC system, the following method of increasing memory space can be used. Divide the total address range into two halves (32k bytes each). One half of this address range will be the MAIN memory address space. The MAIN memory address space will contain logical addresses (those addresses which the Program Counter can generate) in the range 8000 to FFFF and is accessed when A15 is a '1'. This includes the Interrupt vectors and the Reset vector memory locations. The other half of the address range will be the BANK memory address space. The BANK memory address space will contain logical addresses in the range 0000 to 7FFF and is accessed when A15 is a '0'. This includes the on-chip I/O, registers, and RAM at locations 0000 to 01FF.

Now, four additional address lines are created using Port B pins (B8, B9, B13, B14). This prevents the use of the four timer synchronous outputs TS0-TS3 which are the alternate functions for these pins. The BANK memory is now addressed using A0:A14, B8, B9, B13, B14 and is accessed when A15 is a '0'. The BANK memory address space is now expanded to 512k bytes broken down into 16 individually selectable banks of 32k bytes each selected by these four bits of Port B.

A look at Table 1 and Figure 1 quickly tells you that only one bank in the BANK memory space can share the logical address range 0000:7FFF at any one time. Therefore, programs running in the BANK memory address space can only directly access data and programs in the MAIN memory address space or in it's own bank (selected by B8, B9, B13, B14). On chip resources, which include RAM, I/O, and registers are mapped into logical addresses 0000 to 01FF. These logical addresses are in the BANK memory address space, but, since these addresses are considered to be al-

ways on-chip by the HPC, it never looks at the external address/data bus and will not read external memory in this range. Therefore, the first 256 bytes in each bank of memory in the BANK memory space will not be accessible by the HPC, but this address range (on chip resources) is directly accessible by any bank of memory in the BANK memory address space. This is why Figure 1 shows a total available memory of 536.5k.

The interrupt vectors are mapped into logical addresses FFF0 to FFFF which are in the MAIN memory address space. Interrupts are handled properly if they occur while executing a program out of one of the banks of memory in BANK memory space, since the interrupt vector locations have A15 set to '1' which will allow access to the MAIN memory space. However, these interrupt vectors must either point to a routine in the MAIN memory address space which performs the interrupt service or point to code that selects the appropriate bank of memory in the BANK memory space and go there if the interrupt service routine is located there.

The stack must be located so that it can be directly accessible from anywhere in memory. It can be placed in the MAIN memory space or in the on-chip RAM. Programs and data storage that must be shared and directly accessed by all memory banks in the BANK memory space should also reside in the MAIN memory space.

HPC OPERATING MODES

The HPC must be configured to run in one of it's Expanded modes of operation by setting the EA bit in the PSW to be able to address the BANK memory range of 0000 to 7FFF. This memory expansion addressing scheme will work if the HPC is configured in either the Normal Expanded mode (EXM pin tied low) or ROMless Expanded mode (EXM pin tied high). The Normal mode differs from the ROMless mode only by the fact that the HPC will access the on-chip ROM for addresses in the range of E000 to FFFF (in the case of the HPC16083) and will access the external MAIN memory for addresses in the range of 8000 to DFFF.

The external data bus size is determined once, at reset, by sampling the state of \overline{HBE} (B12). If \overline{HBE} is high when sampled, the HPC enters 8-bit mode. In 8-bit mode, only pins A0-A7 are used to transfer data and pins A8-A15 continue to hold the most-significant eight bits of the address. So, only the lower eight bits of the address need to be latched externally (Figure 2). If HBE is low when sampled, the HPC enters 16-bit mode. In 16-bit mode, all 16 pins of Port A are used to transfer data as well as addresses. Two octal latches are then required externally to hold each address as it is issued by the HPC. The signal ALE from the HPC clocks the latches (Figure 3).

Keep in mind that if the external memory is configured as 8-bit memory, then the program stack must be in internal on-chip RAM because it has to be accessible as 16-bit words. If the external memory is configured as 16-bit memory then the stack can be in external RAM but must be in the MAIN memory address space to be directly accessible by all banks.

PROGRAMMING CONVENTIONS

A convention must be followed for maintaining linkages between the programs and data running in the MAIN memory space and the programs and data running in the BANK memory space. For the following discussion, the MAIN memory space will be referred to as just another bank of memory.

MAIN bank reserved portion

A portion of the MAIN memory bank should be reserved for Jump instructions to subroutines in the MAIN memory bank that need to be called by programs running in any selected bank in the BANK memory space. These Jump instructions serve as entry points for programs and subroutines. Typically, common functions that are required by programs running in several banks would be put in the MAIN memory bank. These could include: interrupt service routines, I/O drivers, and data handling and conversion routines. This portion also contains address pointers to tables of data in the MAIN memory bank that also are required by programs running in any selected bank in the BANK memory space. See Listing 1 for an example.

BANK memory reserved portion

A portion of each bank in the BANK memory space should be reserved for Jump instructions to subroutines in that bank that need to be called by programs running in the MAIN memory bank. These Jump instructions serve as entry points for programs and subroutines. For example, each bank in the BANK memory space could contain routines that perform unique but related functions. One bank could be reserved for math routines; another bank could perform message handling; and yet another could contain diagnostic routines. All of these functions could be scheduled and executed from some sort of Supervisor running in the MAIN memory bank performing the linkages to all these routines thru the entry points. This reserved portion of each bank also contains address pointers to tables of data in that bank that also are required by programs running in the MAIN memory bank. In the case of a bank running message handling routines, address pointers could be inserted to point to buffers that programs running in MAIN memory need to access. See Listing 2 for an example.

Linkage areas

These reserved portions of each memory bank (MAIN space or BANK space) must be fixed and known to each other memory bank that requires access to programs and data in that bank. Therefore, one other requirement in each bank is a set of labels that are assigned the values of the pointer locations to subroutines and tables in the bank of interest (see Listings 3 and 4).

One last requirement in the MAIN memory bank, if it is to perform bank to bank moves and for general housekeeping, is to reserve two byte locations to be used to keep track of the bank currently selected (high byte value on Port B) being used in the transfer of data (see Listing 5).

From the MAIN memory bank, the user can access all memory in the system. He can call subroutines in any bank in the BANK memory space and read/write data to the entire memory. From any bank in the BANK memory space, the user can call subroutines in the MAIN memory bank and read/write data to the MAIN memory bank in addition to his own local bank.

The basic procedure used to call a program in the BANK memory space from the MAIN memory bank is merely to set the proper value on the Port B select lines and execute a Jump to SubRoutine through a pointer in the selected bank:

Interrupts

Regardless of where the interrupt service routine actually resides, an image of the bank selected must be retained by the service routine to allow it to return to the appropriate bank when complete. If the interrupt service routine is in the MAIN memory bank, the linkage is handled in the normal fashion where the interrupt vector points to the service routine. The interrupt service can reside in the BANK memory space and takes a little extra overhead for the linkage.

To call a program in the MAIN memory bank from the BANK memory space, merely execute a Jump to SubRoutine through a pointer in the MAIN memory bank:

```
JSRL CMPBLNK ;see Listing 1 and 4
```

EXAMPLE SOFTWARE

Now that a convention has been established for communicating between the MAIN memory space and the BANK memory space, let's take a look at some sample code that can be used to move data between these memory spaces. In order to make the selection of bank memory efficient, it is important to keep in mind that the four bits of the high byte of Port B that are used to select a bank of memory in the BANK memory space can be written to directly since the other 4 bits of this byte of Port B are used for memory control outputs (the external control bus) and are not affected by a write to the high byte of Port B.

Bank to Bank data transfer by MAIN

Listing 6 shows the setup required to initialize the linkage area in order to perform a transfer of data from one bank to another bank in the BANK memory space by a program running in the MAIN memory space. This involves setting up the RAM locations that are used to 'select' the source bank and the destination bank, select the source bank to determine the starting address of the area to move, select the destination bank to determine the starting address of the area to move data into, then finally calling the subroutine in MAIN memory that performs the move. After the setup portion, the subroutine that performs the transfer is presented. This code assumes that the external memory is configured in 16-bit mode.

Bank to MAIN data transfer by Bank

Listing 7 presents a similar example for moving blocks of data from a bank in BANK memory to MAIN memory by a program running in that bank. This code also assumes that the external memory is configured in 16-bit mode.

External 8-bit mode

If the external memory is configured in 8-bit mode, the setup portion changes because the initialization of the RAM address pointers SSTART, DSTART and DEND requires building word address pointers from word pointers in the external reserved areas of each bank. In 8-bit mode, this requires two 8-bit transfers compared to one 16-bit transfer in 16-bit mode (see Listing 8). Once these address pointers have been built, however, the subroutine that actually performs the move does not have to change because 1) word transfers are allowed between On-chip RAM and registers regardless of the mode and 2) the subroutine performs byte moves. To improve speed in the 16-bit mode, this subroutine can be modified to perform 16-bit moves. However, keep in mind that this will impose the restriction on the address pointers in the linkage areas of requiring that addresses be on word boundaries. Listing 9 presents a similar example for moving blocks of data from a bank in BANK memory to MAIN memory by a program running in that bank.

PROGRAM DEVELOPMENT

The MOLE monitor software can support the development of HPC programs in multiple banks of memory. It provides the means of qualifying a trigger condition, as set in Trace or Breakpoint functions, with the memory bank number. The BANK command will allow a trigger only when executing in the memory bank of interest. The MOLE supports a total of 16 memory banks which are normally selected by 4 bits of Port B as described earlier. See the HPC Personality Board User's Manual for further detail on this command.

CONCLUSION

What has been presented is a method to expand the memory space of the HPC to 544k. Although this method utilized four bits of Port B to accomplish the extra addressing, theoretically, the remaining 8 bits could have been used if not required for other purposes. This could mean a maximum addressability for the HPC of greater than 128 Megabytes. However, the MOLE will only support the fixed definition of four extra address lines. Clever utilization of existing resources can enable you to get the most out of hardware and software limited only by one's imagination.

TABLE I. Logical Addresses vs Physical Memory Locations

Logical Address	Bank #	Hi Byte Port B	Physical Address
0000:7FFF	0	00	00000:07FFF
0000:7FFF	1	01	08000:0FFFF
0000:7FFF	2	02	10000:17FFF
0000:7FFF	3	03	18000:1FFFF
0000:7FFF	4	20	20000:27FFF
0000:7FFF	5	21	28000:2FFFF
0000:7FFF	6	22	30000:37FFF
0000:7FFF	7	23	38000:3FFFF
0000:7FFF	8	40	40000:47FFF
0000:7FFF	9	41	48000:4FFFF
0000:7FFF	A	42	50000:57FFF
0000:7FFF	B	43	58000:5FFFF
0000:7FFF	C	60	60000:67FFF
0000:7FFF	D	61	68000:6FFFF
0000:7FFF	E	62	70000:77FFF
0000:7FFF	F	63	78000:7FFFF
8000:FFFF	—	—	08000:0FFFF (MAIN)

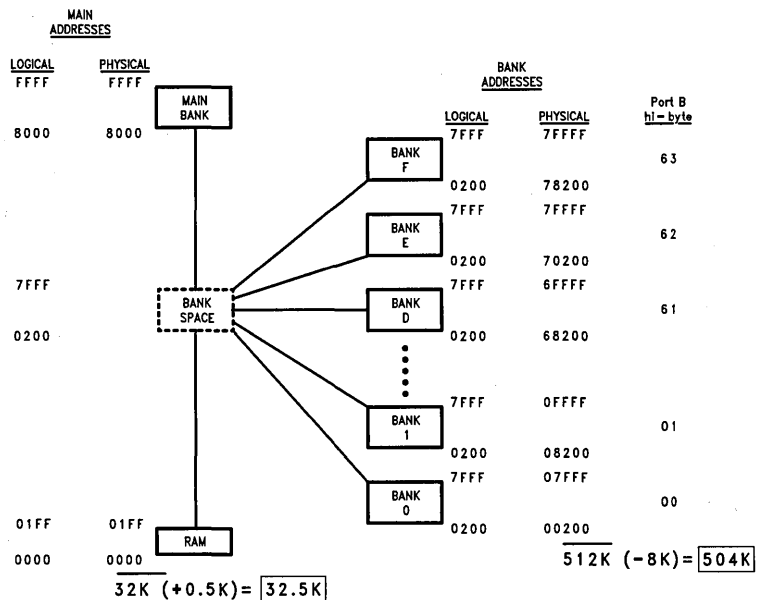


FIGURE 1. How BANK Memory is Mapped into the HPC Address Space

TL/DD/9342-1

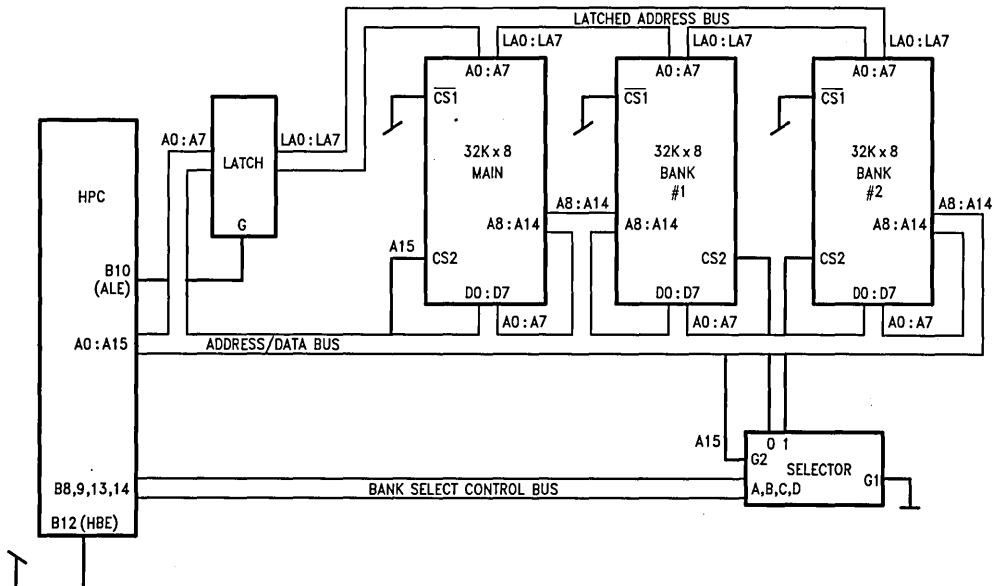


FIGURE 2. HPC in 8-Bit Mode

TL/DD/9342-2

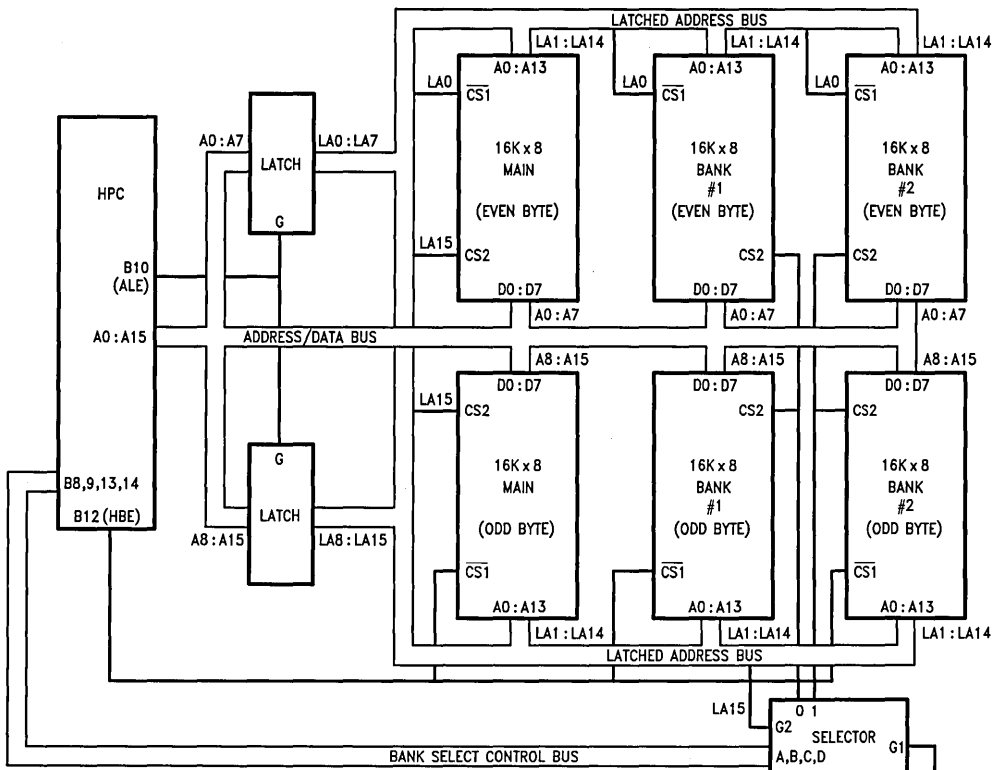


FIGURE 3. HPC in 16-Bit Mode

TL/DD/9342-3

```

      .=08000      ;set PC counter to 8000
;This code resides in the MAIN memory bank
;
;   The following address pointers are inserted to allow
;   programs running in BANK memory to find these
;   locations. They represent the starting and ending
;   location for code in MAIN memory.
;
      .WORD INIT   ;addr pointer to first location in bank
      .WORD PROGEND ;addr pointer to last location in bank
;
;   The following Jump instructions are inserted to allow
;   programs running in BANK memory to call these
;   subroutines. They represent subroutines that compare
;   blocks of memory in MAIN memory space with blocks of
;   memory in BANK memory space or compare blocks of memory
;   in BANK memory for zeros.
;
      JMPL CMPM      ;entry for compare blocks (MAIN-BANK)
      JMPL CMPBFB    ;entry for compare BANK cleared

```

LISTING 1. MAIN Bank Reserved Portion

```

      .=0200      ;set PC counter to 200
;This code resides in any bank in BANK memory
;
;   The following address pointers are inserted to allow
;   programs running in MAIN memory to find these
;   locations. They represent the ending location for code
;   in this bank of BANK memory.
;
      .WORD PROGEND ;addr pointer to last loc in this bank
;
;   The following Jump instructions are inserted to allow
;   programs running in MAIN memory to call these
;   subroutines. They represent subroutines that compare
;   blocks of memory in MAIN memory space with blocks of
;   memory in this bank, diagnostic routines, and interrupt service routine.
;
      JMPL CMPMB    ;entry for comp blocks (MAIN-this bank)
      JMPL BTEST    ;entry for this bank's diag routines
      JMPL BINTS    ;entry for this bank's interrupt service routine

```

LISTING 2. Typical Bank Reserved Portion

```

;This code resides in the MAIN memory bank
;
;   linkages to Bank 0
;
B0START = 0200           ;addr of pointer to first avail loc
;
CMPMBO = 0202           ;addr of JMPL to routine that compares
;                       move results
BOTEST = 0205           ;addr of JMPL to test routines
;
;   linkages to Bank 1
;
B1START = 0200           ;addr of pointer to first avail loc
;
CMPMB1 = 0202           ;addr of JMPL to routine that compares
;                       move results
B1TEST = 0205           ;addr of JMPL to test routines
;
;   linkages to Bank 2
;
B2START = 0200           ;addr of pointer to first avail loc
;
CMPMB2 = 0202           ;addr of JMPL to routine that compares
;                       move results
;                       = 0205           ;addr of JMPL to test routines
;
B2INTS = 0208           ;addr of JMPL to interrupt service routine
                        LISTING 3. MAIN Memory Bank Linkage Area

```

```

;This code resides in any bank in BANK memory
;
;   linkages to MAIN memory
;
MSTART = 08000          ;addr of pointer to first avail loc
MEND = 08002            ;addr of pointer to last avail loc
;
CMPM = 08004            ;addr of JMPL to routine that compares
;                       move results
CMPBLNK = 08007         ;addr of JMPL to routine that compares
;                       if a block in selected BANK is zero
                        LISTING 4. Typical Bank Linkage Area

```



```

;This code resides in the MAIN memory bank
;
; The following locations are used for bank to bank moves
; and compares
BANKS = 01C0      ;source bank byte value
BANKD = 01C1      ;destination bank byte value
;
BANK0 = 0         ;Port B high byte value to select bank 0
BANK1 = 1         ; 1
BANK2 = 2         ; 2
BANK3 = 3         ; 3
BANK4 = 020       ; 4
BANK5 = 021       ; 5
BANK6 = 022       ; 6
BANK7 = 023       ; 7
BANK8 = 040       ; 8
BANK9 = 041       ; 9
BANKA = 042       ; 10
BANKB = 043       ; 11
BANKC = 060       ; 12
BANKD = 061       ; 13
BANKE = 062       ; 14
BANKF = 063       ; 15
;
; Main Memory Bank is logical and physical address range
; 8000:FFFF. Switched Memory Banks are logical addresses
; in the range 0000:7FFF combined with the
; Port B(14,13,9,8) bits to create physical addresses in
; the range 00000:7FFFF
;

```

LISTING 5. BANK Memory Management

```

LD M(OE3),BANK1 ;set bank select lines to select bank 1
JSRL B1TEST     ;see Listing 2 and 3
.
.
.
INT35:
LD BANKS,M(OE3) ;save bank interrupted from
LD M(OE3),BANK2 ;set bank select lines to select bank 2
JSRL B2INTS     ;see listing 2 and 3
.
.
.
LD M(OE3),BANKS ;restore bank interrupted from
RETI
.
.
.
.IPT 2,INT35    ;set interrupt vector

```

```

;This code resides in the MAIN memory bank
;
LD M(BANKS),BANK0      ;prepare to move data from Bank 0
LD M(BANKD),BANK1      ;to Bank 1
LD M(OE3),BANK0        ;select Bank 0
LD W(SSTART),W(BOSTART);set starting address in source bank
LD M(OE3),BANK1        ;select Bank 1
LD W(DSTART),W(B1START);set starting address in destination bank
LD W(DEND),W(B1START)  ;set ending address in destination bank
ADD W(DEND),1023        ;to 1K greater than starting address
JSRL MOVBB              ;do it
.
.
.
.
.
;
; This subroutine moves data from bank memory to bank memory
; where the source bank is defined by the contents of the byte
; at RAM location BANKS and the destination bank is defined by
; the contents of the byte at RAM location BANKD. In addition,
; the following locations must be set up before calling:
;
; SSTART → RAM location containing source bank start address
; DSTART → RAM location containing destination bank start address
; DEND → RAM location containing destination bank end address
;
MOVBB:
LD B,W(DSTART)         ;B ← starting address (destination)
LD K,W(DEND)           ;K ← ending address (destination)
LD X,W(SSTART)         ;X ← starting address (source)
LOOPBB:
LD M(OE3),M(BANKS)     ;select source BANK
LD A,M(X+)             ;byte at source into A
                       ;increment source pointer
LD M(OE3),M(BANKD)     ;select destination BANK
XS A,M(B+)             ;A into byte at destination, bump pntr
JP LOOPBB              ;back for more if B less than K
RET

```

LISTING 6. Move Data by MAIN from BANK to BANK (16-Bit Mode)

```

;This code resides in any bank in BANK memory
;
LD W(SSTART),TABLE1 ;starting address of table in this memory
LD W(DSTART),W(MSTART) ;starting address in main memory
LD W(DEEND),TABLE1+1023 ;ending address in main memory
JSRL MOVE ;do it
.
.
.
.
.
;
; This subroutine moves data from this bank to main memory
;
; SSTART → RAM location containing source memory start address
; DSTART → RAM location containing destination memory start addr
; DEEND → RAM location containing destination memory end address
;
MOVE:
LD B,W(DSTART) ;B ← starting address (destination)
LD K,W(DEEND) ;K ← ending address (destination)
LD X,W(SSTART) ;X ← starting address (source)
LOOPBM:
LD A,M(X+) ;byte at source into A
;increment source pointer
XS A,M(B+) ;A into byte at destination, bump pntr
JP LOOPBM ;back for more if B less than K
RET

```

LISTING 7. Move Data by BANK from BANK to MAIN (16-Bit Mode)

```

;This code resides in the MAIN memory bank
;
LD M(BANKS),BANK0      ;prepare to move data from Bank 0
LD M(BANKD),BANK1      ;to Bank 1
LD M(OE3),BANK0        ;select Bank 0
LD M(SSTART),M(BOSTART) ;set starting address in source bank
LD M(SSTART+1),M(BOSTART+1)
LD M(OE3),BANK1        ;select Bank 1
LD M(DSTART),M(B1START) ;set starting address in destination bank
LD M(DSTART+1),M(B1START+1)
LD M(DEND),M(B1START)  ;set ending address in destination bank
LD M(DEND+1),M(B1START+1)
ADD M(DEND),L(1023)    ;to 1K greater than starting address
ADC M(DEND+1),H(1023)
JSRL MOVBB             ;do it
    .
    .
    .
    .
    .
;
; This subroutine moves data from bank memory to bank memory
; where the source bank is defined by the contents of the byte
; at RAM location BANKS and the destination bank is defined by
; the contents of the byte at RAM location BANKD. In addition,
; the following locations must be set up before calling:
;
; SSTART → RAM location containing source bank start address
; DSTART → RAM location containing destination bank start address
; DEND → RAM location containing destination bank end address
;
MOVBB:
LD B,W(DSTART)         ;B ← starting address (destination)
LD K,W(DEND)           ;K ← ending address (destination)
LD X,W(SSTART)         ;X ← starting address (source)
LOOPBB:
LD M(OE3),M(BANKS)     ;select source BANK
LD A,M(X+)              ;byte at source into A
                        ;increment source pointer
LD M(OE3),M(BANKD)     ;select destination BANK
XS A,M(B+)              ;A into byte at destination, bump ptr
JP LOOPBB               ;back for more if B less than K
RET

```

LISTING 8. Move Data by MAIN from BANK to BANK (8-Bit Mode)

```

;This code resides in any bank in BANK memory
;
LD M(SSTART),L(TABLE1) ;starting address of table in this memory
LD M(SSTART+1),H(TABLE1)
LD M(DSTART),M(MSTART) ;starting address in main memory
LD M(DSTART+1),M(MSTART+1)
LD M(DEND),M(MSTART) ;set ending address in main memory
LD M(DEND+1),M(MSTART+1)
ADD M(DEND),L(1023) ;to 1K greater than starting address
ADC M(DEND+1),H(1023)
JSRL MOVE ;do it
.
.
.
.
.
;
; This subroutine moves data from this bank to main memory
;
; SSTART → RAM location containing source memory start address
; DSTART → RAM location containing destination memory start addr
; DDEND → RAM location containing destination memory end address .
;
MOVE:
LD B,W(DSTART) ;B ← starting address (destination)
LD K,W(DEND) ;K ← ending address (destination)
LD X,W(SSTART) ;X ← starting address (source)
LOOPBM:
LD A,M(X+) ;byte at source into A
;increment source pointer
XS A,M(B+) ;A into byte at destination, bump ptr
JP LOOPBM ;back for more if B less than K
RET

```

LISTING 9. Move Data by BANK from BANK to MAIN (8-Bit Mode)

The code listed in the App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

Assembly Language Programming for the HPC™

National Semiconductor
Application Note 510
Steve McRobert



AN-510

HOW TO WRITE SHORT, EFFICIENT, BUT UNDERSTANDABLE ASSEMBLER PROGRAMS

INTRODUCTION

One of the design objectives of the HPC family was that it should be very easy to use. With this in mind the instruction set has been designed so that it obeys a very simple set of rules. Once these rules have been learned, the programmer can write code with very little reference to instruction manuals.

The HPC is fully memory mapped. Every piece of hardware attached to an HPC core appears as a byte or a word in a linear 64K byte address space. Any data movement or arithmetic instruction can operate on any memory location and everything in the HPC has a memory location, including the accumulator. All of the I/O ports, the peripheral control registers, RAM and ROM are treated in exactly the same fashion as far as the assembly language programmer is concerned.

The HPC assembly language syntax can be explained by describing the instruction codes and the addressing modes. The instruction code tells the processor what operation it is performing, such as an add, a subtract, a multiply, a divide or a data movement instruction. The addressing mode is the way that the programmer specifies the value or values to be operated on to the microprocessor itself.

ADDRESSING MODES

Operations can be performed on any memory location. One can, for example, increment or decrement any byte or word of any memory location in the HPC. Increment and decrement are examples of single address instructions. These are instructions which have only one operand. Other examples are the bit set, bit test and bit clear instructions. These five instructions are good examples of the basic thinking behind the HPC instruction set. All of these instructions use the same four addressing modes.

Direct

The simplest addressing mode to understand is that known as direct. In this mode the address of the variable to be operated on is included as part of the sequence of bytes that comprises the entire instruction. For example, in order to perform a decrement on memory location 0F0 this value is included in the string of bytes that forms the instruction.

Examples:

```
DECSZ 0F0.B
INC    0F0.W
```

The increment instruction, like most other instructions with HPC, can operate on either a byte or a word. A byte access is specified by putting a B after the address of the variable, a word access by writing W.

Register Indirect

This addressing mode usually generates less bytes of code than any other. HPC has two 16-bit registers, B and X, which

can be used as general purpose memory locations but also have a specific function as pointers to memory. These instructions take up very little ROM space because the address of the variable to be operated on is contained in the pointer register and the pointer register to be used is specified as part of the instruction. An instruction such as increment, using register indirect, can thus be only 1 byte long as it does not need to be followed by a byte specifying the address of the variable.

Examples:

```
INC    [B].B ;byte increment, B pointer
INC    [X].W ;word increment, X pointer
```

Indirect

B and X provide two 16-bit pointers to memory. Programmers will often wish to have more than two pointers in use at any one time. HPC therefore provides indirect addressing mode. In this mode a 16-bit pointer to the location to be accessed is stored in the basepage of the HPC. The instruction, therefore, is followed by a single byte which specifies the address of this 16-bit pointer. The bottom 192 bytes of RAM are on chip with the HPC and are in the so-called base page. The base page is normally used for storing frequently accessed variables as only a single byte of address is required to access a base page variable. When using indirect addressing mode, the 16-bit pointer value must always be in the base page.

Examples:

```
DECSZ [0].W ;decrement a word
INC    [0FE].B ;increment a byte
```

The base page is in the region of 0 to 0FF bytes. This area also contains the most frequently used registers such as the accumulator. The programmer can thus use indirect addressing mode with registers such as the accumulator acting as the pointer. This is an example of the simplicity of the HPC instruction set. Any operation can be performed on any HPC register simply by invoking its address in the HPC 64 kbyte addressing space.

Indexed

The last of the four basic addressing modes is indexed mode. Indexed is very similar to indirect except that an 8- or 16-bit immediate value precedes the address of the 16-bit pointer and is added to it to generate the address of the variable to be accessed. This allows a table of values to be located anywhere in memory and the pointer register need only be incremented or decremented to move through the table of values.

Examples:

```
INC    OFF00 [4].W ;increment a word
DECSZ 02 [2].B ;decrement a byte
```

Bit Operations

The bit operations of the HPC allow any bit in the memory of the HPC to be accessed. The addressing modes for these three operations, SBIT, RBIT and IFBIT, always refer to the memory location as a byte. The individual bit of the byte to be tested, using the four addressing modes already described, is actually coded into the opcode itself. This could be described as an implied addressing mode but this definition is not normally used in HPC. The way this works can be seen from the opcode map in the programmers guide of the HPC, where it can be seen that there are in fact eight opcodes shown for each of the three different bit instructions.

Example:

```
SBIT 5, 2.B ;set bit 5 of byte
           ;at address 2.
```

Double Register Indirect

A rule of thumb when trying to decide which addressing mode one can use with which opcode in HPC is that you can use any combination of addressing mode and opcode that is sensible. An example of this is a special addressing mode which works only for the bit instructions. This addressing mode is known as double register indirect and uses a combination of the B and X registers to index into any bit of a 64k bit string, the lower boundary of which can be located anywhere in memory.

When using this addressing mode the B register points to the lowest byte of this 8k byte string, while the most significant 13 bits of the X register point at the individual byte in the string that is being accessed. The three least significant bits of the X register point at the bit of the byte that the instruction is pointing at. By using this addressing mode, words of any length can be scanned for whether individual bits are set or cleared. This addressing mode, while unusual, fits into the scheme of things as it clearly is only of any relevance to the individual bit instructions.

Examples:

```
SBIT X, [B].B ; Set bit
IFBIT X, [B] B ; test bit
```

Note that the bit instructions only operate on bytes, to allow operations on words would require twice as many opcodes for no gain.

Two Address Instructions

The five instructions described so far have only one operand. There are many more instructions in the HPC instruction set which have two operands, such as arithmetic instructions, the comparison instructions and data movement instructions. The HPC instruction set allows any of these instructions to use any of the four addressing modes already described. An instruction such as multiply, for example, when written in the HPC assembler syntax as shown below shows the opcode followed by the destination operand, which is then followed by the source operand. The result of the operation in all cases except the comparison instructions winds up in the destination operand. The comparison instructions, IFEQ and IFGT do not affect the values of any memory location but, like all other two operand instructions, can operate on any two words or bytes in the HPC addressing space.

Examples:

```
MUL A, [B].B
MUL 0.W,2.W
```

The destination operand in HPC may be either the accumulator or a byte or word of memory accessed using the direct addressing mode. If the destination operand is the accumulator, the source operand may be addressed using direct, register indirect, indirect or indexed addressing modes as well as the familiar immediate addressing mode. The programmer can thus load the accumulator with an 8- or 16-bit immediate value which follows the opcode, multiply the accumulator with that value, divide the accumulator by that value or compare the accumulator by that value. Using the accumulator as the destination operand gives maximum flexibility in the choice of addressing mode for the source operand and also tends to produce a shorter instruction in terms of its length in bytes as the opcode does not have to include the address of the destination operand.

Examples:

```
LD A, #37 ;load A With
           ;immediate value.

add OFE.W,# OF000 ;Add immediate to
                  ;memory.
```

Instruction Lengths

Tables are provided in the HPC users manual to allow the user to estimate the number of bytes an instruction will use and the time this instruction will take to execute. To use these tables the programmer must be aware of the name of the addressing mode he is using. This is perfectly clear for the single address instructions described at the beginning of this note but perhaps needs some explanation for two operand instructions.

For two operand instructions with the accumulator as the destination, the addressing mode is named after that used for the source operand. For example, load accumulator using a value pointed at by indirect addressing mode is referred to simply as indirect addressing mode.

Operations on Direct Memory

There are two addressing modes which allow operations to be performed directly on memory locations. If the destination operand is directly addressed memory, then the source operand may be directly addressed memory or an immediate value. These two are the only combinations of addressing modes that can be used where the destination operand is a memory location.

Examples:

```
DIV 010.W, OF000.W
direct-direct mode

DIV OF0.B,#10
immediate direct mode.
```

Special Symbols

Some special symbols have been allocated in the HPC cross assembler. These are A, B, K, X, PC and SP. The programmer can also define his own symbols using the equals directive of the assembler. The way that the symbols described above would be defined using the equals directive are shown below by way of example.

Example:

```
A = 0C8.W
B = 0CC.W
X = 0CE.W
K = 0CA.W
PC = 0C6.W
SP = 0C4.W
```

Note that these symbols cannot be redefined so the above set of definitions should never be included in a user program.

IMPLIED ADDRESSING MODES

Some of the HPC's opcodes have been shortened by using implied addressing mode. A few examples have already been shown. This section describes some more special cases. It could be said that accumulator as destination is an example of an implied addressing mode, where the address of the destination is coded into the instruction. There are some special purpose instructions which use implied addressing mode for instructions which are used very frequently. In most cases these instructions look exactly the same to the programmer as instructions using the addressing modes described earlier. For example there is a special opcode for load B with an immediate value. The programmer could do this using the immediate direct addressing mode but a special opcode has been provided to make this instruction shorter.

Load B and K is a special immediate load which loads both the B and K registers in one operation.

Carry Flag

The carry flag may be accessed using the standard bit test instructions because it can be read in the processor status word, but as carry must so often be set and tested, special instructions to do this have been included which do not require the address of the carry flag.

Multiply and Divide

Finally, the divide double and multiply instructions both have to manipulate 32-bit values. These therefore have to store an operand in two concatenated registers. The HPC instruction set cannot specify two registers with one address. Therefore these instructions default to using the X register as the high word of their 32-bit value.

The source and destination of a multiply instruction are specified as normal except that the 32-bit answer is stored in the destination operand with the 16 high bits of the answer stored in the X register. The divide double instruction basically performs the inverse of multiply, taking the 32-bit value formed by X concatenated with the destination value and dividing it by the source value. Divide double, like divide, yields a 16-bit result and a 16-bit remainder. For both divide double and divide the remainder is stored in the X register. In both cases the K register is used for intermediate value storage and is cleared as a result of this operation.

As the result of divide double can only be a 16-bit value, a full 32-bit divide is performed by following a 16-bit divide with a 32-bit divide as shown below. The example below shows how the divide instructions work together and also highlights the combinations of addressing modes that can and cannot be used with HPC.

```

LD      B, #11
DIV     HIGH.W, #10
LOOP:  DIVD    LOW.W, #10
LD A, X
ST A, [B].B
DECSZ B
JP LOOP

```

This example shows the conversion of a 32-bit binary value in words low and high into a 10-digit BCD number in the 10 bytes starting from 1. The conversion is performed one digit at a time and the B register is used to point at the byte's location where the digit is to be stored. The first instruction of the programme therefore is to initialize the B register. The divide instruction divides word high by 10 using immediate direct addressing mode and stores the answer back in word high. The remainder is stored in the X register. The divide double instruction then divides X concatenated with word low by 10. Because X contains a remainder, the result of this division will always be a 16-bit value and can thus be stored in word low. The remainder is stored in X and is in fact the modulus and is thus the BCD digit that we have derived on this pass through the numbers.

We now wish to store the remainder into one of our BCD digit locations using register indirect mode. We need to load the value into the accumulator from X. The X register is nothing special in this application, so load A with word X is in fact an example of direct addressing mode.

Now that our BCD value is in the accumulator, we can store this in the byte location using B register indirect addressing mode.

The next instruction is decrement skip on zero. This uses direct addressing mode to decrement the B register. This instruction is an example of many in HPC which perform more than one function. As well as decrementing the memory location specified, this instruction also compares it with zero after the decrement has been performed. If the result is zero, the instruction following the decrement skip on zero instruction is skipped. That is to say it is ignored and control passes to the instruction following it. In this example the final instruction of the routine is a single byte jump back to the divide instruction. The overall loop is executed ten times in order to perform the conversion. On the final pass through the loop, B becomes zero and execution of this algorithm is terminated.

Auto Increment/Decrement Instructions

This multi-function instruction capability is best illustrated by the four special addressing modes register increment or decrement with or without conditional skip, which work only with the data movement instructions load and exchange. The load instruction in general uses any of the five two-address modes or the two combination modes to transfer data from one location to another.

The exchange instruction is similar except that the destination must always be the accumulator. Exchange not only takes the source and puts the value into the destination but also takes the value from destination and puts it into source. Clearly there is no immediate addressing mode for exchange as a destination cannot be stored into an immediate value.

When load and exchange are used with the X register as a pointer and register indirect mode, a suffix + or - can be added after the X. In this case, once the data movement operation has been performed, the X register is incremented or decremented by one or two according to whether

there has been a byte or a word access respectively. A further refinement on this is provided by the load and exchange with conditional skip instructions, LDS and XS respectively. These only work with the B register as the pointer and perform two more operations rather similar to the decrement skip on zero instruction. Once the increment or decrement has been performed, the B register is compared with the K register, otherwise known as the limit register. If an increment has been performed and B is greater than K, the instruction following the movement instruction will be skipped. If a decrement is performed, the instruction is skipped if B is less than K.

An example of how these specialized instructions are used is given by the block move routine shown below;

```
LD X, #START
LD BK, #BEGIN, #END
LOOP: LD A, [X+].W
XS A, [B+].W
JP LOOP
```

This routine moves a block of data from one location to another. The X register is initialized first and is used as a pointer to the first value to be moved in the source block. The B and K registers point to the first and last values respectively in the destination block. The loop itself consists of only three bytes. The first instruction loads the accumulator with the word pointed to by the X register and increments X by two. A second instruction exchanges the accumulator with the word pointed to by the B register, increments the B register by two and compares it with K. If B is greater than K, the jump instruction is skipped and this loop is terminated.

The example shows how HPC code can perform a great deal with very few instructions and use up very few bytes of code while doing so.

These auto increment/decrement instructions are the only examples where an addressing mode cannot be used for any instruction where it might make sense. It is however fairly easy to remember which addressing modes these can be used with. Auto increment/decrement can be used with the load and exchange instructions for the X register. Auto increment or decrement with conditional skip can be used with load and exchange instructions using the B register as a pointer. No other combinations are allowed.

We have not provided specific string move or search instructions but the auto increment/decrement operations provide building blocks allowing the programmer to assemble his own stock. In the block move instruction shown above, the value being moved is in the accumulator in between the load and exchange instructions. The programmer can then compare this value with anything he wishes, fill BCD to ASCII, pack BCD, unpack BCD or perform any operation he likes on a string of data.

HPC ASSEMBLY CODE

The addressing modes usable for each opcode are described in a shorthand form.

Example :

```
ADD MA < MA + MemI
```

In the above syntax MA means directly addressed memory or the accumulator and MemI means memory addressed using any of the four basic single-address addressing modes or an immediate value. This would be better written as shown below:

```
A < A + MemI
or M < M + M
or M < M + I
```

Expanding the syntax highlights that the flexible addressing modes such as register indirect may only be used if the destination is the accumulator. It also shows that if the destination is direct memory the source may only be an immediate value or another direct memory location.

When writing assembly code the programmer writes the same mnemonic whether a memory location is a piece of RAM or ROM or an I/O port or the accumulator. In general any source or destination variable may be a byte or a word and combinations are allowed. Care must be taken when storing word into a byte location that the programmer really wishes to truncate that value to byte and throw away the upper 8 bits of the value. When loading a byte into a word location the upper 8 bits of the word location will be filled with zeros. If memory external to the HPC is used, this may be 8 or 16 bits wide. The programmer must be aware of this when writing his assembly language as HPC cannot cope with the programmer requesting a 16-bit access to 8-bit wide external memory. The HPC will not convert this to two sequential 8-bit accesses.

The only exception to this rule is that a pointer word in indirect or indexed addressing modes must always be in the base page. This is because only one byte has been allowed in the overall length of the instruction for the address of the pointer.

For all other addressing modes there is no difference in the assembly language the programmer writes between accessing a variable that is in the base page and a variable that is above address OFF.

The programmer should be aware however that variables in the base page consume less bytes per access and the instruction will execute more quickly than non-base page variables. When studying the data sheet to see how long an instruction is, the programmer will see that the table result is different according to whether variables are base page or not. The programmer should therefore allocate base page to variables which are used most often.

EXECUTION SPEED

There are 64 bytes of RAM above the base page. These, like the base page RAM, require zero wait states to access even when the processor is running at full speed. They do however require 2 bytes of code for their addresses. These

64 bytes may best be made use of by using them as the stack area as the 16-bit stack pointer contains the full address and therefore there is no penalty in instruction length in putting the stack in this non-base page on-chip RAM.

Note that there is no difference in execution time between byte and word accesses, that is to say accesses to byte or word variables. When studying the data sheet, differences in program length and therefore in execution time will be observed according to whether the address of a directly addressed variable is a byte or a word. It is important to understand the difference between the width of the variable and the width of the address that is used to access that variable.

The cycles per instruction table is not always clear about the number of wait states applied to different variables. The HPC includes a wait state register which sets the number of wait states to be used when accessing external memory, the internal ROM, or internal registers associated with ports A and B. Wait states may be applied to these on-chip registers to allow compatibility with development tools such as the MOLE™ and HPC Designer Kit board, as when these tools are run on high clock speeds wait states must be applied for accesses to the port recreation logic. The HPC needs wait states for accessing slow external memory and when running at high clock rates.

These wait states may be applied in order that the MOLE can provide a perfect emulation of a single-chip HPC. In the MOLE the HPC is running with external memory and thus the A port and some of the B port are used for address/data and control lines respectively. The A port and part of the B port must therefore be recreated external to the HPC. In the case of the MOLE this is done using a large array of PAL®s. Because they are external to the HPC, one wait state must be applied when accessing these externally recreated ports at high clock speeds. If wait states could not be applied to

these ports in a masked ROM HPC, the MOLE would not be able to provide full speed emulation. This is just one example of how the design of the HPC has been influenced by the need to emulate it 100% exactly at full speed. Apart from this no wait states are applied to any access to address locations below 200 HEX, regardless of the addressing mode used.

The HPC data sheet does not make it clear how many wait states are applied when register indirect addressing mode is used. It implies that wait states are always applied when register indirect or similar addressing modes are used, but this is not the case.

The best way to time a piece of code is to write the code and then run it through the cross assembler to generate a source plus object listing. The number of bytes generated by each instruction can then be easily read and only the cycles and accesses table need be looked up in order to calculate how long each instruction takes to execute.

Note that accesses to internal ROM are subject to at least one wait state for exactly the same reason as accesses to the A or B ports.

SUMMARY

The HPC is fully memory mapped. The I/O Ports, Peripheral Control Registers, RAM and ROM are treated exactly the same. This makes the HPC easy to program. The HPC instruction set has relatively few opcodes but allows any of these opcodes to be used with any addressing mode so as to provide an Instruction Set with great power and flexibility.

Once the contents of this note have been understood, HPC code can be written without referring to any document more lengthy than the HPC Instruction Set description in the data sheet.

A Software Driver for the HPC Universal Peripheral Interface Port

National Semiconductor
Application Note 550
Brian Marley



ABSTRACT

This application note covers the use of the National Semiconductor HPC46083 High-Performance microController as an intelligent Peripheral Interface and Interrupt controller for another "Host" CPU, using its 8-bit or 16-bit parallel UPI (Universal Peripheral Interface) Port. Included in the discussion is the source text of an HPC driver program, which can be tailored as an "executive" for a wide variety of HPC tasks. A simple application is built from this software, which interfaces a National NS32CG16 CPU to a typical front panel (LED indicators, LCD alphanumeric display, pushbuttons and beeper).

1.0 INTRODUCTION

The National Semiconductor HPC family of microcontrollers includes as a feature the ability to be slaved to another "Host" processor over that processor's memory bus. This feature, called the Universal Peripheral Interface (or UPI) Port, allows:

1. Transfer of either 8-bit or 16-bit data in a single bus transaction,

2. Polling to determine the status of the port from either side (Ready for Write/Ready for Read), and
3. Interruption of the host by the HPC with full vectoring.

The HPC, then, can serve as a front-end controller for the host, freeing it from control and/or communication tasks that might burden its capacity for interrupt service, and providing vectored interrupting for higher-level (and therefore less frequent) communication.

2.0 THE UPI PORT

2.1 Internal Structure

Figure 1 shows the internal structure of the UPI Port. It connects via three registers to the HPC's on-chip data bus, and via a set of pins (Port A) to the host's bus. The control interface between the HPC and the host consists of two low-active strobe signals ($\overline{\text{URD}}$ and $\overline{\text{UWR}}$) and an address signal (UA0) output by the host, and two handshake signals ($\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$) output from the HPC.

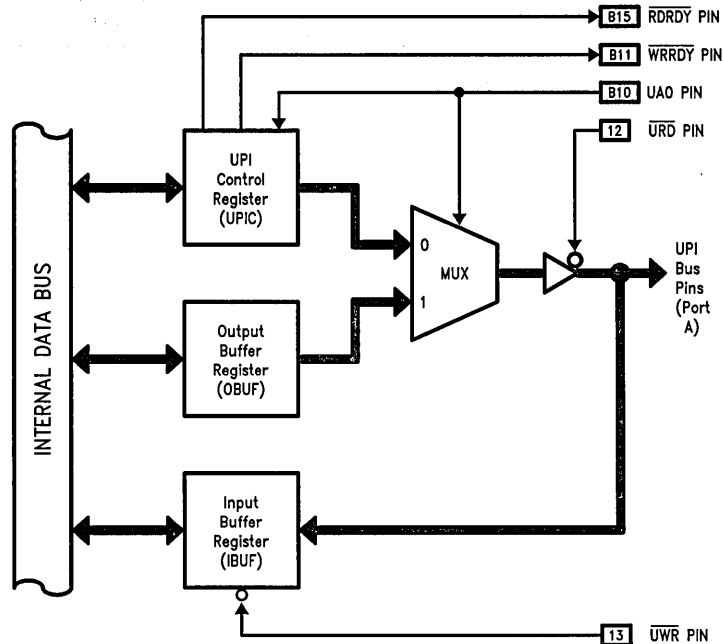


FIGURE 1. UPI Internal Structure

TL/DD/9976-1

The UPI Port may be configured either as a 16-bit bus (using all of Port A: pins A0–A15) or as an 8-bit bus (pins A0–A7), allowing pins A8–A15 to be used as general-purpose bit-programmable I/O pins. This selection is made by HPC firmware.

2.2 Basic Operations

Three types of operation may be performed over the UPI Port:

1. Transfer of a byte or word of data from the host to the HPC's IBUF register. This is called a "UPI Write" operation.
2. Transfer of a byte or word of data from the HPC's OBUF register to the host. This is called a "UPI Read" operation.
3. Polling by the host to determine whether the HPC is ready for the next UPI Write or UPI Read operation. This involves the host reading the UPIC (UPI Control) register, which contains the states of the \overline{WRRDY} and \overline{RDRDY} pins as two of its bits.

As shown in *Figure 2*, whenever the host writes to the HPC (by pulsing the \overline{UWR} signal low) data is latched into the HPC's IBUF register. At this time also, the value on the UA0 pin is latched into the UPIC (UPI Control) register, allowing

HPC firmware to route the data just written. (For example, this bit can be used by the HPC firmware to distinguish between commands and data written to it.) The rising edge of \overline{UWR} is detected by an edge-trigger circuit on-chip, which may be used to trigger an interrupt or for polling, to alert the HPC firmware to the presence of new data. The \overline{WRRDY} handshake signal, normally low, goes high until the HPC firmware has sampled the data written to it (by reading internally from the IBUF register).

Figure 3 shows the sequence of events in reading data from the HPC. The transfer starts when the HPC writes a value to the internal OBUF register. The \overline{RDRDY} handshake signal, normally high, goes low to indicate that data is present for the host. (This pin can be used to interrupt the host as well.) By pulsing the \overline{URD} pin low while holding the UA0 pin to a "1", the host reads the contents of the OBUF register, and the \overline{RDRDY} pin goes back high.

The polling operation (*Figure 4*) allows the host to monitor the \overline{RDRDY} and \overline{WRRDY} conditions as data bits, by pulsing the \overline{URD} pin low with a "0" held on the UA0 pin. This effectively reads from the UPIC register; the \overline{WRRDY} condition appears on bit 0 (the least-significant bit), and the \overline{RDRDY} condition appears on bit 1 (the next most significant bit). Polling in this manner does not affect the state of the \overline{RDRDY} bit.

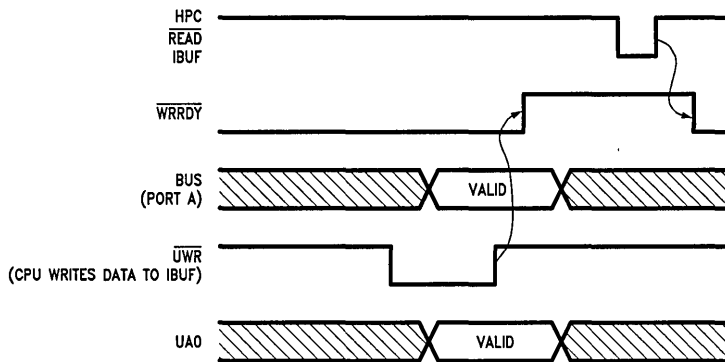


FIGURE 2. UPI Write Operation

TL/DD/9976-2

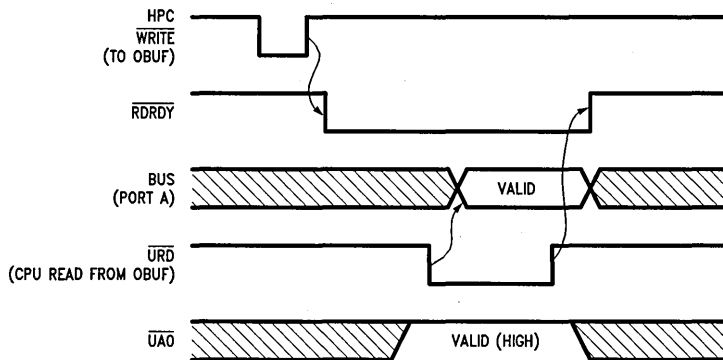


FIGURE 3. UPI Read Data Operation

TL/DD/9976-3

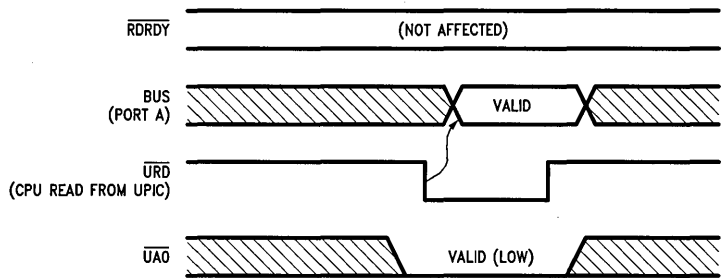


FIGURE 4. UPI Poll Operation

TL/DD/9976-4

2.3 Typical Hardware Configurations

Typical connections between the host and the HPC are shown in *Figures 5* through *7*.

2.3.1 Polled Synchronization

In the simplest case (*Figure 5*), the \overline{WRRDY} and \overline{RDRDY} signals are not used, and the host synchronizes itself with the HPC strictly by polling the UPIC register for the Read Ready and Write Ready conditions. The only additional logic always required is a pair of OR gates to activate \overline{URD} and \overline{UWR} only when the HPC is selected by the host's address decoder. Depending on the host, it may also be necessary to add WAIT states, as is often required in peripheral interfaces to match the bus timing characteristics of the two ends.

Sophisticated synchronization schemes are not available using this simple an interface, but it does save the HPC \overline{RDRDY} and \overline{WRRDY} pins for any other general-purpose I/O functions.

2.3.2 Interrupt-Driven Synchronization

Assuming that the host has interrupt control capability, the circuit above can be enhanced to implement an interrupt-driven synchronization scheme, as shown in *Figure 6*. A falling edge on either \overline{RDRDY} or \overline{WRRDY} will trigger an interrupt to the host, informing it when the HPC becomes ready for either direction of data transfer. No additional logic is required (except for possible buffering or inversion), but only dedication of the \overline{WRRDY} and/or \overline{RDRDY} pins for the interrupt function. It is not necessary for both \overline{RDRDY} and \overline{WRRDY} conditions to trigger interrupts; one can be polled and the other interrupt-driven, as dictated by the require-

ments of the system and the structure of the host and HPC software. Also, depending on the host, it is often possible for the HPC itself to provide interrupt vectoring, thus eliminating the need for an external interrupt controller entirely. The approach taken in the driver program, described below, implements the HPC as the interrupt controller, with interrupts asserted only by the \overline{RDRDY} pin.

2.3.3 Hardware Synchronization

Figure 7 shows the connections required to implement hardware synchronization between the host and the HPC. In this scheme, there is no host software involved in synchronizing with the HPC; if the host attempts a UPI transfer for which the HPC is not prepared, the host is held in "Wait states" until the HPC is ready. Note that the UPIC register is an exception; Wait states are not to be inserted when the CPU polls the UPI port's status ($UA0 = 0$).

The main advantage of this scheme is speed: the CPU and HPC transfer data as fast as they can both run the transfer loop. (One will generally find that the HPC stays ahead of the CPU; the CPU tends to be in the critical path due to more complex buffer management algorithms.) The main disadvantage is that if the HPC is allowed to be interrupted in the middle of the transfer, the CPU is not free to do anything else at all, including servicing its own interrupts.

In addition to the logic to detect when to hold the host (at the bottom of the figure), additional gating is required on the \overline{UWR} signal, to prevent it from being asserted until the \overline{WRRDY} signal is active. This is required because the IBUF register of the HPC is a fall-through latch, and its contents would be lost if \overline{UWR} were allowed to go active too soon.

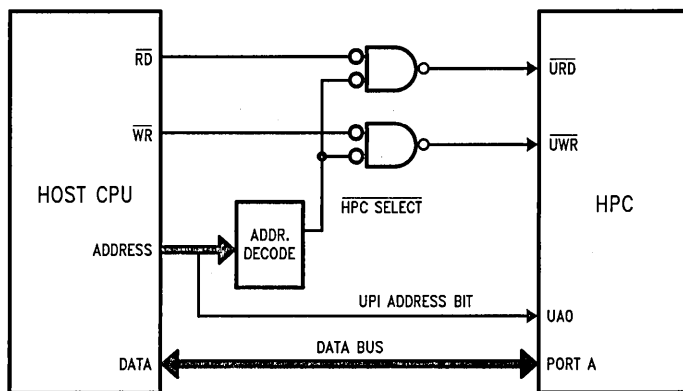


FIGURE 5. Polling Interface

TL/DD/9976-5

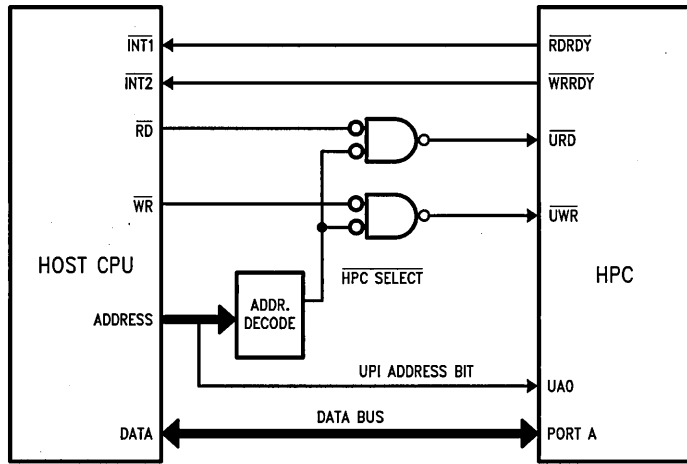


FIGURE 6. Interrupt-Driven Interface

TL/DD/9976-6

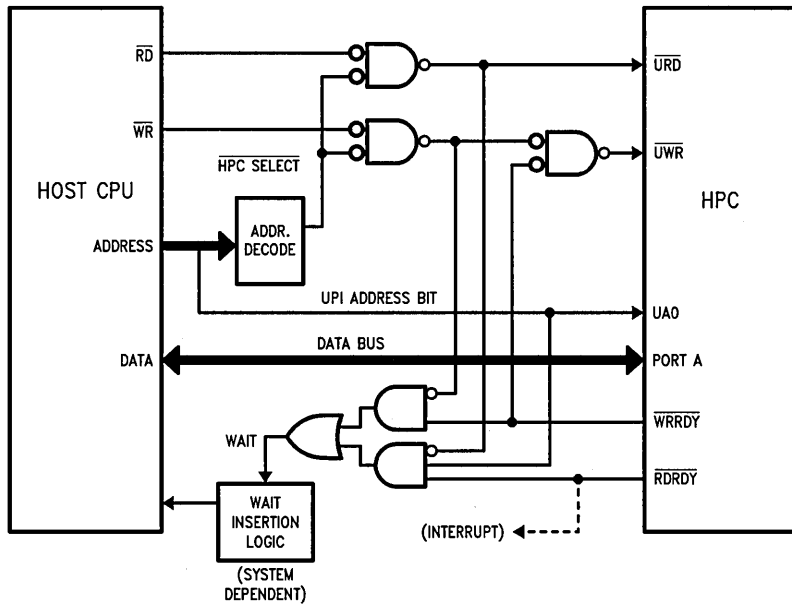


FIGURE 7. Hardware-Synchronized Interface

TL/DD/9976-7

Figures 8 and 9 illustrate the timing involved in hardware synchronization. Figure 8 shows the host attempting two UPI Read accesses in quick succession; the second Read access is held pending until the HPC has supplied the data. Figure 9 shows the host attempting two UPI Write accesses in quick succession; it is held in Wait states (with the \overline{UWR} signal suppressed) until the HPC has emptied the first value from the IBUF register.

This scheme and the interrupt-driven scheme above are not mutually exclusive; as shown in Figure 6, one might tie \overline{RDRDY} or \overline{WRRDY} , or both, to CPU interrupts. The application hardware described implements both schemes, leaving CPU software the option of using hardware synchronization or not. The driver program in the HPC operates the same, independent of the option used.

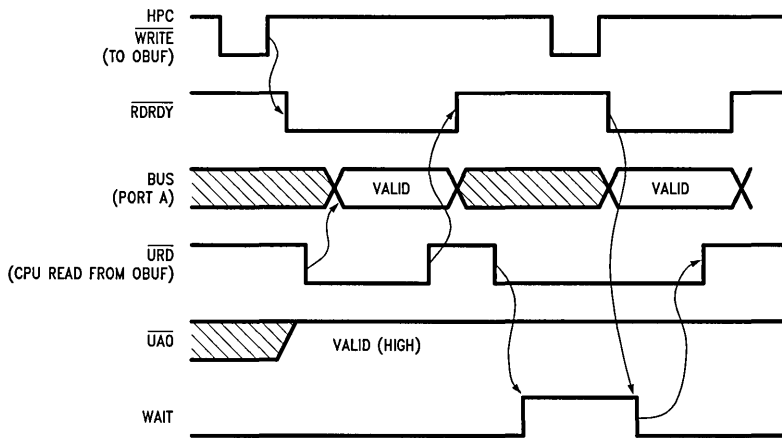


FIGURE 8. Hardware Synchronization: Read Operations

TL/DD/9976-8

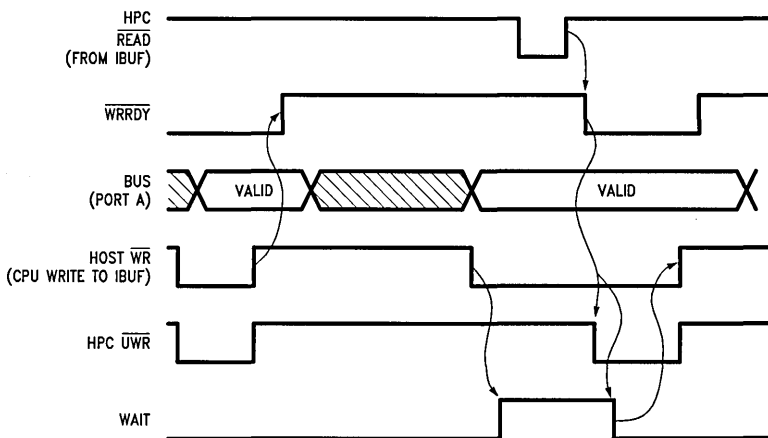


FIGURE 9. Hardware Synchronization: Write Operations

TL/DD/9976-9

3.0 A UPI DRIVER AND SAMPLE APPLICATION

The circuit and program described below implement an interface between the HPC and a National microprocessor, the NS32CG16, as the host CPU. The UPI port is configured to be 8 bits wide. The hardware supports both interrupt-driven (\overline{RDRDY} only) and hardware synchronization, as well as polling.

In order to demonstrate some real commands to support, a set of simple interfaces is attached to the HPC, typical of a front panel.

- Up to 8 pushbuttons
- Up to 8 LED indicators
- A 16-character alphanumeric LCD display
- A speaker for “beeps” on alert conditions or input errors
- A real-time clock interrupt function, giving the CPU the means to measure time intervals accurately.

This application by itself is admittedly not enough to justify the presence of an HPC in a system, but it is a simple application, and we expect that this will often be part of the HPC's job. For a much more comprehensive application, which includes this one as a subset, see the next application note in this series: “The HPC as a Front-End Processor”.

We will describe in this section a specific set of hardware and software, and a UPI command and response protocol to make these interfaces play.

3.1 UPI Port Connections to NS32CG16

The attached schematic shows the HPC UPI port as it has been used a real application. On Sheet 1, a block diagram is given, showing the components involved. The CPU is an

NS32CG16 microprocessor, running at a 15 MHz clock rate (crystal frequency 30 MHz). The HPC component is the HPC46083, running at a crystal frequency of 19.6608 MHz.

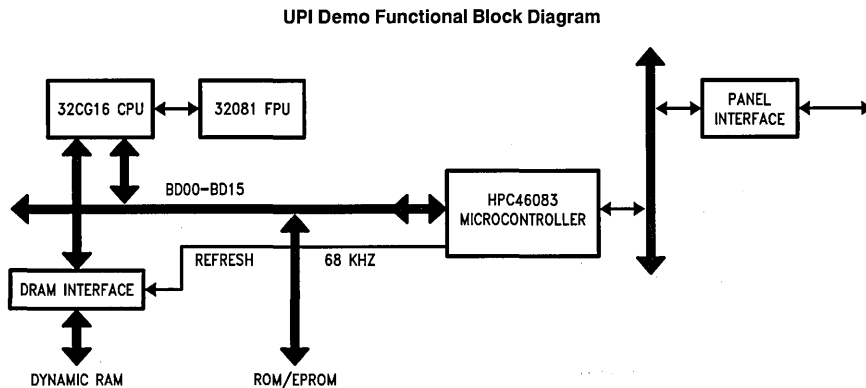
It would be unrealistic to present only the UPI interface section, since tradeoffs and implementation considerations abound when dealing with fast processors and large addressing spaces. For this reason, we include on sheets 5, 6 and 7 the circuitry involved in NS32CG16 address decoding and dynamic RAM control.

The \overline{UREAD} and \overline{UWRITE} UPI strobes are generated for the HPC in area B1 of Sheet 6. In addition, the latched CPU address bit BA09 is used as the UA0 addressing bit.

Hardware and Interrupt synchronization are accomplished as follows. On Sheet 6, area D8, the HPC signals \overline{URDRDY} and $\overline{UWRDRDY}$ enter a synchronizer, and emerge as $\overline{URDRDYS}$ and $\overline{UWRDRDYS}$. The $\overline{URDRDYS}$ signal goes to the CPU as its Maskable Interrupt signal (Sheet 5, area C8). After gating, which yields $\overline{URDRDYSQ}$ and $\overline{UWRDRDYSQ}$, they enter the PAL16L8 in area C7 of Sheet 6. This PAL's relevant outputs are $\overline{WAIT1}$ and $\overline{WAIT2}$, which go to the CPU for Wait State generation, and \overline{ACWAIT} , which also goes to the CPU (as \overline{CWAIT}) after passing through the PAL20R8 device in area D4 of Sheet 6.

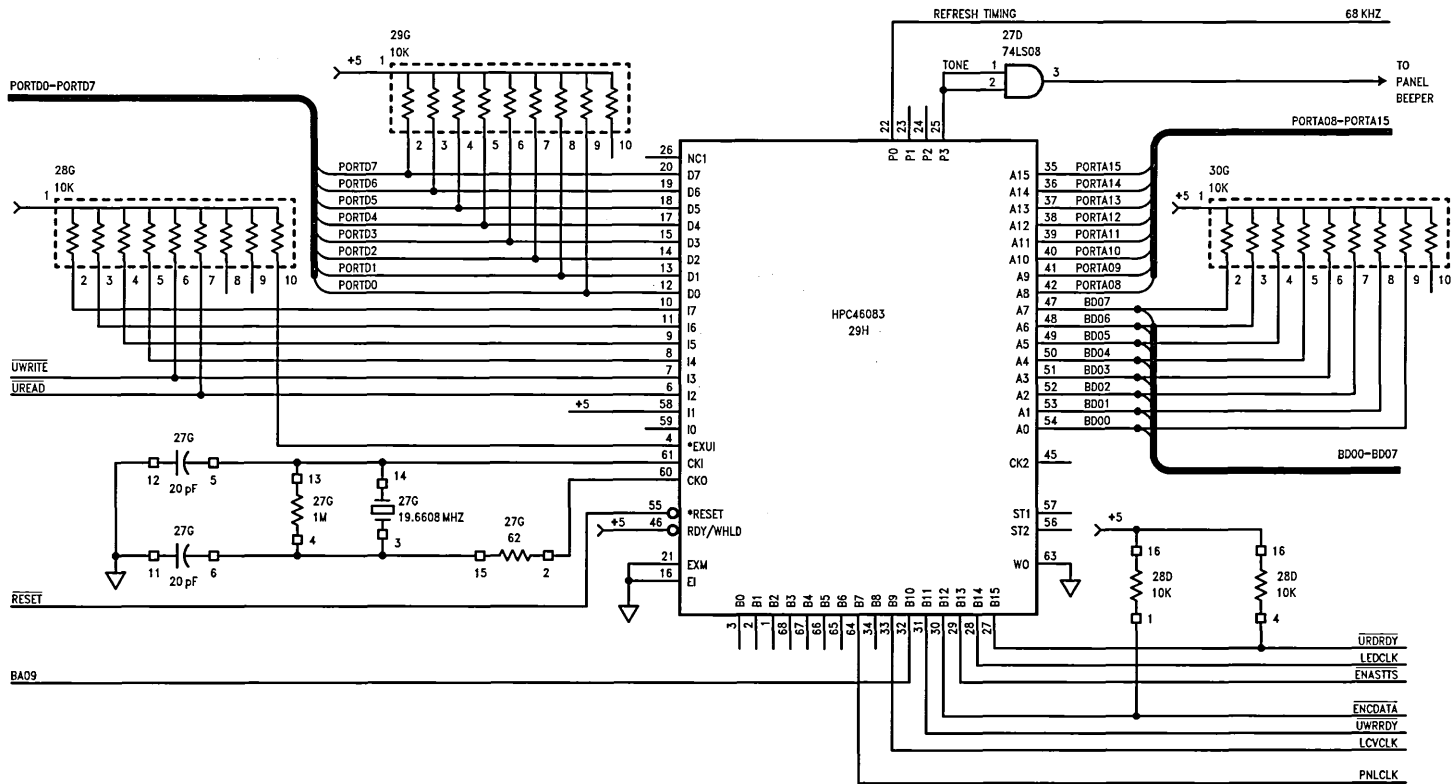
In addition, the HPC provides from Timer T4 a square wave at approximately 68 kHz, which triggers refreshes of dynamic RAM. The signal involved is called “68 kHz”, and goes from the HPC on Sheet 4, area D1, to Sheet 6, area D8. Note that the detector in area D7 is held on at Reset, to preserve RAM contents by continuous refreshing while the HPC is being reset.

3.1.1 Schematic



TL/DD/9976-10

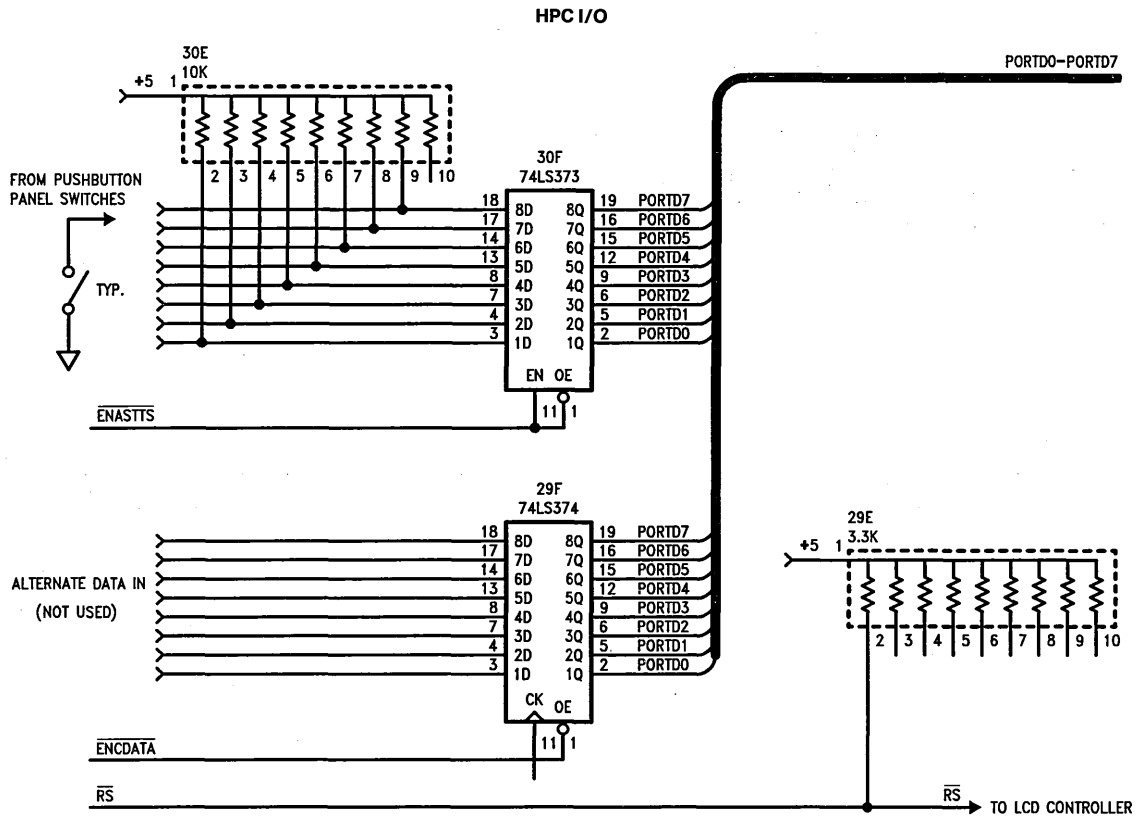
HPC Microcontroller



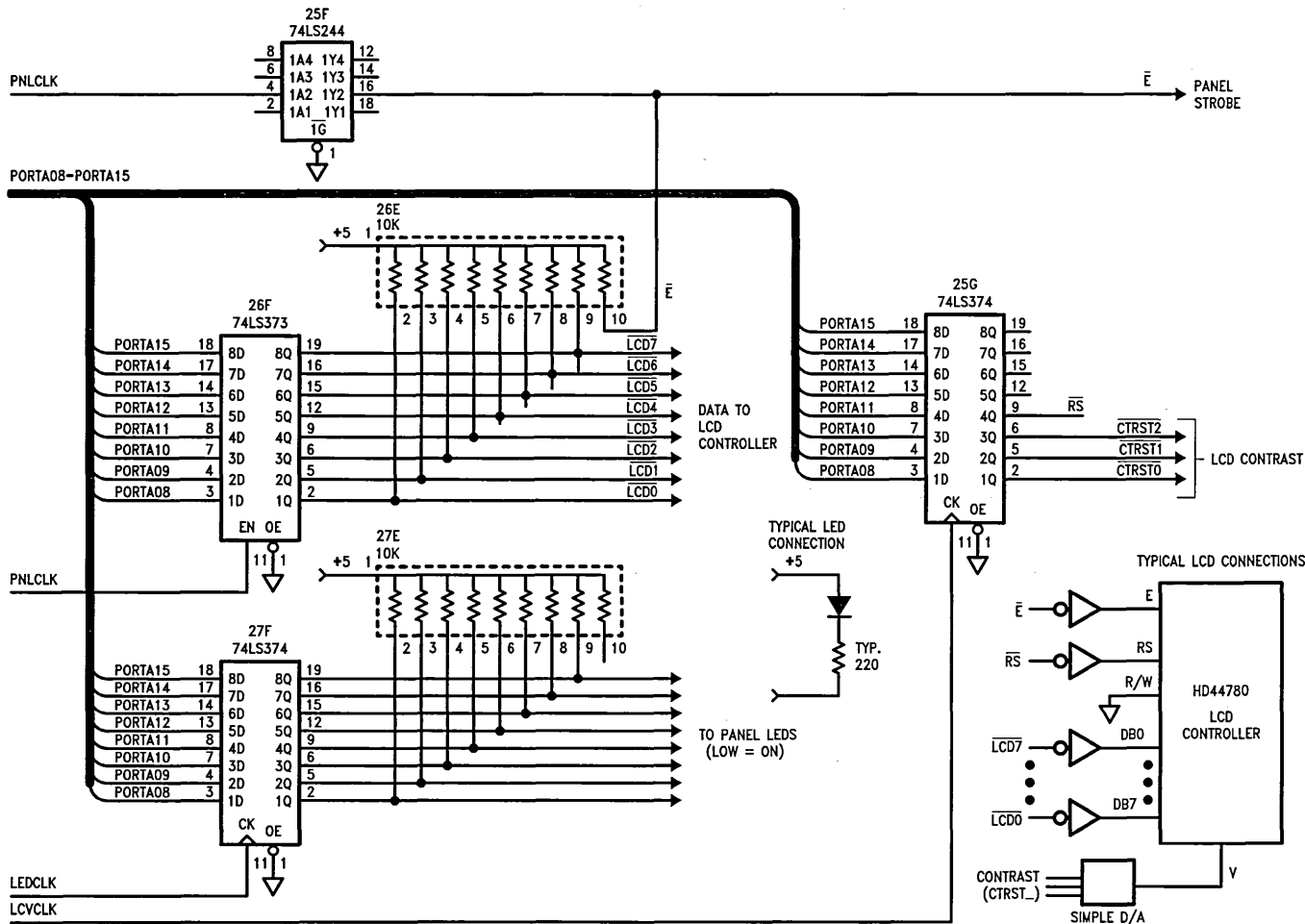
TL/DD/8976-11

5-137

5-138



Panel I/O Interface

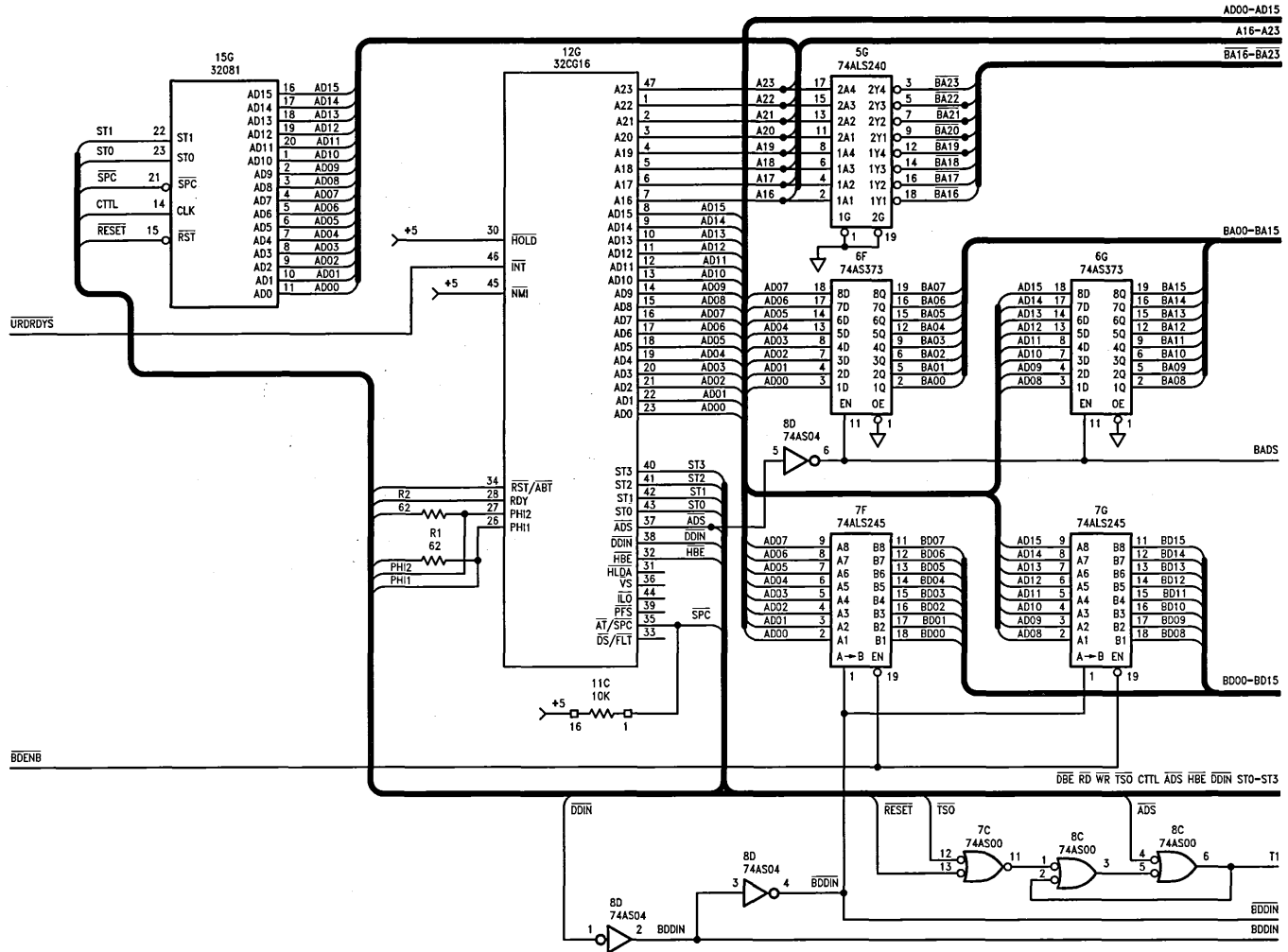


5-139

TL/DD/9976-13

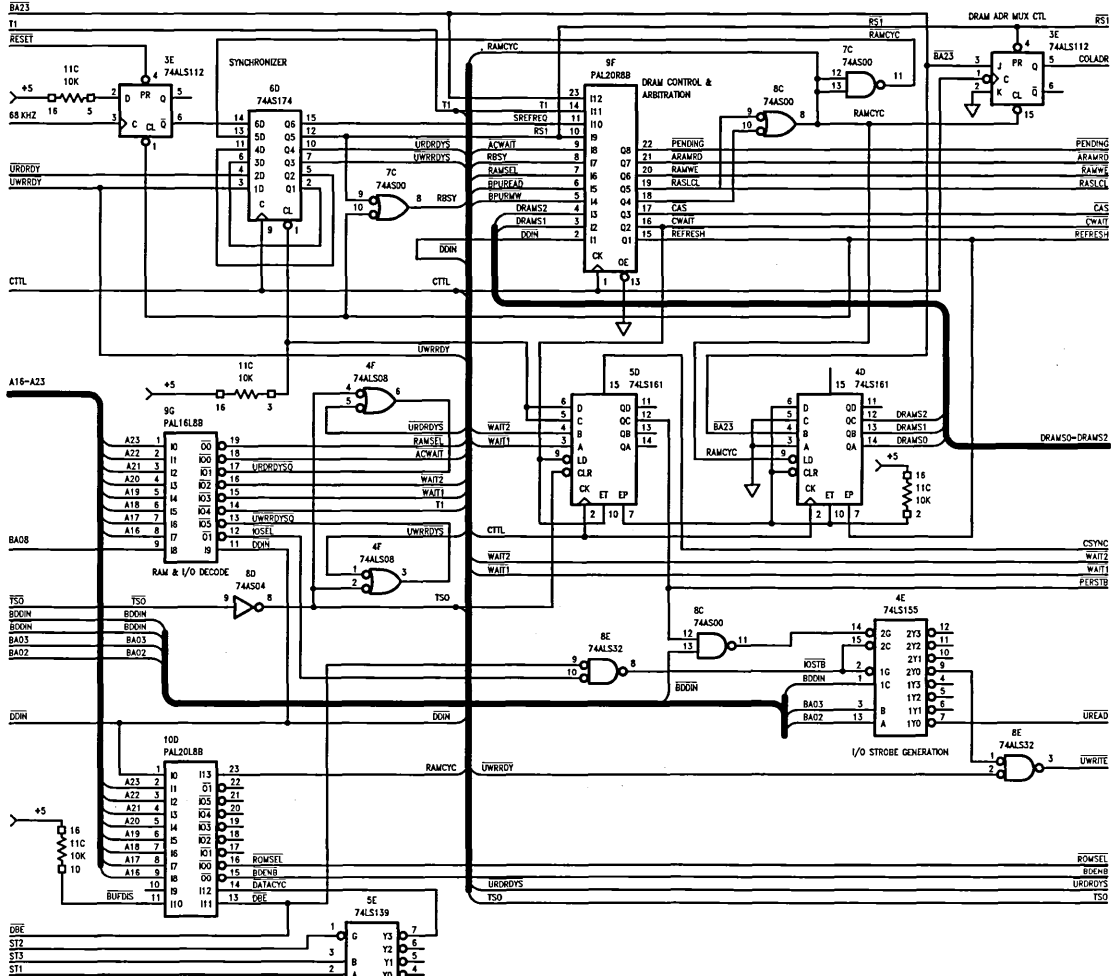


CPU Cluster and Buffering



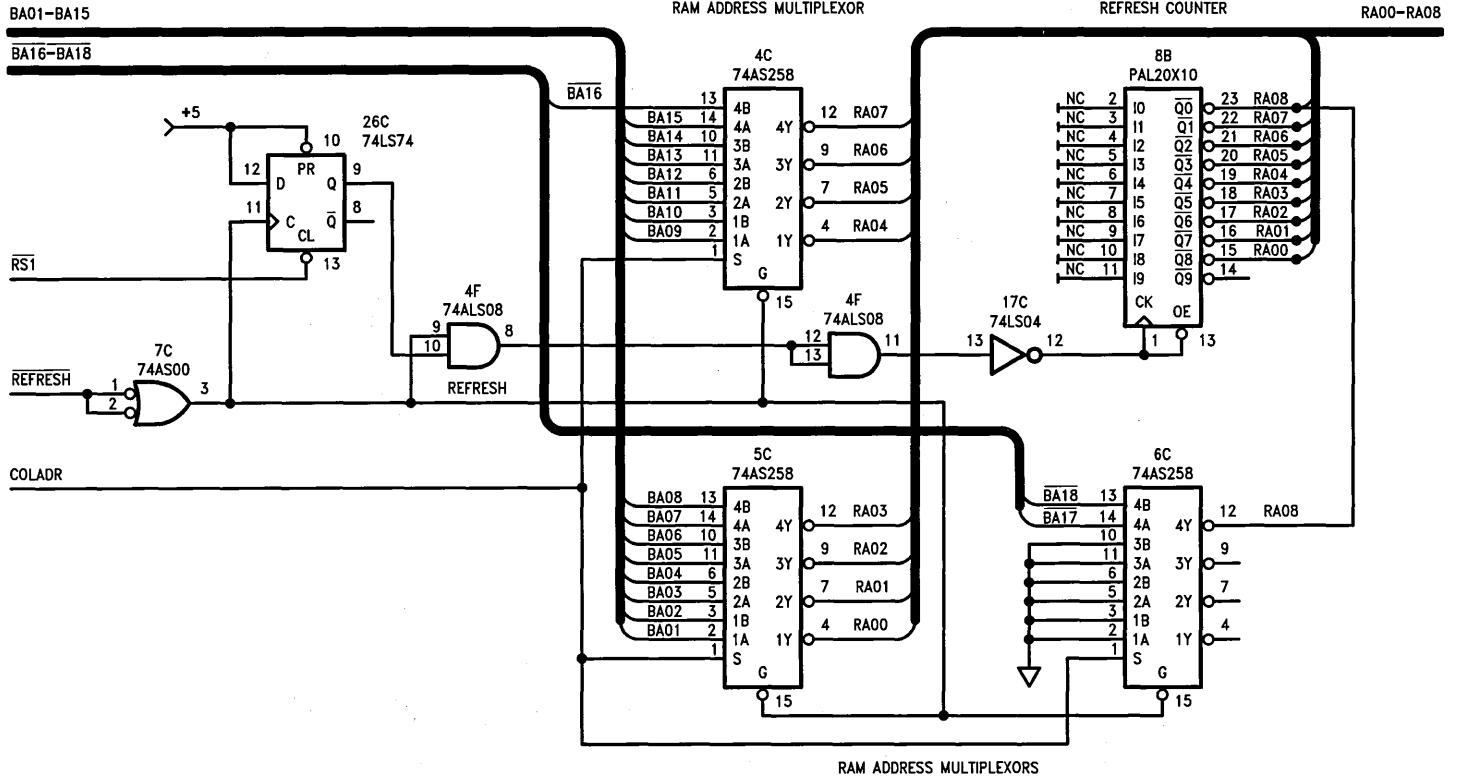
S-140

Address Decoders and Timing Control Logic



5-141

DRAM Address Generation



5-142

RAM ADDRESS MULTIPLEXORS

3.1.2 PAL Equations

Schematic Sheet 7, Area 3D

```

Name      REFRESH.PLD;
Partno    XXXXX;
Date      05/19/87;
Revision  LA;
Designer  FOX;
Company   NSC;
Assembly  X7A;
Location  8B;
Device    p20x10;
/*****
/*
/* REFRESH: 9 BIT REFRESH COUNTER
/*
/*
/*****
/* Allowable Target Device Types: PAL20X10
/*
/*****
/** Inputs **/
Pin 1 = !refresh ;/* refresh pulse
/*
/** Outputs **/
Pin [15..23]= [ra0..8] ;/* ram refresh address
Pin 14 = !refron ;/* refresh enabled output
/** Declarations and Intermediate Variable definitions **/
$define | #
/** Logic Equations **/
!ra0.d = ra0;
!ra1.d = !ra1 $ ra0;
!ra2.d = !ra2 $ ra0 & ra1;
!ra3.d = !ra3 $ ra0 & ra1 & ra2;
!ra4.d = !ra4 $ ra0 & ra1 & ra2 & ra3;
!ra5.d = !ra5 $ ra0 & ra1 & ra2 & ra3 & ra4;
!ra6.d = !ra6 $ ra0 & ra1 & ra2 & ra3 & ra4 & ra5;
!ra7.d = !ra7 $ ra0 & ra1 & ra2 & ra3 & ra4 & ra5 & ra6;
!ra8.d = !ra8 $ ra0 & ra1 & ra2 & ra3 & ra4 & ra5 & ra6 & ra7;
refron.d= 'b'1;

```


Schematic Sheet 6, Area 5D

```

Name      RAM.PLD;
Partno    XXXXX;
Date      07/25/87;
Revision  1A;
Designer  FOX;
Company   NSC;
Assembly  X7A;
Location  9F;
Device    p20r8;
/*****
/*
/* RAM CONTROL: HARDWARE RMW BPU CYCLE, SEPARATE BUSES
/* 6/17: Two States of refadr
/* 6/19: Invert rsl
/*****
/* Allowable Target Device Types: PAL20R8B
/*****

/** Inputs **/

Pin 1 = cttl ; /* clock input
Pin 2 = !ddin ; /* data direction in signal
Pin 3 = drams1 ; /* DRAM state counter, bit 1
Pin 4 = drams2 ; /* DRAM state counter, bit 2
Pin 5 = !bpurmw ; /* BPU read modify write cycle
Pin 6 = !bpuread ; /* BPU source read (comb.)
Pin 7 = !ramsel ; /* Any RAM address decode
Pin 8 = busy ; /* DRAM busy indication (rsl | refresh)
Pin 9 = !acwait ; /* Advanced WAIT from ROM, or I/O
Pin 10 = !rsl ; /* ram cycle delayed by one Tstate
Pin 11 = !srefreq ; /* Refresh Request
Pin 14 = t1 ; /*Processor T1 state
Pin 23 = !a23 ; /* Address 23

/** Outputs **/

Pin 15 = !refresh ; /* refresh cycle
Pin 16 = !cwait ; /* 32C201 cwait
Pin 17 = !cas ; /* CAS, local & cartridge
Pin 18 = !rascart ; /* RAS for DRAM cartridge
Pin 19 = !raslcl ; /* RAS for local DRAM
Pin 20 = !ramwe ; /* DRAM Write enable
Pin 21 = !aramrd ; /* DRAM read
Pin 22 = !pending ; /* DRAM cycle requested, but cttl busy
min [refresh, cwait, cas, rascart, raslcl, ramwe, aramrd, pending] = 2;
/** Declarations and Intermediate Variable Definitions **/
field waitseq = [pending, cwait];
#define wide 0 /* wait sequencer idle */
#define busywt 3 /* wait sequencer waiting for busy DRAM */
#define cextwt 1 /* wait sequencer waiting for cycle extension */

```

Schematic Sheet 6, Area 5D (Continued)

AN-550

```

field ctl = [refresh,cas,raslcl,rascart];
$define idle    00
$define cras    01
$define crascas 05
$define casend  04
$define lras    02
$define lrascas 06
$define refadr  08
$define refras  0b
$define |      #
field drscount = [drams2..drams1];
/** Logic Equations **/
    lcl_sel      = ramsel & !a23;
    cart_sel     = ramsel & a23;
    lclread      = !a23 & ddin;
    lclwrite     = !a23 & !ddin;
    holdoff      rsl;
/*    busy        = refresh | holdoff;    (generated externally)    */
    cart_start   = cart_sel & (tl | pending) & !holdoff;
    local_start  = lcl_sel & (tl | pending) & !holdoff;
    ram_start    = cart_start | local_start;
    drrco        = drscount: [6..7] & ramwe;
sequence waitseq {
/*    acwait & ramsel are mutually exclusive conditions */
present wide    if (ramsel | bpurmw & bpuread) & busy & tl    next busywt;
                 if acwait | (ramsel & !busy & tl & !bpurmw)
                 next cextwt;
                 default next wide;
present busywt  if busy                                          next busywt;
                 if !busy & (bpurmw)                            next wide;
                 if !busy & !(bpurmw)                          next cextwt;
present cextwt  if ramsel & drscount: [0..1] | acwait next cextwt;
                 default next wide;
}
sequence ctl {
present idle    if cart_start                                    next cras;
                 if local_start                                next lras;
                 if !ram_start & srefreq                       next refadr;
                 default next idle;
present cras    if !rsl                                          next cras;
                 if rsl                                         next crascas;
present crascas if (!bpurmw & drscount: [4..7]) | (bpurmw & drrco)
                 next casend;
                 default next crascas;
present lras    next lrascas;

```

5

Schematic Sheet 6, Area 5D (Continued)

```

present lrascas if (!bpurmw & drscount: [4..7]) | (bpurmw & drrco)
                                next casend;

                                default next lrascas;
present casend if srefreq                                next refadr;
                if !srefreq                                next idle;
present refadr if srefreq                                next refadr;
                if !srefreq & !rsl                        next refras;
                if !srefreq & rsl                          next idle;
present refras if ramwe                                next refadr;
                default next refras;
}

/* remember ramwe & aramrd are delayed by one t-state */
ramwe.d = !refresh & (bpurmw & drscount: [6..7] & !ramwe
                    | !bpurmw & !ddin & (ram_start | ctl: cras
                    | (cart_sel & drscount: [0..3]) | ctl:lras)
                    )
| ctl:refras & rsl & !ramwe;

aramrd.d = (bpurmw & drscount: [0..3] | !bpurmw & ddin)
& (ctl:cras | ctl:crascas | ctl:lras | ctl:lrascas);

```

Schematic Sheet 6, Area 7C

AN-550

Name DCD1.PLD;
 Date 07/03/87;
 Revision 1A;
 Designer FOX;
 Company NSC;
 Assembly X7A;
 Location 9G;
 Device pl618;

```

/*****
*/
/* DECODE 1: I/O DECODE, PROM & HPC I/F WAIT CONTROL */
/* 6/3: two waits for hpc write */
/* 6/4: 1 wait min. for ALL i/o, including HPC */
/* 6/4: 3 wait min. for i/o */
/*
/*****
/* Allowable Target Device Types: PAL16L8B */
/*****
/** Inputs **/
Pin [1..8] = [a23..16] /* high order address bus */
Pin 9 = ba8 /* address bit 8 */
Pin 11 = !ddin /* cpu ddin/ */
Pin 13 = !uwrrdys /* (HPC) UWRRDY/, synchronized */
Pin 14 = t1 /* T1 state of CPU */
Pin 17 = !urdrdys /* (HPC) URDRDY/, synchronized */

/** Outputs **/
Pin 12 = !iosel /* I/O select decode */
Pin 15 = !wait1o /* WAIT1 output */
Pin 16 = !wait2o /* WAIT2 output */
Pin 18 = !acwait /* Advance CWAIT for RAM ctl */
Pin 19 = !ramsels /* DRAM address decode */

/** Declarations and Intermediate Variable Definitions **/
#define | #
field address = [a23..16] /* address field */
field waitv = [acwait,wait2o,wait1o]; /* wait value field */
#define nowaits "b'000
#define wait1v ("b'100 & t1)
/* note use of # in next 3 defines because $define not nestable */
#define wait2v ("b'101 & ("b'011 # "b'100 & t1))
#define wait3v ("b'110 & ("b'011 # "b'100 & t1))
#define wait4v ("b'111 & ("b'011 # "b'100 & t1))
#define cwaitonly "b'100

/** Logic Equations **/

ramsels = address: [0780000..07ffff] | address: [0800000..0bffff];
iosels = address: [0fd0000..0ffff] & !ba8;
waitv = wait3v & address: [0000000..00ffff] /* main rom, 3 waits */
| wait4v & address: [0200000..05ffff] /* font rom, 4 waits */
| wait3v & address: [0fd0000..0ffff] & !ba8 /* i/o, 1 wait */
| cwaitonly & address: 0ff0000 & !ba8 &
(| !urdrdys & ddin | !uwrrdys & !ddin);

```

5

Schematic Sheet 6, Area 7A

```

Name      DCD2.PLD;
Partno    XXXXX;
Date      07/27/87;
Revision  1C;
Designer  FOX;
Company   NSC;
Assembly  X7A;
Location  10D;
Device    p2018;
/*****
*/
/* DECODE 2: ROM DECODE, BUFFER CONTROL, BPU DECODE
*/ 5/24: included enbpu in bpucyc generation
*/ 5/28: added bpucyc to rdenb
*/ 5/31: added fcxxxx to bdenb
*/ 6/23: added buffer disable term for SPLICE
*/ 7/25: reconfigured for bpurmw & bpuread
*/ 7/27: inverted polarity of enbpu ≥ enablebpu (for master enb)
/*****
*/ Allowable Target Device Types: PAL20B
/*****
/** Inputs **/
Pin      1      = !ddin      ;/* ddin/ from cpu
Pin      = [2..9]=[a23..16] ;/* high order address bus
Pin      10     = !enablebpu ;/* BPU enable, static bit
Pin      11     = !bufdis    ;/* buffer disable
Pin      13     = !dbe       ;/* dbe/ from tcu
Pin      14     = !datacyc   ;/* data cycle status decode
Pin      23     = !ramcyc    ;/* ram cycle in progress
/** Outputs **/
Pin      15     = !bdenb     ;/* BD bus enable
Pin      16     = !romsel    ;/* Main rom select
Pin      17     = !romcart   ;/* rom cartridge select
Pin      18     = !bpurmw    ;/* BPU read modify write
Pin      19     = !bpuread   ;/* BPU read cycle (comb.)
Pin      20     = !vramsel   ;/* video ram select
Pin      21     = !rdbufin   ;/* RAM data bus direction (in)
Pin      22     = !rdenb     ;/* RAM data bus enable
/** Declarations and Intermediate Variable Definitions **/

field address = [a23..16] ;/* address field
romspace      = address: [0000000..05fffff];
ramspace      = address: [0780000..0bfffff];
stack         = address: [0780000..078fffff];

$define |      #
      min b_ddin = 0;
/** Logic Equations **/
romsel = address: [0000000..00fffff]; /* main rom
romcart = address: [0200000..05fffff]; /* font rom

```

Schematic Sheet 6, Area 7A (Continued)

```

vramsel = address: [0f00000..0f0ffff];          /* video ram (scan buffer) */
/*
/*      bpucyc & b_ddin are D latches implemented in the PAL
/*
/*      basic d latch equation (w/o set or clear) is:
/*          Q = (G & D) | (!G & Q) | (D & Q)
/*
/*      The b_ddin latch is fall through while ramcyc not asserted,
/*      latched while ramcyc is asserted, therefore, for both latches:
*/
g          = !ramcyc;
/*
/*      The bpurmw latch d input is "bpurange", defined as:
*/
bpurange= address: [0000000..05ffffff]          /* rom          */
          | address: [0790000..0bffffff];        /* dram, less stack */
/*
/*      This "d" input would use too many terms. The bpucyc output,
/*      however, need only be latched when it is asserted, as this is
/*      the situation that can allow the cpu and ram control to
/*      not be synchronized. This simplification allows the simplification
/*      of the latch to:
/*          Q = D | (!G & Q)
*/
bpurmw = enablebpu & (!ddin & bpurange & datacyc | (!g & bpurmw));
bpuread = enablebpu & ddin & bpurange & datacyc;
/*
/*      rdenb enables cpu access to the ram data bus
*/
rdenb    = dbe & bufdis &
          ( !bpurmw & bpuread & romspace          /* buffer must be off for bpu
                                                  /* but on for source in rom */
          /* no DRAM or bpu control writes are permitted */
          /* while in inner loop of bitblt */
          /* (within interrupt ok due to vector read!)
          | romspace
          | address: [0fe0000..0feffff]);          /* i/o access to bpu */
!rdbufin = (romspace | address: [0fe0000..0feffff]) & !ddin
          | romspace & bpuread;
bdenb    = dbe & !bufdis & (romspace          /* any rom */
          | address: [0f00000..0f0ffff] /* scan buffer */
          | address: [0fd0000..0fdffff] /* cmd/status */
          | address: [0ff0000..0ffffff] /* non-bpu i/o */

```

3.2 Application Connections

The connections made to the HPC are shown in schematic sheets 2 through 4.

3.2.1 LCD Data

An 8-bit parallel interface connects the upper half of Port A, through buffers and latches on Sheet 4, to a Hitachi HD44780 alphanumeric LCD display controller. The signals in our application are inverted with respect to the HD44780 documentation, due to the nature of the front panel module we used.

Sending data from the HPC to the LCD display involves the following procedure:

1. Setup the \overline{RS} signal: 1 for a command, 0 for data.
This is done by setting up LCD Contrast status on the high-order byte of Port A (pins A8–A15), with the desired \overline{RS} state on pin A11, then pulsing the signal LCVCLK (pin B9) high, the low.
2. Setup the panel data on HPC pins A8–A15.
3. Set the PNLCLK signal (pin B7) low for 1.2 μ s, then high.
This clocks the data into the LCD display controller. Note that the latch in area B6 of Sheet 4 is effectively serving only as a buffer; the PNLCLK Enable signal, being normally high, allows data to fall through whenever it changes when used as described here.
4. Since the handshaking capability of the HD44780 is not being used here, it is necessary for the HPC to use an internal timer to determine when the controller is ready after sending a command or data. The delay time is either 120 μ s or 4.9 ms, depending on the type of command sent.

3.2.2 LCD Contrast (LCD Voltage)

A three-bit value is presented for LCD contrast on signals $\overline{CTRST0}$ through $\overline{CTRST2}$. A value of 000 is highest contrast, and 111 is lowest contrast. To change the contrast, the value is placed on HPC pins A8 (LSB), A9 and A10 (MSB), the LCVCLK (pin B9) is pulsed high, then low.

Note that some other bits within this latch have other functions: bit 3 (from HPC pin A11) is the \overline{RS} signal to the LCD controller, and bit 7 (from pin A15) is used by the HPC firmware as a Fatal Error flag. These bits must be setup correctly whenever the LCD Contrast latch is written to.

3.2.3 LEDs

Up to 8 LED indicators may be connected, through the latch in area A6 of Sheet 4, to the upper byte of Port A. The LED's are assumed to be connected already to their own current-limiting resistors.

The desired data is setup on Port A pins A8–A15, then a pulse is presented on the LEDCLK signal (pin B14); high and then low. Data is presented in complemented form by the HPC (0 = on, 1 = Off). Any or all (or none) of the latch bits may be connected to drive LEDs.

3.2.4 Speaker (Beeper)

A tone is produced on a speaker by enabling Port P pin P3 as the Timer T7 output, and running Timer T7 so as to produce a 3 kHz square wave. Since timer outputs toggle on underflows, this corresponds to a timer underflow rate of 6 kHz. The tone signal is shown in area D1 of Sheet 2.

3.2.5 Pushbutton Switches

Up to eight pushbuttons may be connected to the HPC's Port D pins, through the buffer in area D6 of Sheet 3. Each

pushbutton is assumed to be an SPST switch, shorting to ground when depressed. The pull-up resistors present a "1" level otherwise. The HPC must de-bounce the inputs in its firmware before issuing them to the CPU.

The pushbuttons are examined every 10 ms, by setting the ENASTTS signal (pin B13) low while ensuring that $\overline{ENCDATA}$ (pin B12) is high. This presents the switch outputs onto Port D. Unused bits should be pulled high to avoid triggering spurious pushbutton events.

3.3 Protocol Between CPU and HPC

The scheme supported by the UPI Driver program is asynchronous full-duplex communication with CPU. That is, either side is allowed to speak at any time. To avoid confusion, however, any message is restricted to send data in only one direction: in sequences initiated by the CPU ("Command" sequences), only the CPU talks, and in sequences initiated by the HPC ("Interrupt" sequences), only the HPC talks. Thus, a Command sequence and an Interrupt sequence can be in progress simultaneously without confusion.

Acknowledgement of a Command or an Interrupt sequence is possible; a Command can trigger an acknowledgement Interrupt sequence, and an Interrupt sequence can result in a subsequent Command sequence. The critical distinction, though, is that the acknowledgement need not come immediately. If, for example, the HPC is already in the process of sending an Interrupt message, and receives a Command, it will complete the current Interrupt sequence before acknowledging the Command with a new Interrupt.

Command sequences (from the CPU to the HPC) consist of a one-byte command code, followed by any argument values necessary to complete the command. Each byte written to the HPC triggers an internal interrupt (I3); the HPC buffers up these bytes until a full command has been received, then acts on it in the last byte's interrupt service routine. Commands taking a significant amount of processing time can be scheduled within the HPC using interrupts, either from external events or from one of the HPC's eight timers; each interrupt triggering the next step of the command.

Interrupt sequences (from the HPC to the CPU) operate similarly, but with a small difference. Only the first byte presented by the HPC causes an interrupt to the CPU; this byte is the interrupt vector value, which triggers the interrupt (through the \overline{RDRDY} pin) and selects the CPU's service routine. The CPU remains in its interrupt service routine until the transfer of data associated with that interrupt event is finished, then returns to its previous task. This is not to say that the CPU must keep all other interrupts disabled during an Interrupt sequence, but only that no other interrupt occurring during this time may cause the CPU to read from the HPC, or to terminate reading, until the current Interrupt sequence is complete. With the NS32C016 processor as host, the main challenge is to keep the Interrupt Acknowledge bus cycles from other interrupts, which appear as Read cycles, from causing URD pulses to the HPC. It is possible to distinguish a Non-Maskable Interrupt from a Maskable Interrupt by the address asserted by the CPU in acknowledging the interrupt, and in a larger kind of system containing an NS32202 Interrupt Control Unit, the NS32000 Cascaded Interrupt feature can be used to prevent unwanted reads from the HPC from occurring as a result of other Maskable interrupts as well. In our application hardware, the only type of extraneous interrupt occurring is the Non-Maskable Interrupt; address decoding logic isolates the HPC's UPI port from these.

3.4 Commands

The first byte (command code) is sent to address FFFC00, and any argument bytes are then written to address FFFE00. The CPU may poll the UPIC register at address FD0000 to determine when the HPC can receive the next byte, or it can simply attempt to write, in which case it will be held in Wait states until the HPC can receive it. Unless noted, the CPU may send commands continuously, without waiting for acknowledgement interrupts from previous commands.

- 00 INITIALIZE This command has two functions. The first INITIALIZE command after a hardware reset (or RESET command) enables the !RTC and !BUTTON-DATA interrupts. The INITIALIZE command may be re-issued by the CPU to either start or stop the !RTC interrupts. There is one argument:
 RTC-Interval: One-byte value. If zero, !RTC interrupts are disabled. Otherwise, the !RTC interrupts occur at the interval specified (in units of 10 ms per count).
- 01 SET-CONTRAST The single argument is a 3-bit number specifying a contrast level for the LCD panel (0 is least contrast, 7 is highest contrast). There is no response interrupt. Does not require INITIALIZE command first.
- 02 SEND-LCD This writes a string of up to 8 bytes to the LCD panel. Arguments are:
 flags: a single byte, containing the RS bit associated with each byte of data. The first byte's RS value is in the least-significant bit of the FLAGS byte.
 #bytes: The number of bytes to be written to the LCD display.
 byte[1]—byte[#bytes]: The data bytes themselves.
 The HPC determines the proper delay timing required for command bytes (RS = 0) from their encodings. This is either 4.9 ms or 120 μ s.
 The response from the HPC is the IACK-SEND-LCD interrupt, and this command must not be repeated until the interrupt is received. This command does not require an INITIALIZE command first.
- 03 SEND-LED The single argument is a byte containing a "1" in each position for which an LED should be lit.
 There is no response interrupt, and this command does not require the INITIALIZE command first.
- 04 BEEP No arguments. This beeps the panel for approximately one second. No response interrupt. If a new BEEP command is issued during the beep, no error occurs (the buzzer tone is extended to one second beyond the most recent command). Does not require INITIALIZE command first.

- A5 RESET-HPC Resets the HPC if it is written to address FFFC00. It may be written at any time that the UPI port is ready for input; it will automatically cancel any partially-entered command. The CPU's Maskable Interrupt must be disabled before issuing this command.

After issuing this command, the CPU should first poll the UPIC register at address FD0000 to see that the HPC has input the command (the least-significant bit [Write Ready] is zero). It must then wait for at least 25 μ s, then read a byte from address FFFE00. The HPC now begins its internal re-initialization. The CPU must wait for at least 80 μ s to allow the HPC to re-initialize the UPI port. Since part of the RESET procedure causes Ports A and B to float briefly (this includes the CPU's Maskable Interrupt input pin), the CPU should keep its maskable interrupt disable during this time. It also must not enter a command byte during this time because the byte may be lost.

3.5 Interrupts

The HPC interrupts the CPU, and provides the following values as the interrupt vectors for the CPU hardware. The CPU then reads data from the HPC at address FFFE00. All data provided by the HPC must be read by the CPU before returning from the interrupt service routine, otherwise the HPC would either hang or generate a false interrupt. The CPU may poll the UPIC register at address FD0000 to determine when each data byte is ready, or it may simply attempt to read from address FFFE00, and it will be held in Wait states until the data is provided by the HPC.

Note: All CPU interrupt service routines, including the NMI interrupt routines, must return using the "RETI 0" instruction. Do NOT use "RETI".

- 00–0F (Reserved for CPU internal traps and the NMI interrupt.)
- 11 !RTC Real-Time Clock Interrupt. No data returned. Enabled by INITIALIZE command if interval value supplied is non-zero. Note: this version of HPC firmware issues a non-fatal !DIAG interrupt if the CPU fails to service each !RTC interrupt before the next one becomes pending.
- 17 IACK-SEND-LCD This is the response to the SEND-LCD command, to acknowledge that data has all been written to Panel LCD display. No other data is provided with this interrupt. Always enabled, but occurs only in response to a SEND-LCD command.
- 18 !BUTTON-DATA Pushbutton status has changed: one or more buttons have been either pressed or released. The new status of the switches is reported in a data byte, encoded as follows:
 Any pushbutton that is depressed is presented as a "1". All other bit positions, including unused positions, are zeroes. The pushbuttons are debounced before being reported to

1D IDIAG

the CPU. This interrupt is enabled by the first INITIALIZE command after a reset.

Diagnostic Interrupt. This interrupt is used to report failure conditions and CPU command errors. There are five data bytes passed by this interrupt:

- Severity
- Error Code
- Data in Error (passed, but contents not defined)
- Current Command (passed, but contents not defined)
- Command Status (passed, but contents not defined)

The Severity byte contains one bit for each severity level, as follows:

x	x	x	F	x	x	C	N
---	---	---	---	---	---	---	---

N (Note): least severe. The CPU missed an event; currently only the IRTC interrupt will cause this.

C (Command): medium severity. Not currently implemented. Any command error is now treated as a FATAL error (below).

F (Fatal): highest severity: the HPC has recognized a non-recoverable error. It must be reset before the CPU may re-enable its Maskable Interrupt. In this case, the remaining data bytes may be read by the CPU, but they will all contain the value 1D (hexadecimal). The CPU must issue a RESET command, or wait for a hardware reset. See below for the procedure for FATAL error recovery.

The Error Code byte contains, for non-FATAL errors, a more specific indication of the error condition:

RTC	(Reserved for COMMAND)
-----	------------------------

RTC = Real-Time Clock overrun: CPU did not acknowledge the RTC interrupt before two had occurred.

The other bits are reserved for details of Command errors, and are not implemented at this time.

The remaining 3 bytes are not yet defined, but are intended to provide details of the HPC's status when an illegal command is received.

Note: Except in the FATAL case, all 5 bytes provided by the HPC *must* be read by the CPU, regardless of the specific cause of the error.

Fatal Error Recovery:

When the HPC signals a IDIAG error with FATAL severity, the CPU may use the following procedure to recover:

1. Write the RESET command (A5 hex) to the HPC at address FFFC00.

2. By inspecting the UPIC register at address FD0000, wait for the HPC to read the command (the *WRRDY bit will go low).

3. Wait an additional 25 μ s.

4. Read from address FFFE00. This will clear the OBUF register and reset the Read Ready status of the UPI port. The HPC will guarantee that a byte of data is present; it is not necessary to poll the UPIC register. This step is necessary because only a hardware reset will clear the Read Ready indication otherwise (HPC firmware cannot clear it).

5. Wait at least 80 μ s. This gives the HPC enough time to re-initialize the UPI port.

6. After Step 5 has been completed, the CPU may re-enable the Maskable Interrupt and start issuing commands. Since the HPC is still performing initialization, however, the first command may sit in the HPI IBUF register for a few milliseconds before the HPC starts to process it.

4.0 SOURCE LISTINGS AND COMMENTARY

4.1 HPC Firmware Guide

Refer to this section for help in following the flow of the HPC firmware in the listing below. Positions in the code are referenced by assembly language labels rather than by page or line numbers.

The firmware for the HPC is almost completely interrupt-driven. The main program's role is to poll mailboxes that are maintained by the interrupt service routines, and to send an interrupt to the CPU whenever an HPC interrupt routine requests one in its mailbox.

On reset, the HPC firmware begins at the label "start". However, the first routine appearing in ROM is the Fatal Error routine. This was done for ease of breakpointing, to keep this routine at a constant address as changes were made elsewhere in the firmware.

4.1.1 Fatal Error Routine

At the beginning of the ROM is a routine (label "hangup") that is called when a fatal error is detected by the HPC. This routine is usually called as a subroutine (although it never returns). It disables HPC internal interrupts, and then sets bit 7 of the LCD Contrast Latch as a trigger for a logic analyzer, MOLE or ISE system.

Its next action is to display its subroutine return address in hexadecimal on the LCD panel. This address shows where the error was detected. The HPC then enters an infinite loop, which continuously presents the IDIAG interrupt. It may be terminated either by a hardware reset or by sending the RESET command from the CPU. On receiving the RESET command, the HPC jumps to label "xreset", which is within the command processing routine. The "xreset" rou-

time waits for the CPU to read from the UPI port, then clears a set of registers to simulate a hardware reset and jumps to the start of the program.

4.1.2 Initialization

On receiving a Reset signal, the HPC begins execution at the label "start". A required part of any application is to load the PSW register, to select the desired number of Wait states (without this step, the Reset default is 4 Wait states, which is safe but usually unnecessary).

Other initializations here are application-dependent, and so they relate to our application system and front-panel operations.

At label "srfsh", the program starts the Refresh clock pulses running for the dynamic RAM on our application hardware, from HPC pin P0 (controlled by Timer T4). For debugging purposes, a circuit within the RAM controller section performs continuous refreshes during Reset pulses, so data in dynamic RAM is never lost unless power is removed.

At "supi", the UPI port is initialized for transfers between the HPC and the CPU.

At label "sram", all RAM within the HPC is initialized to zero. This is done for debugging purposes, to help ensure that programming errors involving uninitialized data will have more consistent symptoms.

At "sskint", the stack pointer is initialized to point to the upper bank of on-chip RAM (at address 01C0). The address of the fatal error routine "hangup" is then pushed, so that it will be called if the stack underflows. This is not necessary in all applications, since the Stack Pointer starts at address 0002, but for our purposes it was more convenient to relocate it.

At "tmnit", the timers T1–T3 are stopped and any interrupts pending from timers T0–T3 are cleared.

In addition, some miscellaneous port initializations are performed here. The upper byte of Port A is set as an output port (for data going to the LCD and LED displays), and the Port B pins which select pushbutton data are initialized.

At "sled", the LED control signals are initialized, and all LED indicators on the panel are turned off.

At "stmrs", all timers are loaded with their initial values, and timers T5–T7 are stopped and any interrupts pending from them are cleared. (Timer T4 keeps running for dynamic RAM refresh.)

At "sled", the panel LCD display is initialized to a default contrast level of 5, then commands are sent to initialize it to 8-bit, 2-line mode, with the cursor visible and moving to the right by default. This section calls a subroutine "wrpn!", located at the end of the program, which simply writes the character in the accumulator out to the LCD display and waits for approximately 10 ms. Note that if the CPU fails to initialize the LCD display further, a single cursor (underscore) character is all that appears: a recognizable symptom of a CPU problem.

The program now continues to label "minit", which performs some variable initializations which are necessary for operation of the UPI Driver itself (as opposed to the application). This much must always be present, but any other initializations required by the application should appear here as well. For our front-panel application, there are no such initializations required.

At label "runsys", the necessary interrupts are enabled (from the timers, and from pin I3, which is the UPI port interrupt from the CPU), and the program exits to the Main Program loop at label "mainlp".

4.1.3 Main Program (UPI Output to CPU)

The Main Program is the portion of the UPI Driver that runs with interrupts enabled. It consists of a scanning loop at label "mainlp", calling a set of subroutines (explained below). It is responsible for interrupting the CPU and passing data to it. The HPC is allowed to write data to the CPU only after interrupting it. The main loop scans a bit-mapped variable in on-chip RAM that is set up by interrupt service routines (a word called "alert") to determine whether any conditions exist that should cause an interrupt to the CPU.

The "alert" word contains one bit for each interrupt that the HPC can generate. If a bit is set (by an interrupt service routine), the Main Program jumps to an appropriate subroutine to notify the CPU. Each subroutine first checks whether the UPI interface's OBUF register is empty, and if not, it waits (by calling the subroutine "rdwait"). It then writes the 32000 interrupt vector number to the OBUF register. This has the effect of interrupting the CPU (Because the pin \overline{URDRDY} goes low), and the CPU hardware reads the vector from the OBUF register. If there is more information to give to the CPU, the HPC places it, one byte at a time, into the OBUF register, waiting each time for OBUF to be emptied by the CPU. This technique assumes that the CPU remains in the interrupt service routine until all data has been transferred. If the CPU were to return from interrupt service too early, the next byte of data given to it would cause another interrupt, with the data value taken as the vector number. (Note, however, that a Non-Maskable interrupt is allowed. It simply delays the process of reading data from the HPC. Since the HPC is running its main program at this point, with its internal interrupts still enabled, it is not stalled by this situation.)

Subroutines called from the Main Program loop are:

- sndrtc: sends a Real-Time Clock interrupt to the CPU. No data is transferred; only the interrupt vector.
- sndlak: interrupts the CPU to acknowledge that a string of data (from a SEND-LCD command) has been written to the LCD display. No data is transferred for this interrupt.
- sndbtn: interrupts the CPU to inform it that a pushbutton has been pressed or released. A data byte is transferred from variable "swlsnt", which shows the new states of all the pushbuttons.
- sndiag: interrupts the CPU to inform it of a DIAG interrupt condition, when it is of NOTE severity. (Other !DIAG conditions are handled at label "hangup".)

4.1.4 Interrupt Service Routines

All of the remaining routines are entered by the occurrence of an interrupt.

4.1.4.1 UPI Port Input from CPU (Interrupt I3)

This interrupt service routine, at label "upiwr", accepts commands from the CPU. Each byte of a command triggers an interrupt on the I3 pin. When the last byte is received, the command is processed before the I3 interrupt routine returns. The HPC is therefore immediately ready to start collecting another command.

Any command that involves waiting is only initiated before the I3 routine returns, and interrupts are set up to activate more processing when the time is right. Therefore, this interrupt service routine returns promptly, even for time-consuming commands.

At any time, the "upiwr" routine may be in one of the following states:

1. Waiting for the first byte of a command. In this state, the variable "curcmd" (Current Command) has its top bit ("cmdemp") set, meaning that it is empty. When a byte is received from the CPU in this state, this routine jumps to the label "firstc". The byte is placed in the "curcmd" byte (clearing the top bit), and then a multi-way branch (jidw) is performed, whose destination depends on the contents of the byte. The possible destinations have labels starting with the letters "fc". If the command has only one byte (for example, the command BEEP), it is processed immediately in the "fc" sequence, and the "curcmd" variable is set empty again. If, however, the command is longer than one byte, its "fc" routine will place a value into the variable "numexp", which gives the number of additional bytes that are expected for this command, and then will return from the interrupt. Note that the "curcmd" byte now appears to be full, because its top bit is no longer set.
2. Collecting bytes of a command. The code that is relevant in this state is between the labels "upiwr" and "lastc". This state is in effect while the "cmdemp" bit of "curcmd" is zero and the "numexp" variable is non-zero. Each I3 interrupt causes the routine to place the command byte into a buffer ("cpubuf"), with pointer variable "cpud", decrement the "numexp" variable, and return if the result is non-zero. If the result is zero, then the routine has collected an entire command, and it goes to the label "lastc", and enters state (3) below.

3. In this state, the requested number of bytes has been collected, and this usually means that the entire command, except for the first byte, is in the "cpubuf" area of RAM. The code for this state is at label "lastc". First, the "curcmd" byte is checked to see whether "extended collection" is being performed (bit 6 set: see below). If not, the "curcmd" byte is set empty. A multiway branch is then performed (jidw), which transfers control depending on the command byte in "curcmd". All routines that are destinations of this branch start with the letters "lc". The "lc" routine for each command uses the data in "cpubuf" to process the current command. In some cases, this processing is completed very quickly. For example, at label "lcsled", a value is simply transferred from "cpubuf" to a latch that drives the LEDs on the front panel, and this interrupt service routine returns. But a more complex command can move data out of "cpubuf" to other variables in RAM, and start a timer to sequence the process of executing the command.

In some commands (for example, SEND-LCD), state (3) above is entered twice. This is called "extended collection", and occurs when a command has variable length. State (3) is entered once to collect enough information to determine the exact length of the command. It then sets up the "numexp" variable again, re-entering state (2) to collect the remainder of the command. When state (3) is entered the second time, it processes the command. A bit in the "curcmd" variable (bit 6, called "getcnt") is set in state (1), which indicates that another collection will be performed, and prevents state (3) from setting the "curcmd" byte empty the first time it is entered.

Command Processing Routines

INITIALIZE	I3 interrupt labels:	State 1 = fcinit	State 3 = lcinitt
SET-CONTRAST	I3 interrupt labels: At label "lcsclv" (Set LCD Voltage), the LCD Contrast latch is loaded from the value supplied by the CPU.	State 1 = fcsclv	State 3 = lcsclv
SEND-LCD	I3 interrupt labels: This command uses the "extended collection" feature. At label "fcsclcd", two bytes are requested for collection, but the "getcnt" bit of "curcmd" is set, meaning that these are not the last bytes of the command. At label "lcsclcd" (jumping to label "lcscl1"), the length of the instruction is determined from the # bytes value supplied by the CPU, and a second collection of bytes is requested, this time with the "getcnt" bit off. When the last byte has been collected, control is transferred to the label "lcsclcd", then to "lcscl2". Here, the data bytes for the panel are unloaded from the CPU buffer area "cpubuf" into the LCD string buffer "lcsdbuf". The flag (RS) bits are loaded into variable "lcsdsg", and the number of bytes to be sent to the LCD display is placed into variable "lcsdsc". Timer T6 is now started, to provide scheduling interrupts for writing the bytes from the LCD string buffer to the LCD display.	State 1 = fcsclcd	State 3 = lcsclcd

On occurrence of each T6 interrupt (labels "t6int" and "t6nxt"), one byte is written to the LCD display. Depending on the state of the RS flag for that byte, and the value sent to the panel, T6 may run for either 120 μ s or 4900 μ s before it triggers the next transfer. When the last character has been transferred, and Timer T6 has provided the proper delay after it, the bit "alcdak" is set in the "alert" word, requesting the main program to send an IACK-SEND-LCD interrupt to the CPU.

SEND-LED	I3 interrupt labels: At label "lcsled", the byte provided by the CPU is written to the LED latch.	State 1 = fcsled State 3 = lcsled
BEEP	I3 interrupt labels: At label "fcbEEP", Port P pin P3 is enabled to toggle on each underflow of Timer T7, which has been initialized at the beginning of the program (label "stmrS") to underflow at a rate of 6 kHz. Pin P3, then, presents a 3 kHz square wave to the panel buzzer. To time out the duration of the beep tone, interrupts from Timer T0 are enabled, which then occur once every 53 ms. The variable "beepct" is set up with the number of T0 interrupts to accept, and is decremented on each T0 interrupt. When it has been decremented to zero (meaning that one second has elapsed), pin P3 is reset to a constant zero to turn off the tone.	State 1 = fcbEEP State 3 = (none)

4.1.4.2 Background Timer (T1) Task

The Timer T1 interrupt service routine represents a task that is not triggered directly by CPU commands. Its functions are to interrupt the CPU periodically for the Real-Time Clock function, and to present the !BUTTON-DATA interrupt whenever the pushbutton inputs change state.

Timer T1 is loaded with a constant interval value which is used to interrupt the HPC at 10 ms intervals. When the Timer T1 interrupt occurs (labels 'tmrint', to 't1poll', to 't1int'), then if the real-time interrupt is enabled, the variable "rtcnt" is decremented to determine whether an !RTC interrupt should be issued to the CPU. If so, the bit "artc" in the "alert" word is set, requesting the main program to issue the interrupt. The main program, at label "sndrtc", actually interrupts the CPU. No other data is passed to the CPU with the interrupt.

At label "kbdchk" the panel pushbutton switches are also sampled. If the pattern matches the last sample taken (saved in variable "swlast") then it is considered to be sta-

ble, and it is then compared to the last switch pattern sent to the CPU (in variable "swlsnt"). If the new pattern differs, then it is placed in "swlsnt", and the bit "abutton" in variable "alert" is set, requesting the main program to send a !BUTTON-DATA interrupt. The main program, at label "sndbtn", triggers the interrupt and passes the new pattern to the CPU from variable "swlsnt".

4.1.4.3 Timer T6 Interrupt

Because the LCD controller's command acknowledgement capability was not used in our application, Timer T6 is used to time out the LCD controller's processing times. See the description of the SEND-LCD command above.

4.1.4.4 Timer T0 Interrupt

The interrupt service routine for Timer T0 (labels "tmrint", to "t0poll", to "t0int") is used simply to provide timing for the duration of the speaker tone. The interrupt is enabled in response to the BEEP command from the CPU, and is disabled on occurrence of the interrupt. It provides an interval of approximately one second.

4.2 HPC Firmware Listing

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)

HPCUPI

25-Feb-88 10:05
PAGE 1

```

1
2
3          .title  HPCUPI,'UPI PORT INTERFACE DEMO'
4          ;
5          ; Demo program for HPC46083 UPI Port:
6          ; Demonstrates use of the HPC as an interface
7          ; between an NS32016 CPU and some typical
8          ; front-panel types of devices:
9          ;     LED indicators (up to 8)
10         ;     Pushbuttons (up to 8)
11         ;     LCD alphanumeric display controller (Hitachi HD44780)
12         ;     Speaker for error beeps
13         ; Also generates Real-Time Clock interrupts at a
14         ;     selectable rate.
15         ;
16         ; Generates IDIAG interrupt on errors;
17         ;     severity code of NOTE (e.g. real-time event lost),
18         ;     or FATAL (e.g. bad command).
19         ; Recovery from fatal errors provided by RESET command.
20

```

TL/DD/9976-17

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Declarations: Register Addresses

HPCUPI

25-Feb-88 10:05
PAGE 2

```

21                                     .form 'Declarations: Register Addresses'
22
23 00C0      psW      =      x'00:w ; PSW register
24 00C8      al      =      x'08:b ; Low byte of Accumulator.
25 00C9      ah      =      x'09:b ; High byte of Accumulator.
26 00CC      bl      =      x'0C:b ; Low byte of Register B.
27 00CD      bh      =      x'0D:b ; High byte of Register B.
28 00CE      xl      =      x'0E:b ; Low byte of Register X.
29 00CF      xh      =      x'0F:b ; High byte of Register X.
30
31 00D0      enr      =      x'00:b
32 00D2      irpd     =      x'D2:b
33 00D4      ircd     =      x'D4:b
34 00D6      sio      =      x'D6:b
35 00D8      porti   =      x'D8:b
36 00E0      obuf     =      x'E0:b ; (Low byte of PORTA.)
37 00E1      portah  =      x'E1:b ; High byte of PORTA.
38 00E2      portb   =      x'E2:w
39 00E3      portbl  =      x'E3:b ; Low byte of PORTB.
40 00E4      portbh  =      x'E4:b ; High byte of PORTB.
41 00E6      upic    =      x'E6:b
42 00F0      ibuf    =      x'F0:b ; (Low byte of DIRA.)
43 00F1      dirah   =      x'F1:b ; High byte of DIRA.
44 00F2      dirb   =      x'F2:w
45 00F3      dirbl   =      x'F3:b ; Low byte of DIRB.
46 00F4      dirbh  =      x'F4:b ; High byte of DIRB.
47 00F5      bfun    =      x'F5:w
48 00F6      bfunt   =      x'F6:b ; Low byte of BFUN.
49 00F7      bfunh   =      x'F7:b ; High byte of BFUN.
50
51 0104      portd   =      x'0104:b
52 0120      enu      =      x'0120:b
53 0122      enui    =      x'0122:b
54 0124      rbuf    =      x'0124:b
55 0126      tbuf    =      x'0126:b
56 0128      enur    =      x'0128:b
57
58 0140      t4      =      x'0140:w
59 0142      r4      =      x'0142:w
60 0144      t5      =      x'0144:w
61 0146      r5      =      x'0146:w
62 0148      t6      =      x'0148:w
63 014A      r6      =      x'014A:w
64 014C      t7      =      x'014C:w
65 014E      r7      =      x'014E:w
66 0150      pmode   =      x'0150:w
67 0151      pmdl    =      x'0151:b ; Low byte of PMODE.
68 0152      pmdh    =      x'0152:b ; High byte of PMODE.
69 0153      portp   =      x'0153:w
70 0154      portpl  =      x'0154:b ; Low byte of PORTP.

```

TL/DD/9976-18

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Declarations: Register Addresses

HPCUPI

25-Feb-88 10:05
PAGE 3

```

71 0153      portph  =      x'0153:b ; High byte of PORTP.
72 015C      eicon   =      x'015C:b
73
74 0182      t1      =      x'0182:w
75 0184      r1      =      x'0184:w
76 0186      r2      =      x'0186:w
77 0188      t2      =      x'0188:w
78 018A      r3      =      x'018A:w
79 018C      t3      =      x'018C:w
80 018E      divby   =      x'018E:w
81 018F      divbyl  =      x'018F:b ; Low byte of DIVBY.
82 0190      divbyh  =      x'0190:b ; High byte of DIVBY.
83 0191      tmode   =      x'0191:w
84 0192      tmdl    =      x'0192:b ; Low byte of TMMODE.
85 0193      tmdh    =      x'0193:b ; High byte of TMMODE.
86 0194      tcon    =      x'0194:w
87
88

```

TL/DD/9976-19

```

      .form 'Declarations: Register Bit Positions'
      ; Name      Position      Register(s)
      ; -----
      89
      90
      91
      92
      93
      94 0000      gie      =      0      ; enir
      95 0002      i2      =      2      ; enir, irpd, ircd
      96 0003      i3      =      3      ; enir, irpd, ircd
      97 0004      i4      =      4      ; enir, irpd, ircd
      98 0005      tmrs     =      5      ; enir, irpd
      99 0006      uart     =      6      ; enir, irpd
     100 0007      ei       =      7      ; enir, irpd
     101
     102 0001      umode    =      1      ; ircd
     103 0000      uedone   =      0      ; irpd
     104
     105 0000      tbmt     =      0      ; enu
     106 0001      rbfl     =      1      ; enu
     107 0004      b8or9    =      4      ; enu
     108 0005      xbit9    =      5      ; enu
     109 0002      wakeup   =      2      ; enur
     110 0003      rbit9    =      3      ; enur
     111 0006      frmerr   =      6      ; enur
     112 0007      doeerr   =      7      ; enur
     113 0000      eti       =      0      ; enui
     114 0001      eri       =      1      ; enui
     115 0002      xtclk    =      2      ; enui
     116 0003      xrclk    =      3      ; enui
     117 0007      b2stp    =      7      ; enui
     118
     119 0000      wrrdy    =      0      ; upic
     120 0001      rdrdy    =      1      ; upic
     121 0002      la0      =      2      ; upic
     122 0003      upien    =      3      ; upic
     123 0004      b8or16   =      4      ; upic
     124
     125 0000      t0tie    =      0      ; tmdl
     126 0001      t0pnd    =      1      ; tmdl
     127 0003      t0ack    =      3      ; tmdl
     128 0004      t1tie    =      4      ; tmdl
     129 0005      t1pnd    =      5      ; tmdl
     130 0006      t1stp    =      6      ; tmdl
     131 0007      t1ack    =      7      ; tmdl
     132 0000      t2tie    =      0      ; tmdh
     133 0001      t2pnd    =      1      ; tmdh
     134 0002      t2stp    =      2      ; tmdh
     135 0003      t2ack    =      3      ; tmdh
     136 0004      t3tie    =      4      ; tmdh
     137 0005      t3pnd    =      5      ; tmdh
     138 0006      t3stp    =      6      ; tmdh
  
```

TL/DD/9976-20

Declarations: Register Bit Positions

```

139 0007      t3ack = 7      ; tmndh
140
141 0000      t4tie = 0      ; pwmcl
142 0001      t4pnd = 1      ; pwmcl
143 0002      t4stp = 2      ; pwmcl
144 0003      t4ack = 3      ; pwmcl
145 0004      t5tie = 4      ; pwmcl
146 0005      t5pnd = 5      ; pwmcl
147 0006      t5stp = 6      ; pwmcl
148 0007      t5ack = 7      ; pwmcl
149 0000      t6tie = 0      ; pwmch
150 0001      t6pnd = 1      ; pwmch
151 0002      t6stp = 2      ; pwmch
152 0003      t6ack = 3      ; pwmch
153 0004      t7tie = 4      ; pwmch
154 0005      t7pnd = 5      ; pwmch
155 0006      t7stp = 6      ; pwmch
156 0007      t7ack = 7      ; pwmch
157
158 0000      t4out = 0      ; portpl
159 0003      t4tfn = 3      ; portpl
160 0004      t5out = 4      ; portpl
161 0007      t5tfn = 7      ; portpl
162 0000      t6out = 0      ; portph
163 0003      t6tfn = 3      ; portph
164 0004      t7out = 4      ; portph
165 0007      t7tfn = 7      ; portph
166
167 0000      eiopl = 0      ; eicon
168 0001      eimode = 1     ; eicon
169 0002      eiack = 2     ; eicon
170
171 0005      so    = 5      ; portbl, dirbl, bfunl
172 0006      sk    = 6      ; portbl, dirbl, bfunl
173 0007      pntclk = 7     ; portbl, dirbl
174
175 0001      levclk = 1     ; portbh, dirbh
176          ; ua0 would be 2 , but requires no setup.
177 0003      uwrrdy = 3     ; portbh, dirbh, bfunh
178 0004      cdata = 4     ; portbh (enables non-pushbutton data to Port D).
179 0005      astts = 5     ; portbh (enables pushbutton data to Port D).
180 0006      ledclk = 6     ; portbh, dirbh
181 0007      undrdy = 7     ; portbh, dirbh, bfunh
182
183          ;          CONSTANTS
184          ;
185 0011      xon=  x'11    ; XON character: Control-Q
186 0013      xoff= x'13   ; XOFF character: Control-S
187
188

```

```

189 .form 'Space Declarations'
190 .sect DSECT,BASE,REL ; Basepage RAM variables (addresses 0000-00BF)
191
192 ; WORD-ALIGNED
193 dummy: .dsw 1 ; x'00,01 ; Destroyed on reset (address 0).
194 .set upicsv,dummy ; Temporary image of UPIC register.
195 alert: .dsw 1 ; Alert status bits to main program:
196 ; generate interrupts to CPU.
197 .set alerth,alert+1:b ; Declare top byte of ALERT word.
198 cpued: .dsw 1 ; Current address within CPU command buffer.
199 cpubuf: .dsw 4 ; Buffer for accepting command parameters from CPU.
200 lcdsix: .dsw 1 ; Pointer into LCD character string buffer.
201
202 ;BYTE-ALIGNED
203 curcmd: .dsb 1 ; Current command byte from CPU being processed.
204 numexp: .dsb 1 ; Number of parameter bytes expected before command processing
205 ; begins.
206 lcvs: .dsb 1 ; Image of LCD Voltage (Contrast) latch setting; needed with
207 ; LCD RS (PAUX0) signal coming from this latch.
208 lcdfsg: .dsb 1 ; Holds flag bits for characters sent to Panel LCD display.
209 lcdnum: .dsb 1 ; Number of characters to be sent to LCD display.
210 lcdfsg: .dsb 1 ; Flag bits associated with characters in LCD String Buffer.
211 lcdsct: .dsb 1 ; Counter for characters being sent to LCD display from String
212 ; Buffer.
213 swlast: .dsb 1 ; Last-sampled switch values.
214 swlsnt: .dsb 1 ; Last switch values sent to CPU.
215 beepct: .dsb 1 ; Beep duration count. Counts occurrences of T0 interrupt.
216 rtcivl: .dsb 1 ; Real-Time Clock Interval (units of 10 milliseconds).
217 rtcctn: .dsb 1 ; Real-Time Clock Current Count (units of 10 milliseconds).
218 rtev: .dsb 1 ; Events to check for on Timer T1 interrupts.
219 dsevc: .dsb 1 ; Diagnostic Interrupt: Severity Code.
220 derrc: .dsb 1 ; Diagnostic Interrupt: Error Code.
221 dbyte: .dsb 1 ; Diagnostic Interrupt: Error Byte.
222 dcmd: .dsb 1 ; Diagnostic Interrupt: Current Command.
223 dqual: .dsb 1 ; Diagnostic Interrupt: Qualifier (Command Status).
224
225
226 ; BIT POSITIONS
227
228 ; ALERT status word (low-order byte) bits:
229
230
231 abutton = 0 ; Pushbutton switch state change.
232 artc = 1 ; Real-Time Interrupt detected.
233 adiaq = 2 ; Diagnostic interrupt.
234 alcdak = 3 ; LCD Panel Write Acknowledge.
235 ; (Other bits not defined.)
236
237 ; ALERT status word (high-order byte, named alerth) bits:
238

```

TL/DD/9976-22

```

239 ; (Other bits not defined.)
240
241 ; CURCMD byte: Current CPU command. The lower 5 bits contain the
242 ; command code. The upper two bits contain
243 ; further information about command collection:
244 ;
245 cmdemp= 7 ; Bit 7 (MSB) of curcmd = 1 means that no command is being
246 ; processed and curcmd byte is "empty".
247 getcnt= 6 ; Bit 6 of curcmd = 1 means that the count is being received
248 ; for a variable-length command.
249
250 ; LCVS byte: LCD Voltage (Contrast) Latch memory image.
251 ; Contains voltage value in its least-significant 3 bits,
252 ; RS signal to LCD controller in bit 3, and debugging
253 ; information in its top 4 bits.
254 pntrs= 3 ; Bit 3 is (inverted) RS signal to panel.
255
256 ; RTEVS byte: Events to check for at 10-millisecond intervals.
257 ; (T1 Underflows)
258 ;
259 rtcenb= 0 ; 1 = Real-Time Clock interrupts enabled to CPU.
260
261
262 .sect STACK,RAM16,REL ; On-chip RAM in addresses 01C0-01FF.
263 stackb: .dsw 16 ; Space for 8 words beyond
264 ; interrupt context.
265 avail: .dsw 12 ; Spare portion of this space.
266 lcdbuf: .dsw 4 ; LCD String Buffer.
267

```

TL/DD/9976-23


```

268                                     .form 'Code Section'
269 0000 .sect CSECT,ROM16,REL ; Code space. (On-chip ROM)
270
271                                     ; Declarations of subroutines called by one-byte JSRP instruction.
272
273 0000 .spt rdwait ; Waits for CPU to read a value from UPI port.
274 0000 .spt wrpnl ; Writes to LCD panel (for initialization only).
275
276                                     ; Program starts at label "start" on reset. This routine is the fatal
277                                     ; error handler, located here for convenience in setting breakpoint.
278
279 0000 960018 hgupi: rbit gie,enir ; Fatal error: signal it and halt.
280 0003 96120F R sbit 7,lcvs ; Signal error on most-significant bit of
281                                     ; LCD Contrast Latch.
282 0006 961208 R sbit pn1rs,lcvs ; Select command mode for LCD controller.
283 0009 8C12E1 R ld portah,lcvs ; Place error on Port A for latch.
284 000C 96E309 R sbit lcvc1k,portbh ; Clock LCD Contrast Latch high,
285 000F 96E319 R rbit lcvc1k,portbh ; then low to load it.
286 0012 8601510A R sbit t6stp,pwmdh ;
287 0016 86015118 R rbit t6tie,pwmdh ; Set up Timer T6 for non-interrupt use.
288 001A 40 nop
289 001B 86015119 R rbit t6pnd,pwmdh ; Clear Pending bit.
290 001F 3F00 pop 0.w ; Get error address from stack.
291 0021 870000C4 R ld sp,#stackb ; In case of stack underflow, re-initialize SP.
292 0025 9001 ld A,#x'01
293 0027 2E R jsrl wrpnl ; Clear LCD panel.
294 0028 961218 R rbit pn1rs,lcvs ; Set up panel for data.
295 002B 8C12E1 R ld portah,lcvs ; Place error on Port A for latch.
296 002E 96E309 R sbit lcvc1k,portbh ; Clock LCD Contrast Latch high,
297 0031 96E319 R rbit lcvc1k,portbh ; then low to load it.
298 0034 8801 ld A,1.b ; Process first character of return address.
299 0036 3B swap A
300 0037 990F and A,#x'0F
301 0039 A6007AC888 R ld A,hextab[A].b
302 003E 2E R jsrl wrpnl ; Display it on LCD panel.
303 003F 8801 ld A,1.b ; Process second character of return address.
304 0041 990F and A,#x'0F
305 0043 A6007AC888 R ld A,hextab[A].b
306 0048 2E R jsrl wrpnl ; Display it on LCD panel.
307 0049 8800 ld A,0.b ; Process third character of return address.
308 004B 3B swap A
309 004C 990F and A,#x'0F
310 004E A6007AC888 R ld A,hextab[A].b
311 0053 2E R jsrl wrpnl ; Display it on LCD panel.
312 0054 8800 ld A,0.b ; Process last character of return address.
313 0056 990F and A,#x'0F
314 0058 A6007AC888 R ld A,hextab[A].b
315 005D 2E R jsrl wrpnl ; Display it on LCD panel.
316
317 005E 96E611 hgupi: ifbit rdrdy,upic ; Check to see if 08UF register is full.

```

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Code Section

HPCUPI

25-Feb-88 10:05
PAGE 9

```

318 0061 971DE0          ld      obuf,#vdiag      ; If not, fill it with IDIAG vector
319                                ; continuously.
320 0064 96D213          ifbit  i3,irpd          ; Check for UPI data ready.
321 0067 41              jp      hgup1
322 0068 6A              jp      hgup1
323
324 0069 B2A5F00C        hgup1: ifeq  ibuf,#x'A5    ; Check for RESET command.
325 006D 41              jp      hgrst
326 006E 47              jp      hgup2
327 006F 96E612        hgrst: ifbit  la0,upic
328 0072 43              jp      hgup2
329 0073 B4027A          jmpl   xreset          ; If so, then go reset the HPC.
330
331                                ; This is part of the outer loop, waiting for
332                                ; the RESET command.
333 0076 97F7D2          hgup2: ld      irpd,#x'F7    ; Clear the UMR detector.
334 0079 7B              jp      hgup1          ; and keep looking. This is an
335                                ; infinite loop until RESET is seen.
336
337 007A 30              hextab: .byte  '0','1','2','3','4','5','6','7'
338                                .byte  '8','9','A','B','C','D','E','F'
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389

```

TL/DD/9976-25

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Hardware Initialization

HPCUPI

25-Feb-88 10:05
PAGE 10

```

340                                .form  'Hardware Initialization'
341
342 00BA 97B0C0          start: ld      psw.b,#x'0B    ; Set one WAIT state.
343
344 00BD              srfsh:                ; Start dynamic RAM refreshing,
345                                ; as quickly as possible.
346 00BD B601520B        sbit  t4out,portpl      ; Trigger first refresh
347                                ; immediately.
348 0091 B601500A        sbit  t4stp,pwmdl       ; Stop timer T4 to
349                                ; allow loading,
350                                ; then load it.
351 0095 8300140AB        ld      t4.w,#8          ; Start timer T4.
352 009A B601501A        rbit  t4stp,pwmdl       ; Start timer T4.
353 009E B601520B        sbit  t4tfn,portpl     ; Enable pulses out.
354 00A2 8300142AB        ld      r4.w,#8          ; Load R4.
355
356 00A7              supi:                ; Set up UPI port.
357 00A7 9718E6          ld      upic,#x'1B       ; 8-Bit UPI Mode
358                                ; enabled.
359
360 00AA 96F50B          sbit  uwrrdy,bfunh      ; Enable UWRRDY/ out.
361 00AD 96F30B          sbit  uwrrdy,dirbh
362 00B0 B8F0            ld      A,ibuf           ; Empty IBUF register,
363                                ; in case of false trigger.
364
365 00B2 96F50F          sbit  urdrdy,bfunh      ; Enable URDRDY/ out.
366 00B5 96F30F          sbit  urdrdy,dirbh
367
368 00BB 96D40A          sbit  i2,ircd           ; Set up UREAD/ interrupt.
369 00BB 97FB02          ld      irpd,#x'FB       ; Detects rising edges.
370                                ; Clear any false interrupt
371                                ; due to mode change.
372
373 00BE 96D40B          sbit  i3,ircd           ; Set up UWRITE/ interrupt.
374 00C1 97F7D2          ld      irpd,#x'F7       ; Detects rising edges.
375                                ; Clear any false interrupt
376                                ; due to mode change.
377
378 00C4              sram:                ; Clear all RAM locations.
379                                ; Clear Basepage bank:
380 00C4 2D00BE          ld      BK,#x'0000,#x'00BE ; Establish loop base and limit.
381 00C7 00              sram1: clr  A
382 00C8 E1              xs      A,[B+].w
383 00C9 62              jp      sram1
384
385 00CA A701C001FE        ld      BK,#x'01C0,#x'01FE ; Clear Non-Basepage bank:
386 00CF 00              sram2: clr  A
387 00D0 E1              xs      A,[B+].w
388 00D1 62              jp      sram2
389

```

5

TL/DD/9976-26

```

390 00D2                sskint:                ; Set up Stack and remove
391                    ; individual interrupt enables.
392 00D2 870020C4      R      ld      sp,#stackb+2 ; Move stack to high
393                    ; bank of on-chip RAM.
394 00D6 8700000000AB  R      ld      stackb.w,#hangup ; Safeguard against
395                    ; stack underflow.
396 00DC 9700D0                ld      enir,#x'00 ; Disable interrupts
397                    ; individually.
398
399 00DF 83000192B8      tminit: ld      t0con,#x'08
400 00E4 874400190AB    ld      tmode,#x'4440 ; Stop timers T1, T2, T3.
401 00EA 8355018EAB    ld      divby,#x'0055 ; Timers T2 and T3 set to
402                    ; clock externally.
403 00EF 87CCCB0190AB  ld      tmode,#x'CCCB ; Clear and disable timer
404                    ; T0-T3 interrupts.
405
406 00F5 97FFF1                ld      dirah,#x'FF ; Initialize Port A upper byte for output.
407 00FB 96E30D                sbit   astts,portbh ; Enable and de-assert ENASTTS/ signal
408 00FB 96F30D                sbit   astts,dirbh  ; (enables pushbutton data to Port D).
409 00FE 96E30C                sbit   cdata,portbh ; Enable and de-assert ENCDATA/ signal.
410 0101 96F30C                sbit   cdata,dirbh  ; (enables other data to Port D).
411
412 0104 97FFE1                sled:  ld      portah,#x'FF ; Set up to turn off LED's.
413 0107 96E31E                rbit   ledclk,portbh ; Start with LEDCLK low,
414 010A 96F30E                sbit   ledclk,dirbh ; (enable output),
415 010D 96E30E                sbit   ledclk,portbh ; then high,
416 0110 96E31E                rbit   ledclk,portbh ; then low again.
417
418 0113                    stmrs:                ; Set up remaining timers.
419                    ; (T1-T3 already stopped
420                    ; and pending bits cleared
421                    ; at tminit above, as
422                    ; part of MICROWIRE init.)
423
424 0113 872FFF0182AB    ld      t1,#12287 ; T1 runs at 10-millisecond real-time interval.
425 0119 872FFF0184AB    ld      r1,#12287
426
427                    ; Timer remains stopped, and interrupt
428                    ; disabled, until INITIALIZE command.
429 011F 8744400150AB    ld      pmode,#x'4440 ; Stop timers T4-T7.
430 0125 40                nop ; Wait for valid PND
431 0126 40                nop ; bits.
432 0127 87CCCB0150AB    ld      pmode,#x'CCCB ; Clear and disable
433                    ; interrupts from all
434                    ; PWM timers.
435
436 012D 87FFFF014AAB    ld      r6,#x'FFFF ; No modulus for LCD Display Ready timer.
437
438 0133 83CC014CAB    ld      t7,#204 ; Set T7 to underflow at 6 KHz rate
439 0138 83CC014EAB    ld      r7,#204 ; (= 3 KHz at pin).

```

TL/DD/9976-27

```

440 013D B601531F      rbit   t7tfn,portph   ; Disable beep tone to panel speaker.
441 0141 B601511E      rbit   t7stp,pwmdh   ; Start T7 running.
442
443
444 0145                slcd:                ; Set up LCD display.
445                    ; Requires use of timer T6, so
446                    ; appears after timer initialization.
447
448                    ; First, set up LCD contrast.
449 0145 970A12        R    ld    lcvs,#x'0A   ; Initialize memory image of LCD Voltage
450                    ; latch, containing RS (PAUX0) bit also.
451 0148 8C12E1        R    ld    portah,lcvs  ; Arbitrary initial contrast level of 5,
452                    ; and RS/ (PAUX0/) is high (= "command").
453 0148 96E319        rbit   lcvcclk,portbh ; Start with LVCVCLK low,
454 014E 96F309        sbit   lcvcclk,dirbh  ; (enable output)
455 0151 96E309        sbit   lcvcclk,portbh ; then high,
456 0154 96E319        rbit   lcvcclk,portbh ; then low to get it into LCV latch.
457
458                    ; Initialize PNLCLK (Panel "E" signal).
459 0157 96E20F        sbit   pnlclk,portbl  ; Start with PNLCLK high
460 015A 96F20F        sbit   pnlclk,dirbl   ; (enable output).
461
462                    ; Wait for worst-case command
463                    ; execution time (4.9 ms, twice), in case
464                    ; a panel command was triggered while
465                    ; PNLCLK was floating.
466 015D B6015108        sbit   t6ack,pwmdh   ; Clear T6 PND bit.
467 0161 8732C0148AB    ld     t6,#13000     ; Set T6 to twice 4.9 milliseconds.
468 0167 B601511A        rbit   t6stp,pwmdh   ; Start timer T6.
469 0168 B6015111        lcdlp1: ifbit   t6pnd,pwmdh ; Wait for T6 PND bit
470                    ; to be set.
471 016F 41            jp     lcdgo1
472 0170 65            jp     lcdlp1
473 0171 B601510A        lcdgo1: sbit   t6stp,pwmdh ; Stop timer T6.
474 0175 B6015108        sbit   t6ack,pwmdh   ; Clear T6 PND bit.
475
476                    ; Reset Panel controller (per Hitachi HD44780
477                    ; User's Manual).
478
479                    ; (Panel RS signal was set
480                    ; in LCD Contrast initialization above,
481                    ; so no change needed here to
482                    ; flag these as commands.)
483
484 0179 9038            ld     A,#x'38       ; Send "8-Bit Mode, 2 Lines" command: one;
485 017B 2E            R    jsrl  wrpnl     ;
486 017C 9038            ld     A,#x'38       ; two;
487 017E 2E            R    jsrl  wrpnl     ;
488 017F 9038            ld     A,#x'38       ; three;
489 0181 2E            R    jsrl  wrpnl     ;

```

TL/DD/9976-28

```

490 0182 9038            ld     A,#x'38       ; four times.
491 0184 2E            R    jsrl  wrpnl     ;
492 0185 9008            ld     A,#x'08       ; Disable display.
493 0187 2E            R    jsrl  wrpnl     ;
494 0188 9001            ld     A,#x'01       ; Clear display RAM.
495 018A 2E            R    jsrl  wrpnl     ;
496
497                    ; Initial default mode settings.
498
499 018B 9006            ld     A,#x'06       ; Set mode to move cursor to the right, no
500 018D 2E            R    jsrl  wrpnl     ; automatic shifting of display.
501 018E 900E            ld     A,#x'0E       ; Enable display: non-blinking cursor mode.
502 0190 2E            R    jsrl  wrpnl     ;
503
504
505                    ; CONTINUES TO MAIN PROGRAM INITIALIZATION

```

TL/DD/9976-29

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Main Program Initialization

HPCUPI

25-Feb-88 10:05
PAGE 14

```

506                .form 'Main Program Initialization'
507
508
509 0191            minit:
510                                ; Once-only initializations.
511
512 0191 97010      R      ld      curcmd,#x'00  ; Current Command: top bit set means "none".
513 0194 87000604  R      ld      cpud,#cpubuf  ; Set CPU command index to beginning of buffer.
514 0198 97011      R      ld      numexp,#8   ; Arbitrary starting value.
515
516                                ; Arbitrary set of initialization values for variables,
517                                ; in effect until receipt of the first INITIALIZE
518                                ; command.
519
520 0198 87000002  R      ld      alert,#0     ; No events pending.
521
522 019F            runsys:
523                                ; Enable interrupts, start timers and go to main loop.
524 019F 960000      sbit    tmrs,enir        ; Enable timer interrupts. (Done here
525                                ; to allow certain commands without an
526                                ; INITIALIZE command first.)
527 01A2 960000      sbit    i3,enir         ; Enable CPU Command interrupt.
528 01A5 960000      sbit    gie,enir       ; Enable interrupt system.
529
530
531

```

TL/DD/9976-30

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Main Scan Loop

HPCUPI

25-Feb-88 10:05
PAGE 15

```

531                .form 'Main Scan Loop'
532
533                ; Declarations
534
535 0011            vrtc =      x'11  ; Real-Time Clock vector number.
536 0017            vldak =     x'17  ; Acknowledge finished writing to LCD panel.
537 0018            vbutton =   x'18  ; Pushbutton status change: a button pressed or
538                                ; released.
539 001D            vdiag =     x'1D  ; Diagnostic Interrupt.
540
541
542                                ; Error Vectors for unimplemented or
543                                ; unexpected interrupts.
544
545                ; level 0 is Reset, provided by assembler.
546 FFFC 0000      R      .ipt  1,hangup      ; NMI: never expected.
547 FFFA 0000      R      .ipt  2,hangup      ; UPI READ READY: never expected.
548 FFF6 0000      R      .ipt  4,hangup      ; I4 Interrupt Vector: never expected.
549 FFF2 0000      R      .ipt  6,hangup      ; UART Interrupt Vector: never expected.
550 FFF0 0000      R      .ipt  7,hangup      ; EI Interrupt Vector: never expected.
551
552 01A8            mainlp:
553
554
555 01A8 820002FC  R      chkalt: ifeq  alert,w,#x'00  ; Check for alert conditions.
556 01AC 64                jp      chkalt          ; If none, keep looping.
557
558 01AD 960211      R      ifbit  rtc>alert.b      ; Check for RTC interrupt request.
559 01B0 3010                jsrl  sndrtc          ; If so, then send Real-Time Clock interrupt.
560
561 01B2 960213      R      ifbit  alcdak>alert.b   ; Check for LCD Panel write done.
562 01B5 3013                jsrl  sndlak          ; If so, then send LCD Acknowledge interrupt.
563
564 01B7 960210      R      ifbit  abutton>alert.b  ; Check for a pushbutton change.
565 01BA 3016                jsrl  sndbtn          ; If so, then report the change to the CPU.
566
567 01BC 960212      R      ifbit  adiag>alert.b   ; Check for Diagnostic Interrupt.
568 01BF 3023                jsrl  sndiag          ; If so, then send interrupt and data.
569
570 01C1 79                jmpl  chkalt          ; No "responses" defined yet; just close loop.
571

```

TL/DD/9976-31

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Main: Send Real-Time Clock Interrupt

HPCUPI

25-Feb-88 10:05
PAGE 16

```

572                .form 'Main: Send Real-Time Clock Interrupt'
573
574                ; No data transfer; just trigger interrupt and continue.
575
576 01C2            sndrtc:
577 01C2 960219      R      rbit   rtc>alert.b      ; Clear ALERT bit.
578 01C5 2F                R      jsrl  rdwait        ; Check that UPI interface is ready.
579                                ; If not, loop until it is.
580
581 01C6 9711E0      ld      obuf,#vrtc        ; Load Real-Time Clock vector into OBUF for CPU.
582 01C9 3C                ret                    ; Return to main loop.
583

```

TL/DD/9976-32

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO

HPCUPI

25-Feb-88 10:05
PAGE 17

Main: Send LCD Write Acknowledge Interrupt

```

584                .form 'Main: Send LCD Write Acknowledge Interrupt'
585                ; No data transfer; just trigger interrupt and continue.
586
587
588 sndlak:
589 01CA 96021B      R    rbit   alcdak,alert.b ; Clear ALERT bit.
590 01CD 2F          R    jsrl   rdwait ; Check that UPI interface is ready.
591                ; If not, loop until it is.
592
593 01CE 9717E0      ld     obuf,#vldcak ; Load LCD-Acknowledge vector into OBUF for CPU.
594 01D1 3C          ret     ; Return to main loop.
595

```

TL/DD/9976-33

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO

HPCUPI

25-Feb-88 10:05
PAGE 18

Main: Send Pushbutton Status to CPU

```

596                .form 'Main: Send Pushbutton Status to CPU'
597
598 01D2                sndbtn:
599 01D2 2F          R    jsrl   rdwait ; Check that UPI interface is ready.
600                ; If not, loop until it is.
601
602 01D3 9718E0      ld     obuf,#vbutton ; Load BUTTON-DATA vector into OBUF for CPU.
603
604 01D6 2F          R    jsrl   rdwait ; Check that UPI interface is ready.
605                ; If not, loop until it is.
606
607 01D7 960018      R    rbit   gie,enir ; *** Begin Indivisible Sequence ***
608 01DA 8C18E0      R    ld     obuf,swlsnt ; Load Pushbutton Data Byte into OBUF for CPU.
609 01DD 96021B      R    rbit   abutton,alert.b ; Clear ALERT bit.
610 01E0 960008      R    sbit   gie,enir ; *** End Indivisible Sequence ***
611 01E3 3C          ret     ; Return to main loop.
612

```

TL/DD/9976-34

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Main: Send Diagnostic Interrupt to CPU

HPCUPI

25-Feb-88 10:05
PAGE 19

```

613          .form 'Main: Send Diagnostic Interrupt to CPU'
614
615 01E4          sndiag:
616 01E4 2F      R      jsrl   rdwait   ; Wait for UPI interface ready.
617 01E5 971DE0 ld      obuf,#vdiag ; Load vector into OBUF for CPU.
618 01E8 2F      R      jsrl   rdwait   ; Wait for UPI interface ready.
619 01E9 96D018 rbit   gie,enir     ; *** Begin Indivisible Sequence ***
620 01EC 8C1DE0 R      ld      obuf,dsevc ; Transfer Severity Code.
621 01EF 97001D R      ld      dsevc,#0   ; Clear it.
622 01F2 881E   R      ld      A,derrc   ; Get Error Code.
623 01F4 97001E R      ld      derrc,#0   ; Clear it.
624 01F7 96021A R      rbit   adiag,alert.b ; Clear ALERT bit.
625 01FA 96D000 R      sbit   gie,enir     ; *** End Indivisible Sequence ***
626 01FD 2F      R      jsrl   rdwait   ; Wait for UPI interface ready.
627 01FE 88E0   R      st      A,obuf   ; Transfer Error Code.
628 0200 2F      R      jsrl   rdwait   ; Wait for UPI interface ready.
629          ; Remaining bytes will have meaning only for
630          ; command errors.
631 0201 8C1FE0 R      ld      obuf,dbyte ; Transfer Byte Received.
632 0204 2F      R      jsrl   rdwait   ; Wait for UPI interface ready.
633 0205 8C20E0 R      ld      obuf,dccmd ; Transfer Current Command.
634 0208 2F      R      jsrl   rdwait   ; Wait for UPI interface ready.
635 0209 8C21E0 R      ld      obuf,dqual ; Transfer Command Count.
636 020C 3C      R      ret     ; Return to main program loop.
637

```

TL/DD/9976-35

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
UPI (13) Interrupt: Data from CPU

HPCUPI

25-Feb-88 10:05
PAGE 20

```

638          .form 'UPI (13) Interrupt: Data from CPU'
639
640 FFFB 0002   R      .ipt   3,upiw   ; Declare upiw as vector for Interrupt 3.
641
642 0200          upiw:
643 0200 AFC8   push  A          ; Write Strobe received from CPU.
644 020F AFC0   push  psw       ; Save Context
645
646 0211 8CE600 R      ld      upicsv.b,upic ; Save UPIC register image for LAB bit test.
647
648 0214 961017 R      ifbit  cmdemp,curcmd ; If expecting first byte of a command,
649 0217 94CC   impl  firstc    ; then go process it as such.
650
651 0219 88F0   ld      A,ibuf   ; If not, input it for entry into cpubuf.
652
653 021B 9CA5   ifeq  A,#x'AS   ; Check for RESET command.
654 021D 46     jp      lcrst   ;
655 021E 960012 R      ifbit  la0,upicsv.b ; Check for command argument written to proper
656          ; address.
657 0221 48     jp      lcord   ; If so, go process as a normal argument.
658 0222 3622   jsrl  hangup    ; If not, process as a FATAL error, generating
659          ; IDIAG interrupt.
660
661 0224 96E612 lcrst: ifbit  la0,upic  ; Continue checking for a RESET command.
662 0227 42     jp      lcord   ;
663 0228 94C6   impl  xreset    ; If so, go reset the HPC.
664
665 022A AD048E R      lcord: x      A,[cpuad].b ; If not, place it in next available cpubuf
666          ; entry.
667 022D A904   R      inc    cpuad
668 022F BA11   R      decsz numexp
669 0231 B4010F R      jmp    upwret ; If not final byte of command, then return.
670
671 0234 8810   R      lastc: ld    A,curcmd ; Else, process current command.
672 0236 96C816 ifbit  getcnt,A,b ; Check if extended collection is being made.
673 0239 47     jp      lastc1  ; If not, then:
674 023A 96100F R      sbit  cmdemp,curcmd ; Set command slot available again.
675 023D 87000604 R      ld    cpuad,#cpubuf ; Reset CPU buffer pointer to beginning.
676
677 0241 991F   lastc1: and  A,#x'1F ; Mask off flag bits.
678 0243 E7     shl  A          ; Scale by two, and then
679 0244 40
680 0245          .odd
681 0245 EC     .jidw
682          ; jump based on command value:
683
682 0246 0A00   lastab: .ptw   lcinic ; 0 = INITIALIZE command.
683 0248 2C00   .ptw   lclscv ; 1 = SET-CONTRAST command.
684 024A 4200   .ptw   lclscd ; 2 = SEND-LCD command.
685 024C 8C00   .ptw   lclsed ; 3 = SEND-LED command.
686 024E F300   .ptw   illc  ; (BEEP command has only one byte. Error.)

```

TL/DD/9976-36

```

687
688
689 ; Process INITIALIZE Command.
690 0250 97011C R lclnit: ld rfevs,#x'01 ; Enable only Real-Time Clock interrupts, but
691 0253 8200050C R ifeq cpbuf.b,#0 ; disable them again if
692 0257 961C18 R rbit rtenb,rfevs ; the command argument is zero.
693 025A 8C061A R ld rtcivl,cpbuf.b ; Put argument into Real-Time
694 ; Clock interval.
695 025D 8C0618 R rtrcnt,cpbuf.b ; Put argument into Real-Time
696 ; Clock count.
697 0260 8601900C sbit t1tie,tmdl ; Enable Timer T1 interrupt, if not already
698 ; enabled.
699 0264 8601901E rbit t1stp,tmdl ; Start timer, if not already running.
700
701 0268 87000002 R ld alert.w,#0 ; Set no events pending.
702
703 026C 970017 R ld swlast,#0 ; Set up initial switch values.
704 026F 970018 R ld swlsnt,#0 ; (Both current and last sent)
705
706 0272 94CF jmpl upwret ; Return.
707
708
709 ; Process SET-CONTRAST Command.
710
711 0274 8806 R lclscv: ld A,cpbuf.b ; Load LCD Voltage latch (Contrast) from byte
712 ; supplied by CPU.
713 0276 01 comp A ; (3-bit value is in complemented form.)
714 0277 9907 and A,#x'07 ; Use only lower three bits.
715 0279 82F812D9 R and lcvs,#x'F8 ; Clear field in memory image.
716 027D 89C012DA R or lcvs,A,b ; Merge new field into image.
717 0281 8C12E1 R ld portah,lcvs ; Place on Port A (input to latch).
718 0284 96E309 sbit lcvclk,portbh ; Clock latch.
719 0287 96E319 rbit lcvclk,portbh
720 028A 9487 jmpl upwret
721
722
723 ; Process SEND-LCD Command.
724
725 028C 961016 R lclslcd: ifbit getcnt,curcmd ; Check for first or second collection
726 028F 9435 jmpl lclslc1 ; phase.
727
728 0291 lclslc2: ; Second phase: begins execution of the LCD
729 ; command.
730 0291 A1060030AB R ld lcdbuf.w,cpbuf.w ; Copy CPU buffer to LCD string buffer.
731 0296 A10B003AAB R ld lcdbuf+2.w,cpbuf+2.w
732 029B A10A003CAB R ld lcdbuf+4.w,cpbuf+4.w
733 02A0 A10C003EAB R ld lcdbuf+6.w,cpbuf+6.w
734 02A5 8C1416 R lcdsct,lcdnum ; Move number of characters to string
735 ; count byte
736 02AB 8916 R inc lcdsct ; (incremented by one because of

```

TL/DD/9976-37

```

737 ; extra interrupt occurring after
738 ; last character has been sent).
739 02AA B700300E R ld lcdsix,#lcdbuf ; Set string pointer to first byte.
740 02AE 8C1315 R ld lcdsfg,lcdfgs ; Move flag bits to string location.
741
742 02B1 87FFFF014AAB ld r6,#x'FFFF ; Set up R6 and T6 to trigger string
743 02B7 8300014BAB ld t6,#0 ; transfer.
744 02BC 8601510B sbit t6tie,punch ; Enable timer T6 interrupt.
745 02C0 8601511A rbit t6stp,punch ; Start timer to trigger (immediate)
746 ; interrupt from timer T6.
747 02C4 947D jmpl upwret
748
749 02C6 lclslc1: ; First phase: Prepare to collect up to 8
750 ; more bytes of command.
751 02C6 8C0613 R ld lcdfgs,cpbuf.b ; Get flag bits supplied by CPU.
752 02C9 8C0714 R ld lcdnum,cpbuf+1.b ; Get character count from CPU.
753
754 02CC 8C1411 R ld numexp,lcdnum.b ; Request another collection of
755 ; data from the CPU (the string of
756 ; data for the panel).
757 02CF B7000604 R ld cpued,#cpbuf ; Reset CPU collection pointer to start
758 ; of command buffer.
759 02D3 96101E R rbit getcnt,curcmd ; Declare that it will be the final
760 ; collection.
761 02D6 9468 jmpl upwret
762
763
764 ; Process SEND-LED Command.
765
766 02DB 8806 R lclsled: ld A,cpbuf.b ; Load LED latch from byte supplied by CPU.
767 02DA 01 comp A ; (Data goes to LED's in complemented form.)
768 02DB 8BE1 st A,portah ; Place new value on Port A (input to latch).
769 02DD 96E30E sbit ledclk,portbh ; Clock latch.
770 02E0 96E31E rbit ledclk,portbh
771 02E3 945E jmpl upwret
772
773

```

TL/DD/9976-38

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Processing of First Byte of Command (Code)

HPCUPI

25-Feb-88 10:05
PAGE 23

```

774                                     .form 'Processing of First Byte of Command (Code)'
775
776                                     ; One-byte commands are processed in this section.
777                                     ; Longer commands are scheduled for collection of
778                                     ; remaining bytes, and are processed in routines
779                                     ; above.
780
781 02E5 88F0          R firstc: ld   A,ibuf      ; Get command from UPI port.
782 02E7 960012      R          ifbit   l00,upicv.b ; Check for out-of-sequence condition
783                                     ; (argument instead of command).
784 02EA 36EA          R          jsrl   hangup   ; If so, process as a FATAL error (previous
785                                     ; command was too short).
786
787                                     ; Processing of RESET command.
788
789 02EC 9CA5          R          ifeq   A,#'A5     ; Check for RESET command.
790 02EE 41           R          jp     xreset
791 02EF 59           R          jp     fcord
792
793                                     ; This code is entered whenever a RESET
794                                     ; command is received.
795
796 02F0           R          xreset: ld     obuf,#vdiag ; Present dummy value for CPU,
797                                     ; (in case a value was already in OBUF),
798 02F3 2F           R          jsrl   rdwait   ; and wait for it to be read by CPU.
799 02F4 9000          R          ld     A,#0     ; Initialize registers.
800 02F6 B8E6          R          st     A,upic
801 02F8 ABF0          R          st     A,ibuf.w  ; (Actually all of DIRA.)
802 02FA ABF2          R          st     A,dirb
803 02FC ABF4          R          st     A,bfun
804 02FE B8D4          R          st     A,ircd
805 0300 B60152AB     R          st     A,portp
806 0304 ABC4          R          st     A,sp
807 0306 ABC0          R          st     A,psw   ; Then, through RESET vector,
808 0308 3C           R          ret
809                                     ; jump to start of program.
810
811                                     ; Here, process an ordinary command (not RESET).
812
813 0309           R          fcord:  and   A,#'1F   ; Use only least-significant 5 bits.
814 030B 9D11          R          ifgt   A,#'11   ; Check for command out of range.
815 030D 9432          R          jmlt   ilc
816 030F 8810          R          st     A,curcmd ; Save as current command.
817
818 0311 E7           R          shl   A         ; Scale by two, and then
819 0312 40           R          .odd
820 0313 EC           R          jidw
821                                     ; jump based on command value:
822 0314 0A00          R          firstab: .ptw fcinit ; 0 = INITIALIZE command.

```

TL/DD/9976-39

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Processing of First Byte of Command (Code)

HPCUPI

25-Feb-88 10:05
PAGE 24

```

823 0316 0000          .ptw fcslcv ; 1 = SET-CONTRAST command.
824 0318 0F00          .ptw fcslcd ; 2 = SEND-LCD command.
825 031A 1400          .ptw fcslcd ; 3 = SEND-LED command.
826 031C 1600          .ptw fcbeep ; 4 = BEEP command.
827
828 031E 970111       R fcinit: ld   numexp,#1 ; First byte of INITIALIZE command.
829                                     ; Expects 1 more byte (RTC interval).
830 0321 9420          R          jmlt   upwret ; Return.
831
832                                     ; First byte of SET-CONTRAST command.
833 0323 970111       R fcslcv: ld   numexp,#1 ; Set up to expect one more byte.
834 0326 5C           R          jmlt   upwret
835
836                                     ; First byte of SEND-LCD command.
837 0327 970211       R fcslcd: ld   numexp,#2 ; Set up to expect one more byte.
838 032A 96100E       R          sbit   getcnt,curcmd ; Note extended collection mode in Current
839                                     ; Command byte.
840 032D 55           R          jmlt   upwret
841
842                                     ; First byte of SEND-LED command.
843 032E 970111       R fcslcd: ld   numexp,#1 ; Send to LED's: Set up to expect one more byte.
844 0331 51           R          jmlt   upwret
845
846                                     ; Process one-byte BEEP command.
847 0332 96100F       R fcbeep: sbit   cmdemp,curcmd ; No arguments; set CURCMD byte empty.
848 0335 8601530F     R          sbit   t7tfn,portph ; Enable beep tone to panel speaker.
849 0339 86019008     R          sbit   t0tie,tmdl ; Enable Timer T0 interrupt.
850 033D 971319       R          ld     beepct,#19 ; Initialize duration count (approximately
851                                     ; 1 second, in units of Timer T0 overflows).
852 0340 42           R          jmlt   upwret
853
854
855 0341 3741          R          ilc:  jsrl   hangup   ; Process illegal command codes.
856
857                                     ; Return from UPI Write interrupt.
858 0343          R          upwret:
859 0343 3FC0          R          pop   psw
860 0345 3FC8          R          pop   A
861 0347 3E           R          reti
862

```

TL/DD/9976-40

```

863                .form 'Timer Interrupt Handler'
864
865 FFF4 4803        R        .ipt 5,tmrint ; Declare entry point for Timer Interrupt.
866
867 0348 AFC8        tmrint: push A ; Save context.
868 034A AFCC        push B ;
869 034C AFC0        push psw ;
870
871 034E B6019015    t1poll: ifbit t1pnd,tmmdl ; Poll for Timer T1 interrupt (Real-Time Clock).
872 0352 54          jmpl t1int ; If set, go service it.
873
874 0353 B6015111    t6poll: ifbit t6pnd,pwmdh ; Poll for Timer T6 interrupt (LCD Panel Timing
875 0357 944C        jmpl t6int ; Interrupt).
876
877 0359 B6019011    t8poll: ifbit t8pnd,tmmdl ; Poll for Timer T8 interrupt (Beep Duration).
878 035D 41          jp t8pdg ; If set, check the Enable bit; T8 is not
879 035E 46          jp t8notp ; always enabled to interrupt, but it runs
880 ; continuously.
881
882 035F B6019010    t8pdg: ifbit t8tie,tmmdl ; If enable is also set, then go service T8.
883 0363 9488        jmpl t8int ;
884 0365            t8notp: ; (This label is deliberately here.)
885
886 0365 3765        noint: jsrl hangup ; Error: no legal timer interrupt pending.
887
888

```

TL/DD/9976-41

```

889                .form 'Timer T1 Interrupt Service Routine'
890
891 0367 B601900F    t1int: sbit t1ack,tmmdl ; Acknowledge T1 interrupt.
892 0368 961C10        ifbit rtcenb,rtevs ; Check if RTC interrupts are enabled.
893 036E 41          jp t1int1
894 036F 57          jmpl kbdkchk ; If not, then go check other events.
895 0370 8A1B        R t1int1: decsz rtcnt ; Decrement interval value.
896 0372 54          jmpl kbdkchk ; If interval has not elapsed, then go check
897 ; for other events.
898 0373 8C1A1B        R ld rtcnt,rtcivl ; Reload counter value for next interval.
899 0376 960211        R ifbit rtc,alert.b ; Check if CPU has received previous interrupt
900 0379 44          jp t1rerr ; request; report error if not.
901 037A 960209        R sbit rtc,alert.b ; Set Real-Time Interrupt request to main
902 037D 49          jp kbdkchk ; program.
903 037E 961D08        R t1rerr: sbit 0,dsevc ; Signal NOTE severity.
904 0381 961E0F        R sbit 7,derrc ; Signal multiple-RTC error.
905 0384 96020A        R sbit addiag,alert.b ; Request !DIAG interrupt from main program.
906
907 0387            kbdkchk: ; Check keyboard switches.
908 0387 96E31D        rbit astts,portbh ; Enable pushbutton data to Port D.
909 038A B6018488        ld A,portd ; Sample pushbutton switches.
910 038E 96E30D        sbit astts,portbh ; Disable pushbutton data to Port D.
911 0391 98FF        xor A,#'FF ; Complement low-order 8 bits of A.
912 0393 8E17        R x A,swlast ; Exchange with last sample.
913 0395 96170C        R ifeq A,swlast ; Check if the data is stable (same as last
914 ; sample).
915 0398 41          jp kbint1
916 0399 49          jmpl tmochk ; If not, go check other events (if any).
917
918 039A 96180C        R kbint1: ifeq A,swlsnt ; Check if the data differs from the last
919 ; pattern sent to the CPU.
920 039D 45          jmpl tmochk ; If not, go check other events (if any).
921
922 039E 8B18        R st A,swlsnt ; Place new pattern in "last sent" location.
923 03A0 960208        R sbit abutton,alert.b ; Request "BUTTON-DATA" interrupt to CPU.
924
925
926 03A3            tmochk: ; *** Insert any other RTC events here. ***
927
928
929 03A3 9459        jmpl tmrret ; Return from Timer T1 interrupt.
930
931

```

TL/DD/9976-42

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Timer T6 Interrupt Service Routine

HPCUPI

25-Feb-88 10:05
PAGE 27

```

932 .form 'Timer T6 Interrupt Service Routine'
933
934 ; Timer T6 interrupt routine: sends characters from
935 ; LCD String Buffer to the panel.
936 t6int: sbit t6stp,pwmh ; Stop timer T6.
937 sbit t6ack,pwmh ; Acknowledge T6 interrupt.
938
939 R decsz lcdscnt ; Decrement LCD character count.
940 sbit t6nxtc ; If not done, go send another character.
941
942 R sbit alcdak,alert.b ; If done, request main program to send LCD
943 ; Acknowledge interrupt to CPU.
944 jmpl tmrret
945
946 R t6nxtc: ld A,lcdfsfg ; Get flags byte (for panel RS signal).
947 shr A ; Shift right, LSB into carry.
948 R st A,lcdfsfg ; Store shifted value back.
949 R sbit pn1rs,lcvs ; Determine proper state for RS signal from
950 R ifc ; current character's flag (= flag inverted).
951 R rbit pn1rs,lcvs ; Send new RS value to LCD Voltage (LCV) latch.
952 R ld portah,lcvs ; Clock the latch. RS signal is now valid.
953 R sbit lcvclk,portbh
954 R rbit lcvclk,portbh
955
956 R ld A,[lcdsix].b ; Get next LCD character from string buffer.
957 R inc lcdsix ; Increment character pointer.
958 R comp A ; Complement character, then
959 R st A,portah ; place it on Port A for LCD display.
960 R rbit pn1clk,portbl ; Clock it into panel.
961 R sbit pn1clk,portbl
962 R comp A ; Restore A to uncomplemented form for
963 ; test performed below.
964
965 ld t6,#148 ; Set up normal delay time in timer T6
966 ; (120 microseconds).
967 ifgt A,#x'03 ; Check whether the longer delay
968 R jp t6nxt2 ; (4.9 milliseconds) is necessary.
969 ; This happens if RS=0 and the byte sent to
970 R ifnc ; the panel is a value of hex 03 or less.
971 R ld t6,#6022 ; If so, change timer to 4.9 milliseconds.
972
973 t6nxt2: rbit t6stp,pwmh ; Start Timer T6 to time out the character.
974 R jmpl tmrret ; Return from the interrupt.
975
976

```

TL/DD/9976-43

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Timer T0 Interrupt Service Routine

HPCUPI

25-Feb-88 10:05
PAGE 28

```

977 .form 'Timer T0 Interrupt Service Routine'
978
979 t0int: ; Count duration of beep tone. Restore beep signal
980 ; to zero and re-enable switch sampling interrupt
981 ; when done.
982 R sbit t0ack,tmdl ; Acknowledge interrupt from Timer T0.
983 R decsz beepct ; Check whether beep time has finished.
984 R jmpl tmrret ; No: return from interrupt.
985 R rbit t0tie,tmdl ; Yes: disable Timer T0 interrupts and
986 ; continue.
987 R and portph,#x'0F ; Disable speaker output.
988 R jmpl tmrret ; Return from interrupt.
989
990 ; Common return for timer interrupt service routines.
991 R tmrret: pop psw ; Restore context.
992 R pop B
993 R pop A
994 R reti
995
996

```

TL/DD/9976-44

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Subroutine to Wait for OBUF Empty

HPCUPI

25-Feb-88 10:05
PAGE 29

```

997 .form 'Subroutine to Wait for OBUF Empty'
998
999 ; RDWAIT subroutine: waits until the CPU has read a byte from the
1000 ; UPI interface.
1001
1002 R rdwait: ifbit rdrdy,upic ; Check to see if OBUF register is full.
1003 R ret
1004 R jp rdwait
1005
1006

```

TL/DD/9976-45

```

1007          .form 'Write to Panel Subroutine'
1008
1009          ; Write Panel subroutine.
1010          ; Used only at initialization or to report a
1011          ; fatal protocol error, since it performs
1012          ; the timing delay using timer T6 without interrupts.
1013          ; (Panel RS signal must be set up previously in the
1014          ; LCV latch by the calling routine.)
1015
1016 040A 01      wrpnl: comp  A          ; Complement value for bus.
1017 040B 8BE1    st        A,portah    ; Put value on panel bus.
1018 040D 96E21F rbit     pnlclk,portbl ; Set Panel Clock low,
1019 0410 96E20F  sbit     pnlclk,portbl ; then high again;
1020          ; pulse width approx.
1021          ; 1.2 microsec.
1022
1023          ; Wait for another
1024          ; 4.9 milliseconds (twice).
1025 0413 8732C80140AB ld      t6,#13000    ; Twice 4.9 milliseconds.
1026 0419 8601511A rbit     t6stp,pwmch  ; Start timer T6.
1027 041D 86015111 wrppl:  ifbit  t6pnd,pwmch ; Wait for PND to be set.
1028 0421 41      jp        wrpgo
1029 0422 65      jp        wrppl
1030 0423 8601510A wrpgo:  sbit     t6stp,pwmch ; Stop timer T6.
1031 0427 8601510B sbit     t6ack,pwmch  ; Clear T6 PND bit.
1032 0428 3C      ret          ; Return from subroutine.
1033
1034          ; END OF PROGRAM: RESET VECTOR SET TO LABEL "start".
1035
1036 042C          .end  start

```

TL/DD/9976-46

abutton	0000	Abs	Null	
adiag	0002	Abs	Null	
ah	00C9	Abs	Byte	
al	00C8	Abs	Byte	
alcclak	0003	Abs	Null	
alert	0002	Rel	Word	BASE
alerth	0003	Rel	Byte	BASE
artc	0001	Abs	Null	
astts	0005	Abs	Null	
avail	0020	Rel	Word	RAM16
b2stp	0007	Abs	Null	
b8or16	0004	Abs	Null	
b8or9	0004	Abs	Null	
beepcct	0019	Rel	Byte	BASE
bfun	00F4	Abs	Word	
bfunh	00F5	Abs	Byte	
bfunl	00F4	Abs	Byte	
bh	00CD	Abs	Byte	
bl	00CC	Abs	Byte	
cdata	0004	Abs	Null	
chkalt	01A8	Rel	Null	ROM16
cmdemp	0007	Abs	Null	
cpuad	0004	Rel	Word	BASE
cpubuf	0006	Rel	Word	BASE
curcmd	0010	Rel	Byte	BASE
dbyte	001F	Rel	Byte	BASE
decmd	0020	Rel	Byte	BASE
derrc	001E	Rel	Byte	BASE
dirah	00F1	Abs	Byte	
dirb	00F2	Abs	Word	
dirbh	00F3	Abs	Byte	
dirbl	00F2	Abs	Byte	
divby	018E	Abs	Word	
divbyh	018F	Abs	Byte	
divbyl	018E	Abs	Byte	
doeerr	0007	Abs	Null	
dqual	0021	Rel	Byte	BASE
dsevc	001D	Rel	Byte	BASE
dummy	0000	Rel	Word	BASE
ei	0007	Abs	Null	
eiack	0002	Abs	Null	
eicon	015C	Abs	Byte	
eimode	0001	Abs	Null	
eipol	0000	Abs	Null	
enir	0000	Abs	Byte	
enu	0120	Abs	Byte	
enui	0122	Abs	Byte	
enur	0128	Abs	Byte	
eri	0001	Abs	Null	
eti	0000	Abs	Null	

TL/DD/9976-47

fcbeep	0332	Rel	Null	ROM16
fcinit	031E	Rel	Null	ROM16
fcord	0309	Rel	Null	ROM16
fcslcd	0327	Rel	Null	ROM16
fcslcv	0323	Rel	Null	ROM16
fcstled	032E	Rel	Null	ROM16
firstab	0314	Rel	Null	ROM16
firstc	02E5	Rel	Null	ROM16
fmerr	0006	Abs	Null	
getcnc	0006	Abs	Null	
gie	0000	Abs	Null	
hangup	0000	Rel	Null	ROM16
hextab	007A	Rel	Byte	ROM16
hgrst	006F	Rel	Null	ROM16
hgupi	005E	Rel	Null	ROM16
hgupi1	0069	Rel	Null	ROM16
hgupi2	0076	Rel	Null	ROM16
i2	0002	Abs	Null	
i3	0003	Abs	Null	
i4	0004	Abs	Null	
ibuf	00F0	Abs	Byte	
illc	0341	Rel	Null	ROM16
ircd	0004	Abs	Byte	
irpd	0002	Abs	Byte	
kbdchk	0387	Rel	Null	ROM16
kbint1	039A	Rel	Null	ROM16
lab	0002	Abs	Null	
lastab	0246	Rel	Null	ROM16
lastc	0234	Rel	Null	ROM16
lastc1	0241	Rel	Null	ROM16
lcdbuf	0038	Rel	Word	RAM16
lcdfgs	0013	Rel	Byte	BASE
lcdgo1	0171	Rel	Null	ROM16
lcdlp1	0168	Rel	Null	ROM16
lcdnum	0014	Rel	Byte	BASE
lcdsct	0016	Rel	Byte	BASE
lcdsfg	0015	Rel	Byte	BASE
lcdsix	000E	Rel	Word	BASE
lcinit	0250	Rel	Null	ROM16
lcoord	022A	Rel	Null	ROM16
lcrst	0224	Rel	Null	ROM16
lcslc1	02C6	Rel	Null	ROM16
lcslc2	0291	Rel	Null	ROM16
lcslcd	028C	Rel	Null	ROM16
lcslcV	0274	Rel	Null	ROM16
lcsled	0208	Rel	Null	ROM16
lcvclk	0001	Abs	Null	
lcvs	0012	Rel	Byte	BASE
ledclk	0006	Abs	Null	
mainlp	01A8	Rel	Null	ROM16

minit	0191	Rel	Null	ROM16
noint	0305	Rel	Null	ROM16
numexp	0011	Rel	Byte	BASE
obuf	0000	Abs	Byte	
pnlclk	0007	Abs	Null	
pnlrs	0003	Abs	Null	
portah	00E1	Abs	Byte	
portb	00E2	Abs	Word	
portbh	00E3	Abs	Byte	
portbl	00E2	Abs	Byte	
portd	0104	Abs	Byte	
porti	0008	Abs	Byte	
portp	0152	Abs	Word	
portph	0153	Abs	Byte	
portpl	0152	Abs	Byte	
psw	00C0	Abs	Word	
pwmh	0151	Abs	Byte	
pwmcl	0150	Abs	Byte	
pwmode	0150	Abs	Word	
r1	0184	Abs	Word	
r2	0186	Abs	Word	
r3	018A	Abs	Word	
r4	0142	Abs	Word	
r5	0146	Abs	Word	
r6	014A	Abs	Word	
r7	014E	Abs	Word	
rbfl	0001	Abs	Null	
rbit9	0003	Abs	Null	
rbuf	0124	Abs	Byte	
rdrdy	0001	Abs	Null	
rdwait	0405	Rel	Null	ROM16
rtccnt	001B	Rel	Byte	BASE
rtcenb	0000	Abs	Null	
rtcivl	001A	Rel	Byte	BASE
rtevs	001C	Rel	Byte	BASE
runsys	019F	Rel	Null	ROM16
sio	00D6	Abs	Byte	
sk	0006	Abs	Null	
slcd	0145	Rel	Null	ROM16
sled	0104	Rel	Null	ROM16
sndbtn	01D2	Rel	Null	ROM16
sndiag	01E4	Rel	Null	ROM16
sndlak	01CA	Rel	Null	ROM16
sndrtc	01C2	Rel	Null	ROM16
so	0005	Abs	Null	
sram	00C4	Rel	Null	ROM16
sraml1	00C7	Rel	Null	ROM16
sraml2	00CF	Rel	Null	ROM16
srfsh	00B0	Rel	Null	ROM16
sskint	00D2	Rel	Null	ROM16

TL/DD/9976-49

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)
 UPI PORT INTERFACE DEMO
 Write to Panel Subroutine

HPCUPI

25-Feb-88 10:05
 PAGE 34

stackb	0000	Rel	Word	RAM16
start	000A	Rel	Null	ROM16
stmrs	0113	Rel	Null	ROM16
supi	00A7	Rel	Null	ROM16
swlast	0017	Rel	Byte	BASE
swlsnt	0018	Rel	Byte	BASE
t0ack	0003	Abs	Null	
t0con	0192	Abs	Byte	
t0int	03ED	Rel	Null	ROM16
t0notp	0365	Rel	Null	ROM16
t0pdg	035F	Rel	Null	ROM16
t0pnd	0001	Abs	Null	
t0poll	0359	Rel	Null	ROM16
t0tie	0000	Abs	Null	
t1	0182	Abs	Word	
t1ack	0007	Abs	Null	
t1int	0367	Rel	Null	ROM16
t1int1	0370	Rel	Null	ROM16
t1pnd	0005	Abs	Null	
t1poll	034E	Rel	Null	ROM16
t1rerr	037E	Rel	Null	ROM16
t1stp	0006	Abs	Null	
t1tie	0004	Abs	Null	
t2	0180	Abs	Word	
t2ack	0003	Abs	Null	
t2pnd	0001	Abs	Null	
t2stp	0002	Abs	Null	
t2tie	0000	Abs	Null	
t3	018C	Abs	Word	
t3ack	0007	Abs	Null	
t3pnd	0005	Abs	Null	
t3stp	0006	Abs	Null	
t3tie	0004	Abs	Null	
t4	0140	Abs	Word	
t4ack	0003	Abs	Null	
t4out	0000	Abs	Null	
t4pnd	0001	Abs	Null	
t4stp	0002	Abs	Null	
t4tfn	0003	Abs	Null	
t4tie	0000	Abs	Null	
t5	0144	Abs	Word	
t5ack	0007	Abs	Null	
t5out	0004	Abs	Null	
t5pnd	0005	Abs	Null	
t5stp	0006	Abs	Null	
t5tfn	0007	Abs	Null	
t5tie	0004	Abs	Null	
t6	0148	Abs	Word	
t6ack	0003	Abs	Null	
t6int	03A5	Rel	Null	ROM16

TL/DD/9976-50

NSC ASMHC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Write to Panel Subroutine

HPCUPI

25-Feb-88 10:05
PAGE 35

t6nxt2	03E8	Rel	Null	ROM16
t6nxtc	0305	Rel	Null	ROM16
t6out	0000	Abs	Null	
t6pnd	0001	Abs	Null	
t6poll	0353	Rel	Null	ROM16
t6stp	0002	Abs	Null	
t6tfn	0003	Abs	Null	
t6tie	0000	Abs	Null	
t7	014C	Abs	Word	
t7ack	0007	Abs	Null	
t7out	0004	Abs	Null	
t7pnd	0005	Abs	Null	
t7stp	0006	Abs	Null	
t7tfn	0007	Abs	Null	
t7tie	0004	Abs	Null	
tbmt	0000	Abs	Null	
tbuf	0126	Abs	Byte	
tminit	000F	Rel	Null	ROM16
tmsh	0191	Abs	Byte	
tmmdl	0190	Abs	Byte	
tmode	0190	Abs	Word	
tmochk	03A3	Rel	Null	ROM16
tmrint	0348	Rel	Null	ROM16
tmrret	03FE	Rel	Null	ROM16
tmrs	0005	Abs	Null	
uart	0006	Abs	Null	
upic	00E6	Abs	Byte	
upicsv	0000	Rel	Word	BASE
upien	0003	Abs	Null	
upiw	020D	Rel	Null	ROM16
upwret	0343	Rel	Null	ROM16
urdrdy	0007	Abs	Null	
usdone	0000	Abs	Null	
usmode	0001	Abs	Null	
uwrrdy	0003	Abs	Null	
vbutton	0018	Abs	Null	
vdiag	001D	Abs	Null	
vicdak	0017	Abs	Null	
vrtc	0011	Abs	Null	
wakeup	0002	Abs	Null	
wrpgp	0423	Rel	Null	ROM16
wrplp	041D	Rel	Null	ROM16
wrpl	040A	Rel	Null	ROM16
wrrdy	0000	Abs	Null	
xbit9	0005	Abs	Null	
xh	00CF	Abs	Byte	
xl	00CE	Abs	Byte	
xoff	0013	Abs	Null	
xon	0011	Abs	Null	
xrclk	0003	Abs	Null	

TL/DD/9976-51

NSC ASMHC, Ver D1-BetaSite (Sep 14 14:30 1987)
UPI PORT INTERFACE DEMO
Write to Panel Subroutine

HPCUPI

25-Feb-88 10:05
PAGE 36

xreset	02F0	Rel	Null	ROM16
xtclk	0002	Abs	Null	

**** Errors: 0, Warnings: 0

TL/DD/9976-52

4.3 Two Demo Programs (NS32CG16 Source Code)

The following two programs run on the NS32CG16 CPU, and exercise the functions implemented in the HPC firmware.

One thing to note in this software is that the interrupt service routines are not written as such; they are simple subroutines called by the actual service routines, which are contained within a modified version of the MON16 monitor program. The reasons for modifying MON16 were two-fold:

1. There is no RAM in the application system within the first 64k of the addressing space. The presence of RAM there is necessary for MON16 to support custom interrupt handlers without internal modification.
2. The HPC requires use of the "RETT 0" instruction, rather than "RETI", to return from maskable as well as non-maskable interrupts.

Given these two constraints, it was considered most useful to modify MON16 to contain a set of interrupt service routines, which would then use a set of addresses in RAM (a table at address "vex") to call custom interrupt servers as standard subroutines. An interrupt service routine calls its custom subroutine after saving the dedicated registers and the general registers, R0, R1 and R2 on the stack.

The symbol "vex" is defined externally, and must be declared to match the address used by the modified MON16. Details of the modified MON16 are available from National Semiconductor Corporation, Microprocessor Applications

Group or the Microcontroller Applications Group, phone (408) 721-5000. These modifications are also a standard part of the MONCG monitor program for the NS32CG016 microprocessor.

4.3.1 Panel Exerciser Program

This program for the NS32CG16 CPU exercises several functions of a panel consisting of the following:

- A two-line (8 chars. per line) LCD panel, arranged horizontally into a single 16-line display.
- A speaker, activated by the BEEP command.
- Six pushbuttons, which are presented by the IBUTTON-DATA interrupt to the CPU as follows:

Keyboard Status Byte

0	PB6	PB5	PB4	0	PB2	PB1	PB0
---	-----	-----	-----	---	-----	-----	-----

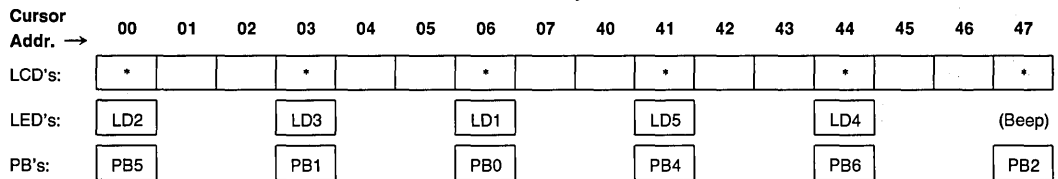
- Five LED's, activated in the SEND-LED command by the following bits:

LED Control Byte

—	—	LD5	LD4	LD3	LD2	LD1	—
---	---	-----	-----	-----	-----	-----	---

The intended layout for the front panel is as shown below. (Please pardon the apparently haphazard assignment of the pushbuttons and LED's; this was dictated by the nature of the module we used for developing this application.)

Front Panel Layout



The locations shown with asterisks on the LCD panel above will display an asterisk character while the corresponding pushbutton below it is depressed. (The number above each LCD location indicates its cursor address in hexadecimal.)

Each time a pushbutton (except PB2) is pressed, the corresponding LED indicator above it is toggled. Rather than toggling an LED, PB2 causes a BEEP command to be issued. The program starts up the panel with the LCD display blank, and LED's LD1 and LD2 on.

```

1
2           # Front Panel Exerciser Program.
3
4           # "vex" contains absolute address of NMI service routine entry point.
5           # "vex"+4 starts list of maskable interrupt routine entry points;
6           #           first is interrupt 0x10.
7
8           # Note: This code assumes that it is running in Supervisor Mode.
9           #           Before running, make sure to set PSR to 0200 hex.
10          #           Also, all unused interrupts automatically branch to label
11          #           "badint"; a breakpoint should be set there.
12
13
14          .globl start,main
15          .globl rtcint
16          .globl lcdint
17          .globl swint
18          .globl badint
19
20          .set hpcctrl,0xFFFC00      # HPC Control/Status I/O location.
21          .set hpcdata,0xFFFE00     # HPC Data I/O location.
22          .set hpcpoll,0xFD0000     # HPC Poll address (UPIC).
23
24          .set INIT,0x0
25          .set SET_CONT,0x1
26          .set SEND_LCD,0x2
27          .set SEND_LED,0x3
28          .set BEEP,0x4
29          .set RESET_HPC,0xA5
30
31          start:
32
33          # Fill interrupt vector locations.
34          T0000000 67ddc000      addr badint,vex      # Interrupt NMI. (Unimplemented)
35                  025a0000
36                  0000
37          T000000a 67ddc000      addr badint,vex+4     # Interrupt 0x10. (Unimplemented)
38                  02500000
39                  0004
40          T0000014 67ddc000      addr rtcint,vex+8    # Interrupt 0x11. Real-Time Clock.
41                  021c0000
42                  0008
43          T000001e 67ddc000      addr badint,vex+12  # Interrupt 0x12. (Unimplemented)
44                  023c0000
45                  000c
46          T0000028 67ddc000      addr badint,vex+16  # Interrupt 0x13. (Unimplemented)
47                  02320000
48                  0010
49          T0000032 67ddc000      addr badint,vex+20  # Interrupt 0x14. (Unimplemented)
50                  02280000
51                  0014
52          T000003c 67ddc000      addr badint,vex+24  # Interrupt 0x15. (Unimplemented)
53                  021e0000
54                  0018
55          T0000046 67ddc000      addr badint,vex+28  # Interrupt 0x16. (Unimplemented)

```

TL/DD/9976-53

```

02140000
001c
42 T00000050 67ddc000      addr  lcdint,vex+32      # Interrupt 0x17. LCD data written.
01e80000
0020
43 T0000005a 67ddc000      addr  swint,vex+36      # Interrupt 0x18. Pushbutton event.
01ea0000
0024
44 T00000064 67ddc000      addr  badint,vex+40     # Interrupt 0x19. (Unimplemented)
01f60000
0028
45 T0000006e 67ddc000      addr  badint,vex+44     # Interrupt 0x1A. (Unimplemented)
01ec0000
002c
46 T00000078 67ddc000      addr  badint,vex+48     # Interrupt 0x1B. (Unimplemented)
0030
47 T00000082 67ddc000      addr  badint,vex+52     # Interrupt 0x1C. (Unimplemented)
01d80000
0034
48 T0000008c 67ddc000      addr  badint,vex+56     # Interrupt 0x1D. Diagnostic: stop.
01ce0000
0038
49 T00000096 67ddc000      addr  badint,vex+60     # Interrupt 0x1E. (Unimplemented)
01c40000
003c
50 T000000a0 67ddc000      addr  badint,vex+64     # Interrupt 0x1F. (Unimplemented)
01ba0000
0040
51 T000000aa 67ddc000      addr  badint,vex+68     # Interrupt 0x20. (Unimplemented)
01b80000
0044
52 T000000b4 67ddc000      addr  badint,vex+72     # Interrupt 0x21. (Unimplemented)
01a60000
0048
53
54 T000000be 54a500c0      movb  $INIT,hpcctrl   # INITIALIZE command.
ffc000
55 T000000c5 54a500c0      movb  $0,hpcdata     # RTC value: feature disabled.
ffe000
56
57 T000000cc 54a5003c0     movb  $SEND_LED,hpcctrl # Initialize LEDs to normal state.
ffc000
58 T000000d3 54a5006c0     movb  $0x06,hpcdata
ffe000
59 T000000da c4a6006c0     movb  $0x06,leds     # Save in memory image.
000145
60
61
62 T000000e1 7da30800      run:   bispsrw $0x800     # Enable interrupts from HPC.
63
64
65
66
67 T000000e5 5cd8c000      main:  movqb  $0,lcdflg     # Set waiting for LCD.

```

```

                                0139
68
69 T000000eb 54a502c0 movb $SEND_LCD,hpcctrl # Turn off LCD cursor and clear panel.
   fffc00
70 T000000f2 54a500c0 movb $0,hpcdata
   fffc00
71 T000000f9 54a502c0 movb $2,hpcdata
   fffc00
72 T00000100 54a500c0 movb $0x0C,hpcdata
   fffc00
73 T00000107 54a501c0 movb $1,hpcdata
   fffc00
74
75 T0000010e f4a600c0 l1: tbitb $0,lcdflg # Wait for panel available.
   000110
76 T00000115 9a79 bfc l1
77
78 T00000117 5cd8c000 movqb $0,kbdflg
   0104
79
80
81 kbdlp:
82
83 T0000011d f4a600c0 l2: tbitb $0,kbdflg # Wait for keyboard data.
   0000fe
84 T00000124 9a79 bfc l2
85
86 T00000126 7da10000 bicpsrw $0x800 # Sample, and update semaphores.
87 T0000012a 14d8c000 movb kbdnew,r0
   00f2
88 T00000130 54d8c000 movb kbdold,r1
   00ed
89 T00000136 c4dec000 movb kbdnew,kbdold
   00e7
90 T00000140 5cd8c000 movqb $0,kbdflg
   00db
91 T00000146 7da30000 bispsrw $0x800
92
93 T0000014a 5f10 movqd $0,r2 # Initialize offset pointer in r2.
94
95 T0000014c 7800 xorb r0,r1 # Generate map of differing bits.
96 T0000014e 1c08 cmpqb $0,r1 # Check that a change actually occurred.
97 T00000150 1a10 bne lcdlp
98
99 T00000152 54a503c0 movb $SEND_LED,hpcctrl # If not, error is shown by turning on
100 T00000159 54a520c0 movb $0x20,hpcdata # ALARM LED.
   fffc00
101
102
103
104 T00000160 6e8408 lcdlp: ffsb r1,r2 # Find first differing bit.
105 T00000163 0abfba bfs kbdlp # If none, go wait for another keyboard event.
106 T00000166 4e4810 cbitb r2,r1 # Clear difference flag.

```

TL/DD/9976-55

```

107
108 T00000169 5cd8c000      movqb  $0,lcdflg      # Do LCD command: first clear Acknowledge flag.
      00b5
109
110 T0000016f 74a500c0      l3:  tbitb $0,hpcpoll
      fd0000
111 T00000176 8a79          bfs    l3
112 T00000178 54a502c0      movb  $SEND_LCD,hpcctrl # Start command to display new bit state.
      fffc00
113
114 T0000017f 74a500c0      l4:  tbitb $0,hpcpoll
      fd0000
115 T00000186 8a79          bfs    l4
116 T00000188 54a502c0      movb  $2,hpcdata      # Flags: One command followed by one data.
      fffe00
117
118 T0000018f 74a500c0      l5:  tbitb $0,hpcpoll
      fd0000
119 T00000196 8a79          bfs    l5
120 T00000198 54a502c0      movb  $2,hpcdata      # Two data bytes follow.
      fffe00
121
122 T0000019f 74a500c0      l6:  tbitb $0,hpcpoll
      fd0000
123 T000001a6 8a79          bfs    l6
124 T000001a8 54e5dac0      movb  lcdloc[r2:b],hpcdata # Send cursor position byte.
      000078c0
      fffe00
125
126 T000001b3 74a500c0      l7:  tbitb $0,hpcpoll
      fd0000
127 T000001ba 8a79          bfs    l7
128 T000001bc 3410          tbitb r2,r0
129 T000001be 8a0c          bfs    l8
130 T000001c0 54a520c0      movb  $0x20,hpcdata    # If new bit is zero, send blank.
      fffe00
131 T000001c7 ea8046       br    lout
132
133 T000001ca 54a52ac0      l8:  movb  $0x2A,hpcdata  # If bit is one, send asterisk instead,
      fffe00
134
135 T000001d1 74a500c0      l9:  tbitb $0,hpcpoll
      fd0000
136 T000001d8 8a79          bfs    l9
137 T000001da 34a002       tbitb $2,r0            # and if the key is MENU,
138 T000001dd 9a0b         bfc    l10
139
140 T000001df 54a504c0      movb  $BEEP,hpcctrl    # then beep,
      fffc00
141 T000001e6 ea27         br    lout
142
143 T000001e8 54a503c0      l10: movb  $SEND_LED,hpcctrl # else toggle appropriate LED.
      fffc00
144 T000001ef f8e6dac0     xorb  ledloc[r2:b],leds
      000039c0

```

TL/DD/9976-56

```

145 T000001fa 000030          l11:  tbitb  $0,hcpoll
      74a500c0
      fd0000
146 T00000201 8a79          bfs    l11
147 T00000203 54ddc000     movb  leds,hpcdata
      001cc0ff
      fe00

148
149 T0000020d f4a600c0     lout: tbitb  $0,lcdflg  # Wait for LCD Acknowledge interrupt.
      000011
      9a79          bfc   lout
150 T00000214 9a79          br    lcdlp          # Go check for any more differing bits.
151
152 T00000216 eabf4a          ret   0          # End of main program.
153
154
155 T00000219 1200          ret   0          # End of main program.
156
157
158 maindat:      # Data for Main Program.
159
160 T0000021b 00          kbdflg: .byte 0          # Keyboard data ready.
161 T0000021c 00          kbdnew: .byte 0          # New keyboard data (from interrupt service).
162 T0000021d 00          kbdold: .byte 0          # Saved (previous) keyboard states.
163 T0000021e 00          lcdflg: .byte 0          # LCD display ready.
164 T0000021f 00          leds:  .byte 0          # LED states.
165
166 T00000220 8683c781     lcdloc: .byte 0x86,0x83,0xC7,0x81,0xC1,0x80,0xC4,0x81
      c180c481
167 T00000228 02050000     ledloc: .byte 0x02,0x08,0x00,0x00,0x20,0x04,0x10,0x00
      20041000

168
169
170
171          # Start of Interrupt Service Routines.
172          # Invoked by ROM interrupt service.  Registers R0..R2 are already
173          # saved, but no ENTER instruction has been performed yet.
174          # Because ROM monitor returns using "RETI", we must bypass it
175          # and return directly with "RETI 0".
176
177 rtcint:      # Interrupt 0x11.  Real-Time Clock.
178
179 T00000230 ea2a          br    badint      # UNEXPECTED (bypass code below)
180
181 T00000232 1fb8          cmpqd  $0,tos     # Interrupt return procedure:
182 T00000234 72e0         restore [r0,r1,r2] # Discard return address to monitor.
183 T00000236 4200         rett   0          # Restore registers saved by monitor.
184
185          # Return from interrupt directly.
186
187 lcdint:      # Interrupt 0x17.  LCD data written.
188 T00000238 dcd8ffff     movqb  $1,lcdflg  # Flag that interrupt has occurred.
      ffe6
189
190          # Interrupt return procedure:
191          # Discard return address to monitor.
192          # Restore registers saved by monitor.
193          # Return from interrupt directly.
194
195 T00000240 1fb8          cmpqd  $0,tos     # Interrupt return procedure:
196 T00000242 72e0         restore [r0,r1,r2] # Discard return address to monitor.
197 T00000244 4200         rett   0          # Restore registers saved by monitor.
198
199          # Return from interrupt directly.
200
201 badint:      # Trap for unimplemented interrupts.  PLACE BREAKPOINT HERE.
202
203          # Interrupt return procedure:
204          # Discard return address to monitor.
205          # Restore registers saved by monitor.
206          # Return from interrupt directly.

```

TL/DD/9976-57

```

192
193 T00000244 dcd8ffff     swint: movqb  $1,kbdflg  # Interrupt 0x18.  Pushbutton event.
      ffd7          # Flag that interrupt has occurred.
194 T0000024a d4aec0ff     movb  hpcdata,kbdnew # Save new keyboard state.
      fe00ffff
      ffd2

195
196 T00000254 1fb8          cmpqd  $0,tos     # Interrupt return procedure:
197 T00000256 72e0         restore [r0,r1,r2] # Discard return address to monitor.
198 T00000258 4200         rett   0          # Restore registers saved by monitor.
199
200          # Return from interrupt directly.
201
202 badint:      # Trap for unimplemented interrupts.  PLACE BREAKPOINT HERE.
203
204          # Interrupt return procedure:
205          # Discard return address to monitor.
206          # Restore registers saved by monitor.
207          # Return from interrupt directly.

```

TL/DD/9976-58

4.3.2 Real-Time Clock Display Program

This program (rtc.s) enables the Real-Time Clock interrupts from the HPC, and counts them to generate a display of elapsed time on the LCD panel.

GNX Series32000 COFF ASSEMBLER Version 2.5 6/6/88

Page: 1

```

1
2
3      # Real-Time Clock Exerciser: Places elapsed time in seconds onto
4      #
5      # "vex" contains absolute address of NMI service routine entry point.
6      # "vex"+4 starts list of maskable interrupt routine entry points;
7      # first is interrupt 0x10.
8
9      # Note: This code assumes that it is running in Supervisor Mode.
10     # Before running, make sure to set PSR to 0200 hex.
11     # Also, all unused interrupts automatically branch to label
12     # "badint"; a breakpoint should be set there.
13
14
15     .globl start,main
16     .globl rtcint
17     .globl lcdint
18     .globl swint
19     .globl badint
20
21     .set hpcctrl,0xFFFC00      # HPC Control/Status I/O location.
22     .set hpcdata,0xFFFE00     # HPC Data I/O location.
23     .set hpcpoll,0xFD0000     # HPC Poll address (UPIC).
24     .set INIT,0x0
25     .set SET_CONT,0x1
26     .set SEND_LCD,0x2
27     .set SEND_LED,0x3
28     .set BEEP,0x4
29     .set RESET_HPC,0xA5
30
31     start:
32
33     # Fill interrupt vector locations.
34     T0000000 67ddc000      addr badint,vex          # Interrupt NMI. (Unimplemented)
35     024e0000
36     0000
37     T000000a 67ddc000      addr badint,vex+4        # Interrupt 0x10. (Unimplemented)
38     02440000
39     0004
40     T0000014 67ddc000      addr rtcint,vex+8      # Interrupt 0x11. Real-Time Clock.
41     02040000
42     0008
43     T000001e 67ddc000      addr badint,vex+12    # Interrupt 0x12. (Unimplemented)
44     02300000
45     000c
46     T0000028 67ddc000      addr badint,vex+16    # Interrupt 0x13. (Unimplemented)
47     02260000
48     0010
49     T0000032 67ddc000      addr badint,vex+20    # Interrupt 0x14. (Unimplemented)
50     021c0000
51     0014
52     T000003c 67ddc000      addr badint,vex+24    # Interrupt 0x15. (Unimplemented)
53     02120000
54     0018
55     T0000046 67ddc000      addr badint,vex+28    # Interrupt 0x16. (Unimplemented)
56     001B

```

TL/DD/9976-59

```

                02080000
                001c
42 T00000050 67ddc000      addr  lcdint,vex+32      # Interrupt 0x17. LCD data written.
                01e40000
                0020
43 T0000005a 67ddc000      addr  swint,vex+36      # Interrupt 0x18. Pushbutton event.
                01e80000
                0024
44 T00000064 67ddc000      addr  badint,vex+40     # Interrupt 0x19. (Unimplemented)
                01ea0000
                0028
45 T0000006e 67ddc000      addr  badint,vex+44     # Interrupt 0x1A. (Unimplemented)
                01ec0000
                002c
46 T00000078 67ddc000      addr  badint,vex+48     # Interrupt 0x1B. (Unimplemented)
                01d60000
                0030
47 T00000082 67ddc000      addr  badint,vex+52     # Interrupt 0x1C. (Unimplemented)
                01cc0000
                0034
48 T0000008c 67ddc000      addr  badint,vex+56     # Interrupt 0x1D. Diagnostic: stop.
                01c20000
                0038
49 T00000096 67ddc000      addr  badint,vex+60     # Interrupt 0x1E. (Unimplemented)
                01b80000
                003c
50 T000000a0 67ddc000      addr  badint,vex+64     # Interrupt 0x1F. (Unimplemented)
                01ae0000
                0040
51 T000000aa 67ddc000      addr  badint,vex+68     # Interrupt 0x20. (Unimplemented)
                01a40000
                0044
52 T000000b4 67ddc000      addr  badint,vex+72     # Interrupt 0x21. (Unimplemented)
                019a0000
                0048
53
54 T000000be 54a500c0      movb  $INIT,hpcctrl    # INITIALIZE command.
                fffc00
55 T000000c5 54a5005c0     movb  $5,hpcdata       # RTC value: interval of 50 milliseconds.
                fffe00
56
57 T000000cc 5cd8c000     movqb  $0,flags        # Clear interrupt flags.
                013e
58
59
60 T000000d2 04a614c0     .set   rtcflg,0        # Bit 0 means RTC interrupt detected.
                000139     .set   lcdflg,1        # Bit 1 means LCD interrupt detected.
                000139     movb  $20,rtcctr       # Clear RTC modulus counter (div by 20).
61 T000000d9 5fd8c000     movqd  $0,timcnt       # Clear seconds counter.
                0133
62
63
64 T000000df 7da30000     run:   bispsrw $0x800    # Enable interrupts from HPC.
65
66
67
                # Neither communication port is selected yet.

```

TL/DD/9976-60


```

68                               main:                # Put main program here.
69
70 T00000e3 4ec8a601      cbitb  $lcdflg,flags # Place cursor at first character of panel.
   c0000127 54a502c0
   fffc00
71 T00000eb 54a502c0      movb   $SEND_LCD,hpcctrl
   fffc00
72 T00000f2 54a500c0      movb   $0,hpcdata
   fffe00
73 T00000f9 54a501c0      movb   $1,hpcdata
   fffe00
74 T0000100 54a508c0      movb   $0x80,hpcdata
   fffe00
75 T0000107 f4a601c0      l1:    tbitb $lcdflg,flags
   000103
76 T000010e 9a79         bfc    l1
77
78 T0000110 4ec8a601      cbitb  $lcdflg,flags # Write initial value of zeroes.
   c00000fa 54a502c0
   fffc00
79 T0000118 54a502c0      movb   $SEND_LCD,hpcctrl
   fffc00
80 T000011f 54a5ffc0      movb   $0xFF,hpcdata
   fffe00
81 T0000126 54a508c0      movb   $8,hpcdata
   fffe00
82 T000012d 54a530c0      movb   $0x30,hpcdata
   fffe00
83 T0000134 54a530c0      movb   $0x30,hpcdata
   fffe00
84 T000013b 54a530c0      movb   $0x30,hpcdata
   fffe00
85 T0000142 54a530c0      movb   $0x30,hpcdata
   fffe00
86 T0000149 54a530c0      movb   $0x30,hpcdata
   fffe00
87 T0000150 54a530c0      movb   $0x30,hpcdata
   fffe00
88 T0000157 54a530c0      movb   $0x30,hpcdata
   fffe00
89 T000015e 54a530c0      movb   $0x30,hpcdata
   fffe00
90 T0000165 f4a601c0      l2:    tbitb $lcdflg,flags
   0000a5
91 T000016c 9a79         bfc    l2
92
93 T000016e f4a600c0      mainlp: tbitb $rtctrl,flags
   00009c
94 T0000175 9a79         bfc    mainlp
95 T0000177 4ec8a600      cbitb  $rtctrl,flags
   c0000093
96
97 T000017f 7ca101       bicpsrb $0x01      # Clear carry.
98 T0000182 4effa600     addpd  $0x01,timcnt # Increment BCD elapsed time.
   00001c0
   00008a
99

```

TL/DD/9976-61

The HPC as a Front-End Processor

National Semiconductor
Application Note 551
Brian Marley



AN-551

ABSTRACT

This application note covers the use of the National Semiconductor HPC46083 High-Performance microController as a front-end processor to collect and block data from RS-232 (serial) and Centronics (parallel) ports for a Host CPU (a typical application being an intelligent graphics-oriented printer). This application note builds on Application Note AN-550 (UPI Port); the result being a program that implements a versatile front-end processor for a National NS32CG16 CPU.

1.0 INTRODUCTION

In Application Note AN-550, "A Software Driver for the HPC Universal Peripheral Interface Port", we saw how a National Semiconductor HPC46083 microcontroller can be connected and programmed to perform intelligent peripheral functions for a host CPU; our example being an application connecting an NS32CG16 CPU through the HPC to a typical front panel.

In this application note, we will expand on the hardware and the driver software presented there in order to implement a very useful function for a high-performance microcontroller: that of a front-end processor for data collection. To demonstrate a real-world application for this kind of function, we implement here an intelligent interface to a Centronics-style parallel input port and an RS-232 serial port, typical of a graphics-oriented printer.

2.0 THE FRONT-END PROCESSOR FUNCTION

As systems start to support higher data rates, one of the ever-present challenges is to minimize the interrupt processing load on the CPU, which can become intolerable if the CPU must process each character received in a separate interrupt. Since the character transfer task is typically so simple (reading a character from an input port and placing it into a memory buffer), it is often the case that the unavoidable context switch time associated with the interrupt outweighs the time spent processing the input character. In addition, the communication task may not be the CPU's highest priority: for example, in band-style laser printers the CPU must keep up with the paper movement; it can neither rerun an image nor stop the paper. The communication rate therefore suffers; even printers running from a Centronics-style parallel port are typically unable to accept data faster than 4k characters per second.

The traditional technique for overcoming this obstacle is to implement Direct Memory Access (DMA) for the communication ports. This is, however, quite a large investment in hardware, requiring an external DMA controller chip and more sophisticated bus structures to support it. In other words, it may be acceptable for a computer system, but it is overly expensive for an embedded controller application. Also, the response time required of the CPU can still be stringent, especially in implementing flow control to pace the character rate from the external system presenting the data.

The HPC46083 microcontroller, however, allows a much more cost-effective approach to the problem. As a peripheral, it interfaces to the CPU much as any peripheral controller

would. In the application documented here, it buffers up to 128 characters before interrupting the CPU, thus dropping the CPU input interrupt processing frequency by over two orders of magnitude, while allowing a character input rate of over 20 kb/sec.

2.1 Data Transfer Technique

The benefit provided by a front-end processor is derived from the efficiency it adds to the process of getting data into the CPU's data buffer; that is, how much of the CPU's processing time gets dedicated to this task.

The efficiency is provided by two means:

1. Reduction of interrupt overhead. By interrupting the CPU only once every 100 characters, the overhead per character becomes virtually negligible.
2. Elimination of error testing overhead. If the CPU were communicating with a UART directly, it would have to poll for error conditions on each character. In our implementation, there are two interrupt vectors for data transfer: one for good data (which transfers a block of data), and one for bad data (which transfers one character and its error flags). The good data interrupt routine, then, which is invoked almost exclusively, contains a very simple inner loop. After reading the character count from the HPC, all that the CPU needs to do is:

- Move a character from the HPC's OBUF register to the current destination address. No time is wasted polling the HPC status; the hardware synchronization technique described in Application Note AN-550 handles this.
- Increment the destination address. (Checking against buffer limits could be done here, but is more efficiently handled outside the inner loop.)
- Decrement the character count and test it; loop if non-zero.

The HPC firmware also supports this technique by guaranteeing that the reporting of character errors (and BREAK conditions) is synchronized with good data, so that the CPU can tell exactly where in the data stream the error occurred.

2.2 Logic Replacement

Front-end processing tasks by no means use up the HPC's capabilities in a system. In our application, the HPC also serves as the CPU's only interrupt controller, allowing a large number of vectors with no additional hardware. It performs additional control tasks such as dynamic RAM refresh request timing, front panel control and real-time clock functions given in Application Note AN-550 with inexpensive interfacing. In a single 4 kbyte program developed in our group, we were also able to add an interface to an inexpensive serial EEPROM device (connected directly to the MICROWIRE/PLUSTM port of the HPC) and to a laser-printer engine for non-imaging control functions, and we also implemented a higher-resolution event timing feature. (These are topics for future application notes, however, and are not dealt with here.)

To summarize, then, the HPC not only can provide front-end processing functions, but can pay for itself by replacing other logic in the system.

3.0 HARDWARE

The following sections refer to the schematic pages included. We will discuss here only the portions involving the Centronics Parallel and RS-232 Serial ports. See Application Note AN-550 for details of the other connections shown (the UPI port and front-panel functions).

3.1 The Centronics Parallel Port

The Centronics port was implemented on the connector designated J5. Most of the interface is diagrammed on Sheet 4 of the schematic.

3.1.1 Control Inputs

Pin 1 of the J5 connector receives the Data Strobe (**STROBE**) input, which signals the presence of valid data from the external system. On Sheet 4, in area C5, this signal appears from the connector. It is filtered using a Schmitt trigger (a spare 1488 RS-232 receiver chip), and is then presented to the HPC (Sheet 3) as interrupt signal I4.

Pin 31 is the Input Prime signal (**PRIME**), which is asserted low by the external system in order to reset the interface. It appears on Sheet 4 in area D5, and is filtered in a similar manner. It is then gated with the signal **ENPRIME** from the Centronics Control Latch, and the resulting signal is presented to the HPC on pin *EXUI, which is the External UART Interrupt input. The gating is used to prevent confusion between **UART** and **PRIME** interrupts: while the Centronics port is selected, only **PRIME** causes interrupts, and while the RS-232 port is selected, this gating keeps **PRIME** interrupts from being asserted.

3.1.2 Data Inputs

Eight data bits, from J5 pins 2 through 9, appear in areas B8 and C8 of Sheet 4. They are latched into a 74LS374 latch on the leading edge of the **STROBE** signal (note the inversion through the Schmitt receiver on **STROBE**). The latch is enabled to present data to the HPC's Port D pins by the signal **ENCDATA**, which comes from HPC pin B12. Note that Port D is also used for inputting pushbutton switch data from a front panel.

3.1.3 Control Outputs

The Centronics control and handshake signals are presented by loading the Centronics Control Latch (Sheet 4, area B4) from the HPC's pins A8 through A15 (Port A Upper) using as a strobe the signal **CCTLCLK** from HPC pin P2.

Pin 10 of connector J5 is the Centronics Acknowledge (**CACK**) pulse, which is used to signal the external system that the HPC is ready for the next byte of data. This is one of the two handshake signals used to pace data flow. It is initialized high by the HPC, and is pulsed low when required.

Pin 11 is the Centronics Busy (**CBUSY**) signal, which is generated by the flip-flop on Sheet 4, area C3. It is set directly by a **STROBE** pulse, and is also loaded from the Centronics Control Latch whenever the HPC finishes reading a byte of data (rising edge of **ENCDATA**). This will clear **CBUSY** under normal conditions, allowing the external system to send another byte of data.

Five additional signals, whose functions vary significantly from printer to printer, are presented on connector J5 from the Centronics Control Latch. These are:

Pin 13, which generally indicates that the printer is selected.

Pin 12, which indicates that the printer needs attention (for example, that it is out of paper).

Pin 32, which indicates a more permanent or unusual problem (lamp check or paper jam).

Pins 33 and 35, which vary more widely in use.

These five pins are manipulated by commands from the CPU; the HPC simply presents them as commanded.

3.1.4 Other Signals

Pin 18 of the Centronics port connector receives a permanent +5V signal (area B2 of Sheet 4), and a set of other pins (middle of Sheet 2) are connected permanently to ground.

3.2 The RS-232 Serial Port

The serial port (on connector J6) makes use of the HPC's on-chip UART and baud rate generator; very little off-chip hardware is required. The entire RS-232 circuit appears on Sheet 3 of the schematic.

This port is implemented in a way typical of printers, and so there are no sophisticated handshaking connections. The interface looks like an RS-232 DTE device: Connector J6 pin 2 is transmitted data (out) and pin 3 is received data (in). The RS-232 data input appears in area B8 of Sheet 3, as signal **RXD**. After the RS-232 receiver, it is presented on the HPC's UART input pin (I6). Note that this pin can be monitored directly as a port bit; this enables the HPC to check periodically for the end of a **BREAK** condition without being subjected to a constant stream of interrupts for null characters.

The Data Set Ready signal (**DSR**) is received from pin 6 of J6, and presented on HPC pin I7, where it can be monitored by the HPC firmware.

The Request to Send signal (**RTS**) is a constant high level placed on J6 pin 4.

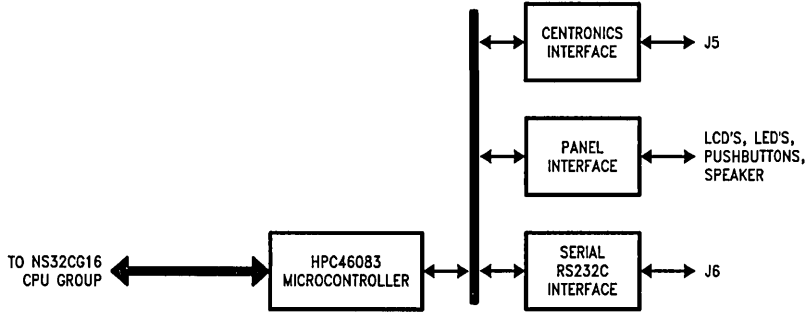
Transmitted data (**TXD**) is presented from the HPC's UART output pin (B0), through a buffering gate, to an RS-232 driver, and then out on J6 pin 3. The buffering gate would be unnecessary if the CMOS 14C88 driver were being used, but the gate was a spare and allowed cost savings using the less expensive TTL 1488 chip.

Data Terminal Ready (**DTR**) is simply presented from a programmable port pin of the HPC (pin B1). It is buffered through a spare inverter, and then presented to RS-232 connector J6 pin 20 through an RS-232 driver. As with the UART output, the buffering would be unnecessary with the 14C88 type of RS-232 driver; however, note that the HPC firmware would have to be modified slightly due to the resulting polarity difference on the pin.

J6 pins 1 (Frame Ground) and 7 (Signal Ground) are, of course, grounded, as shown in this sheet also.

3.3 Schematic Sheets

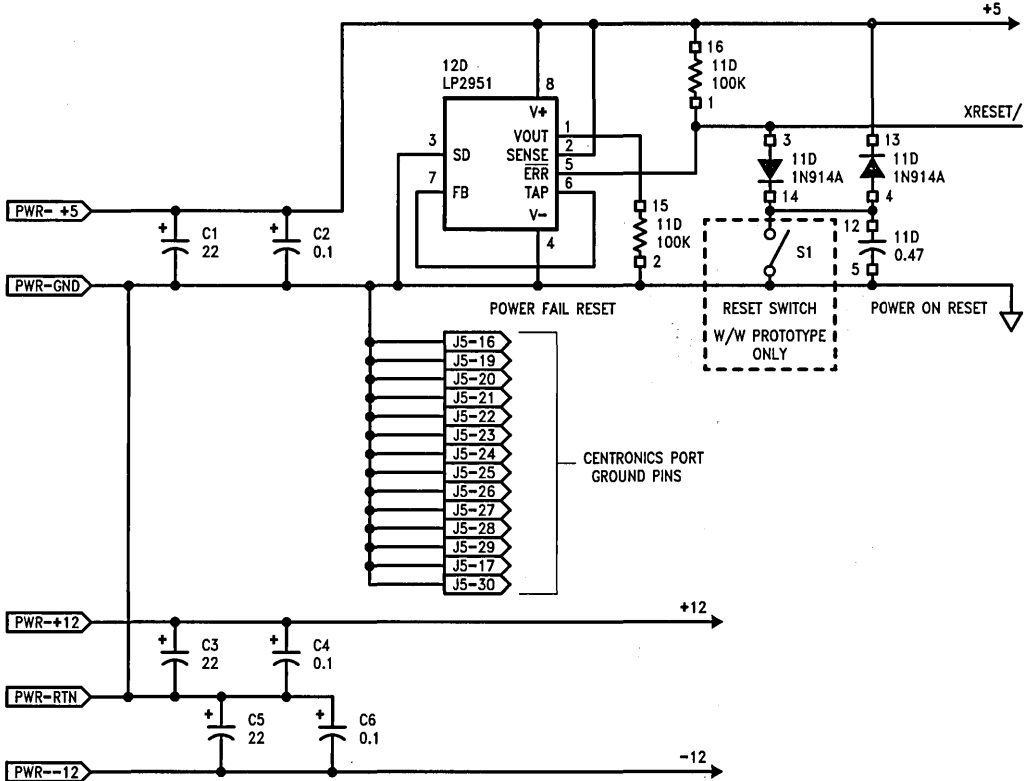
Sheet 1



TL/DD/9977-1

Power and Ground Distribution

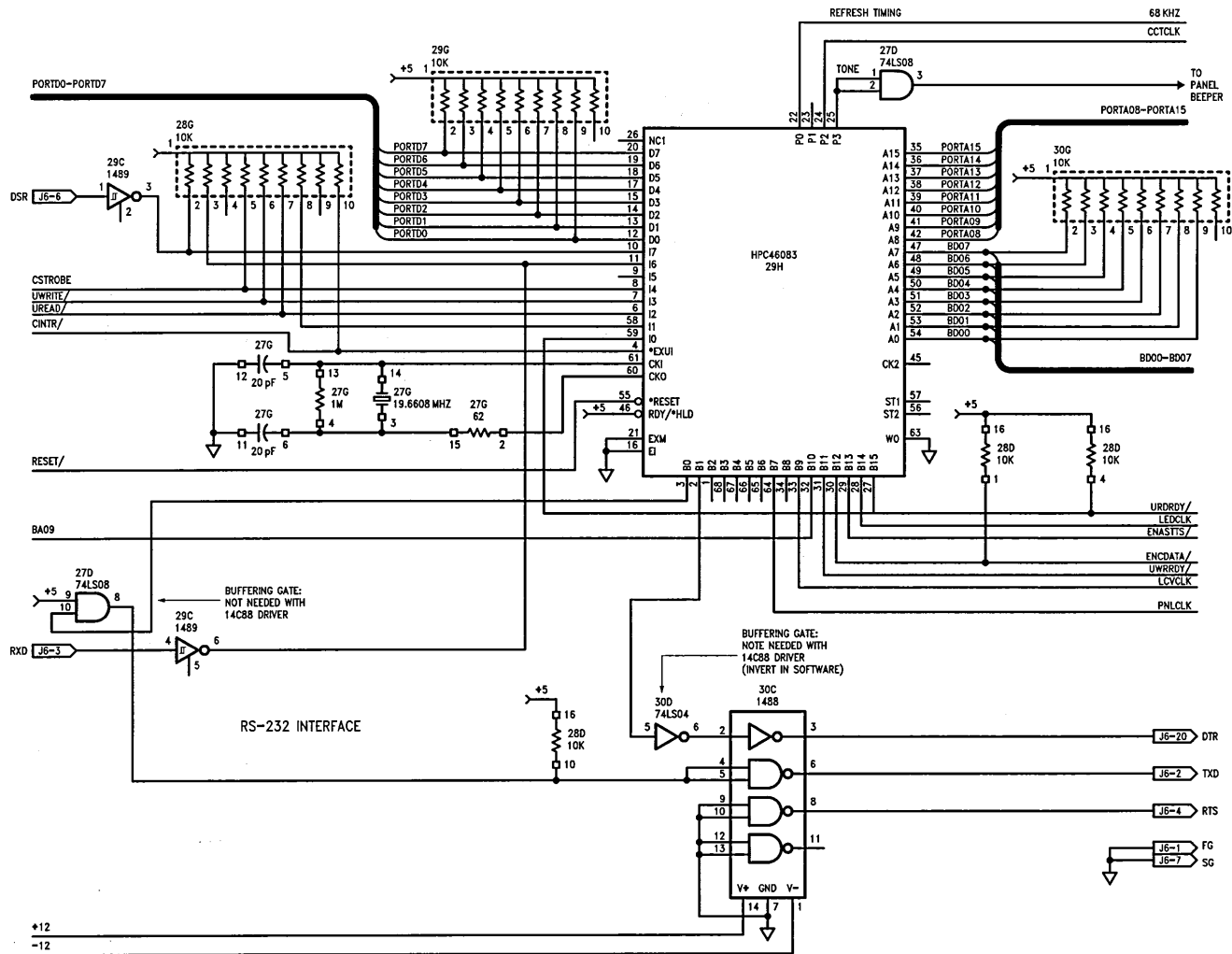
Sheet 2



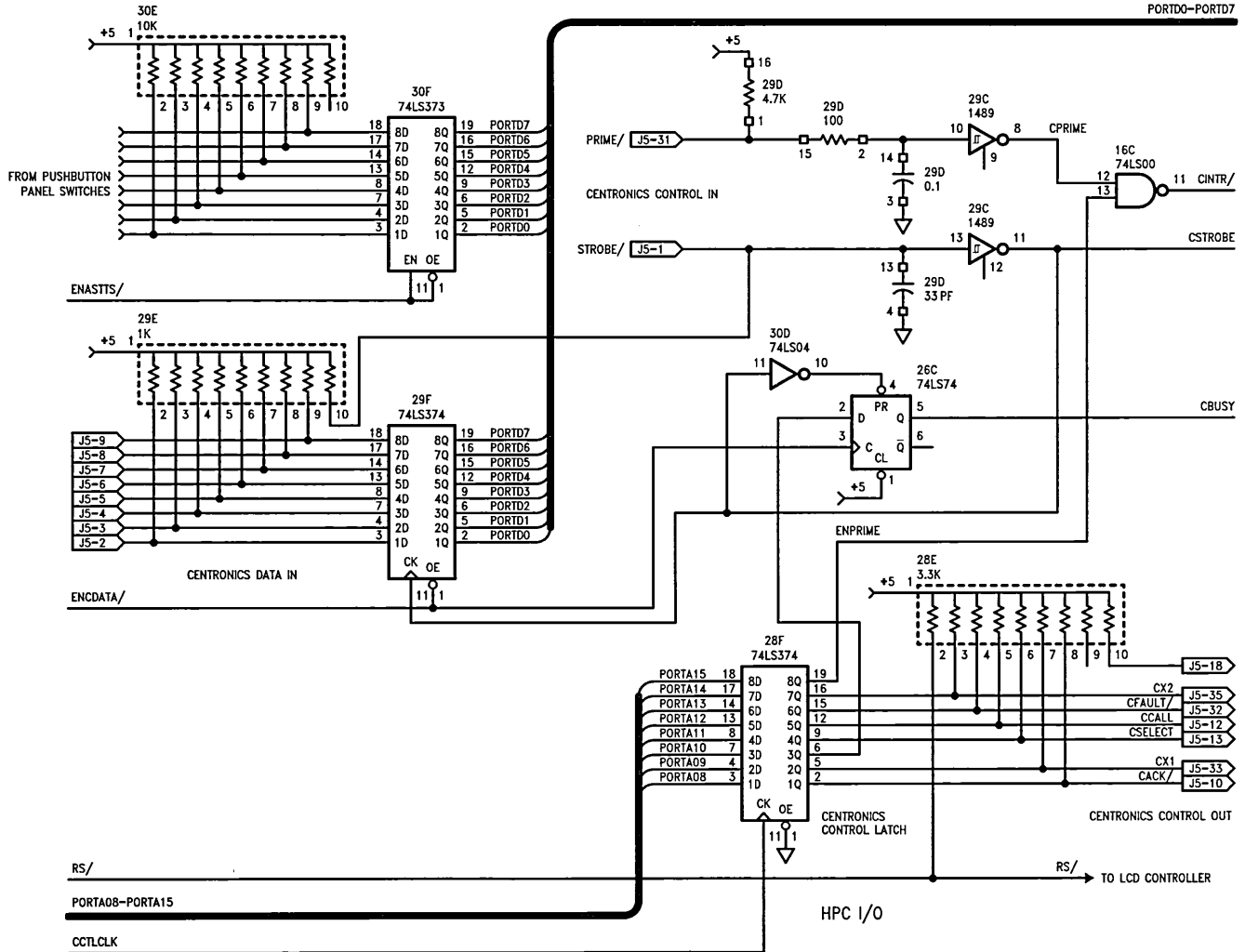
TL/DD/9977-2

Notes: (Unless otherwise specified)

- 1. All capacitance values in microfarads, 50V.
- 2. All resistor values in Ohms, 1/4W, 5%.

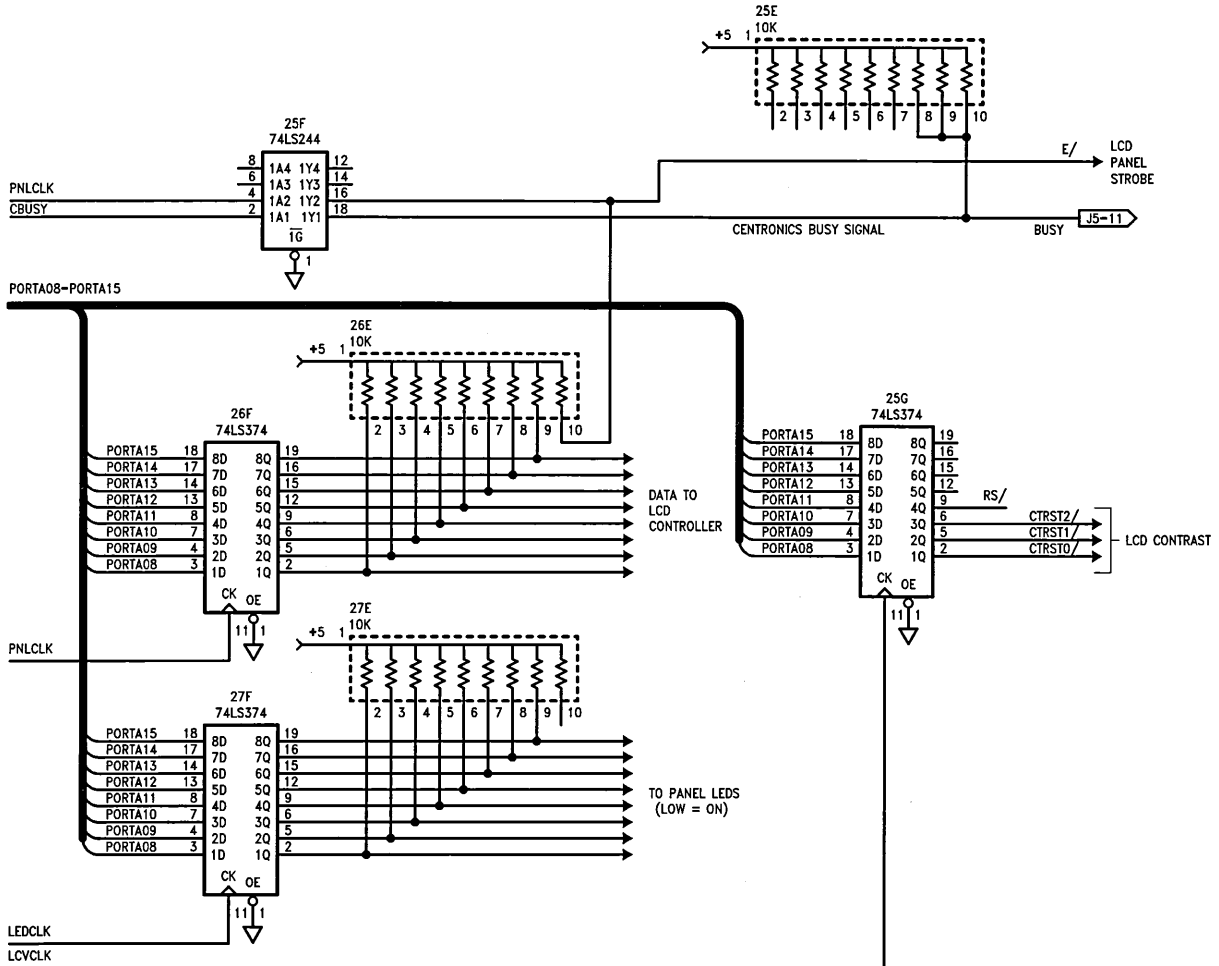


5-188



5-189





S-190

4.0 PROTOCOL

The command and interrupt protocol is a superset of that implemented for Application Note AN-550. The two commands SELECT-CENT and SELECT-UART are added to select and initialize each of the communication ports (Centronics or RS-232). The CPU can exercise control over data buffering by the commands FLUSH-BUF, CPU-BUSY, CPU-NOT-BUSY and SET-IFC-BUSY. It can set Centronics port error flags and status using SET-CENT-STS, and it can test for RS-232 status using the TEST-UART command. The HPC also allows the CPU to send characters out on the RS-232 port using the SEND-UART command.

New interrupts presented by the HPC are !DATA, which transfers up to 128 bytes of buffered data to the CPU, IPRIME and !UART-STATUS, which inform the CPU of port status changes, and !DATA-ERR, which reports in detail any error occurring in characters received. The interrupt !ACK-UART is presented to the CPU to acknowledge that the SEND-UART command has been completed.

Note that the command codes for the front panel functions have been changed. Their formats, however, have not changed, nor have their functions, except that the INITIALIZE command now performs a disconnection function on the RS-232 and Centronics ports.

4.1 Commands

The first byte (command code) is sent to address FFFC00, and any argument bytes are then written to address FFFE00. The CPU may poll the UPIC register at address FD0000 to determine when the HPC can receive the next byte, or it can simply attempt to write, in which case it will be held in Wait states until the HPC can receive it. Except where noted, the CPU may send commands continuously without waiting for acknowledgement interrupts from previous commands.

00 INITIALIZE

This command has two functions. The first INITIALIZE command after a hardware reset (or RESET-HPC command) enables the !RTC and !BUTTON-DATA interrupts. Both data communication ports are set to their "Busy" states until a "SELECT" command is sent. The INITIALIZE command may be re-issued by the CPU to de-select both communication ports, and to either start or stop the !RTC interrupts. There is one argument:

RTC-Interval: One-byte value. If zero, !RTC interrupts are disabled. Otherwise, the !RTC interrupts occur at the interval specified (in units of 10 ms per count).

01 SELECT-CENT

Select the Centronics port and set it ready, using the timing sequence specified by the supplied ACK-Mode argument. Data from the port is enabled, and the IPRIME interrupt is also enabled. Arguments:

ACK-Mode: one byte in the format:

x	x	x	x	x	L	Timing
---	---	---	---	---	---	--------

where the Timing field is encoded as:
00 = BUSY falling edge occurs after ACK pulse.

01 = BUSY falling edge occurs during ACK pulse.

10 = BUSY falling edge occurs before ACK pulse.

and the L bit, when set, requests Line Mode. It suppresses the removal of BUSY and the occurrence of the ACK pulse when the buffer is passed to the CPU. To fully implement Line Mode, this mode should be used with Pass-Count = 1 and Stop-Count = 1, and the CPU must use the SET-CENT-STS command to acknowledge each character itself.

Pass-Count: Number of characters in buffer before the HPC passes them automatically to CPU. One byte.

Stop-Count: Number of characters in buffer before HPC tells the external system to stop. One byte.

Note that the buffer is a maximum of 128 bytes in length, in this implementation.

Requires INITIALIZE command first.

Select Serial port and set it ready, according to supplied arguments. Requires INITIALIZE command first. Arguments are:

Baud: Baud rate selection. One Byte containing.

0 = 300 baud

1 = 600 baud

2 = 1200 baud

3 = 2400 baud

4 = 4800 baud

5 = 9600 baud

6 = 19200 baud

7 = 38400 baud

8 = 76800 baud

Frame: One byte, selecting character length, parity and number of stop bits.

02 SELECT-UART

Value	Data Bits	Parity	Stop Bits
0	8	Odd	1
1	8	Even	1
2	8	None	1
3	8	None	2
4	7	Odd	1
5	7	Even	1
6	7	Odd	2
7	7	Even	2

Flow: One byte, bit-encoded for handshaking and flow control modes:

0	0	0	0	XON	DTR	DSR	
7	6	5	4	3	2	1	0

DSR: 1 = the HPC disables the UART receiver while the DSR input is inactive.

DTR: Polarity of DTR output, and whether it is used as a flow-control handshake.

00 = Permanently low (negative voltage).

01 = Permanently high (positive voltage).

10 = Handshaking: low means ready.

11 = Handshaking: high means ready.

XON: 1 = the HPC performs XON/XOFF flow control.

Pass-Count: Number of characters in buffer before the HPC passes them automatically to CPU. One byte.

Stop-Count: Number of characters in buffer before HPC tells the external system to stop. One byte.

Note that the buffer is a maximum of 128 bytes in length, in this implementation.

Requires INITIALIZE command first.

03 (reserved)

04 FLUSH-BUF

No arguments. Flush HPC data communication buffer to CPU. Any data in the buffer is immediately sent to the CPU (using the !DATA interrupt). This command triggers the !DATA interrupt only if the buffer contains at least one byte. Requires INITIALIZE command and SELECT command first.

05 CPU-BUSY

No arguments. Indicates that the CPU cannot accept any more data (the CPU's data buffer is full). This suppresses the !DATA and !DATA-ERR interrupts. Requires INITIALIZE command and SELECT command first.

06 CPU-NOT-BUSY

No arguments. This undoes a previous CPU-BUSY command, and indicates that the CPU can now accept more data from the HPC. Requires INITIALIZE command and SELECT command first.

07 SET-IFC-BUSY

"Set Interface Busy". No arguments. Commands the HPC to immediately signal the external system to stop

sending characters. This status is removed only by performing a SELECT command. Requires INITIALIZE command and SELECT command first.

08 SET-CENT-STS

"Set Centronics Port Status". Loads Centronics latch from the supplied argument byte. Argument is eight bits, which must be encoded as follows:

ENPRIME	CX2	FAULT	CALL	SELECT	BUSY	CX1	ACK
---------	-----	-------	------	--------	------	-----	-----

The $\overline{\text{ACK}}$ bit should always be a "1". The CPU must use the BUSY bit to generate an $\overline{\text{ACK}}$ pulse: if the BUSY bit is zero, the $\overline{\text{ACK}}$ signal will be automatically pulsed low, then high, (regardless of the previous states of BUSY and $\overline{\text{ACK}}$).

Requires INITIALIZE command and SELECT-CENT command first.

09 SET-CONTRAST

The single argument is a 3-bit number specifying a contrast level for the LCD panel (0 is least contrast, 7 is highest contrast). There is no response interrupt. Does not require INITIALIZE command first.

0A SEND-LCD

This writes a string of up to 8 bytes to the LCD panel. Arguments are:

flags: A single byte, containing the RS bit associated with each byte of data. The first byte's RS value is in the least-significant bit of the FLAGS byte.

#bytes: The number of bytes to be written to the LCD display.

byte[1]-byte[#bytes]: The data bytes themselves.

The HPC determines the proper delay timing required for command bytes (RS = 0) from their encodings. This is either 4.9 ms or 120 μ s.

The response from the HPC is the !ACK-SEND-LCD interrupt, and this command must not be repeated until the interrupt is received. This command does not require an INITIALIZE command first.

0B SEND-LED

The single argument is a byte containing a "1" in each position for which an LED should be lit.

There is no response interrupt, and this command does not require the INITIALIZE command first.

0C BEEP

No arguments. This beeps the panel for approximately one second. No response interrupt. If a new BEEP command is issued during the beep, no error occurs (the buzzer tone is extended to one second beyond the most recent command). Does not require INITIALIZE command first.

0D SEND-UART	The single one-byte argument is sent on the UART port. An acknowledgement interrupt !ACK-UART occurs on completion. This command must not be repeated until the interrupt is received. Requires INITIALIZE and SELECT-UART commands first.	Vector 00–0F (none)	(Reserved for CPU internal traps and the NMI interrupt.)
0E TEST-UART	Triggers a !UART-STATUS interrupt. This command must not be repeated until the interrupt is received. No arguments. Requires INITIALIZE and SELECT-UART commands first.	10 !DATA	Buffer data is being transferred to CPU. This will happen either automatically, at a point defined by the most recent SELECT command, or as the result of a FLUSH-BUF command. It is followed by a one-byte Length (number of characters: current HPC firmware has a range of 1–128), then that number of characters. Enabled by SELECT command after at least one INITIALIZE command.
A5 RESET-HPC	Resets the HPC if it is written to address FFFC00. It may be written at any time that the UPI port is ready for input; it will automatically cancel any partially-entered command. The CPU's Maskable Interrupt must be disabled before issuing this command. After issuing this command, the CPU should first poll the UPIC register at address FD0000 to see that the HPC has input the command (the least-significant bit [Write Ready] is zero). It must then wait for at least 25 μ s, then read a byte from address FFFE00. The HPC now begins its internal re-initialization. The CPU must wait for at least 80 μ s to allow the HPC to re-initialize the UPI port. Since part of the RESET procedure causes Ports A and B to float briefly (this includes the CPU's Maskable Interrupt input pin), the CPU should keep its maskable interrupt disabled during this time. It also must not enter a command byte during this time because the byte may be lost.	11 !RTC	Real-Time Clock Interrupt. No data returned. Enabled by INITIALIZE command if interval value supplied is non-zero. Note: This version of HPC firmware issues a non-fatal !DIAG interrupt if the CPU fails to service each !RTC interrupt before the next one becomes pending.
		12 (reserved)	
		13 !PRIME	Centronics INPUT PRIME signal has become active. No data returned. Enabled by SELECT-CENT command after at least one INITIALIZE command.
		14 (reserved)	
		15 (reserved)	
		16 (reserved)	
		17 !ACK-SEND-LCD	This is the response to the SEND-LCD command, to acknowledge that data has all been written to Panel LCD display. No other data is provided with this interrupt. Always enabled, but occurs only in response to a SEND-LCD command.
		18 !BUTTON-DATA	Pushbutton status has changed: one or more buttons have been either pressed or released. The new status of the switches is reported in a data byte, encoded as follows: Any pushbutton that is depressed is presented as a "1". All other bit positions, including unused positions, are zeroes. The pushbuttons are debounced before being reported to the CPU. This interrupt is enabled by the first INITIALIZE command after a reset.

4.2 Interrupts

The HPC interrupts the CPU, and provides the following values as the interrupt vectors for the CPU hardware. The CPU then reads data from the HPC at address FFFE00. All data provided by the HPC must be read by the CPU before returning from the interrupt service routine, otherwise the HPC would either hang or generate a false interrupt. The CPU may poll the UPIC register at address FD0000 to determine when each data byte is ready, or it may simply attempt to read from address FFFE00, and it will be held in Wait states until the data is provided by the HPC.

Note: All CPU interrupt service routines, including the NMI interrupt routines, must return using the "RETT 0" instruction. Do NOT use "RETI".

19 IUART-STATUS UART status has changed. This interrupt occurs only while the UART is selected. A data byte shows the UART's new state:

Bit	Condition
0 (LSB)	New state of DSR signal. This causes an interrupt only if DSR monitoring was requested in the last SELECT-UART command. The UART receiver is automatically enabled and disabled by the HPC, so no CPU action is required on receiving this interrupt. If a SELECT-UART command is entered, requesting DSR monitoring, and DSR is inactive, a IUART-STATUS interrupt occurs immediately.
1	This bit is set if a UART BREAK has just ended.
2-7	(unused)

Note 1: If the CPU has issued a CPU-NOT-READY command, this BREAK interrupt may be seen before the IDATA-ERR interrupt that announces the start of the BREAK (and its position in the data stream).

Note 2: The DSR and UART input (BREAK) signals are sampled every 10 ms.

1A IDATA-ERR

An error has been encountered in data coming from the currently-selected communication port. It is enabled by the first SELECT command after the first INITIALIZE command. Two data bytes are returned:

errchr: One byte containing the character on which the error was seen (this character is NOT placed in the data buffer).

errfgs: Error flags, detailing the error seen:

Bit	Error Seen
0 (LSB)	(unassigned)
1	(unassigned)
2	UART BREAK condition detected. This may be preceded by one or two framing errors.
3	Error Overflow: More errors occurred than HPC could report (the HPC has no FIFO for error reporting).
4	Buffer Overflow: Flow control failed to stop the external system, and the buffer overflowed.

5	Parity Error: Serial Port only.
6	Framing Error: Serial Port only.
7 (MSB)	Data Overrun: Serial Port only.

If bit 2, 3 or 4 is set, the communication port has been automatically shut down by the HPC. The CPU must issue a new SELECT command to re-enable the port.

When a character is received with an error, all characters appearing before it in the buffer are automatically flushed before this interrupt occurs. This is done to preserve the error character's position in the data stream. If the CPU decides to ignore the presence of an error, the character may be simply appended by the CPU to the data already in its data buffer. Please note: If the CPU has issued a CPU-NOT-READY command, the flush cannot occur, and this interrupt will not be issued until the flush has occurred.

1B IACK-UART

A CPU character has been sent on the UART, and the UART is ready for another. No data is returned with this interrupt. It is always enabled, but occurs only in response to the SEND-UART command.

1C (reserved)

1D IDIAG

Diagnostic Interrupt. This interrupt is used to report failure conditions and CPU command errors. There are five data bytes passed by this interrupt:

Severity
Error Code
Data in Error (passed, but contents not defined)
Current Command (passed, but contents not defined)
Command Status (passed, but contents not defined)

The Severity byte contains one bit for each severity level, as follows:

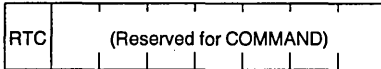
X	X	X	F	X	X	C	N
---	---	---	---	---	---	---	---

N (Note): least severe. The CPU missed an event; currently only the IRTC interrupt will cause this.

C (Command): medium severity. Not currently implemented. Any command error is now treated as a FATAL error (below).

F (Fatal): highest severity. The HPC has recognized a non-recoverable error. It must be reset before the CPU may re-enable its Maskable Interrupt. In this case, the remaining data bytes may be read by the CPU, but they will all contain the value 1D (hexadecimal). The CPU must issue a RESET command, or wait for a hardware reset. See below for the procedure for FATAL error recovery.

The Error Code byte contains, for non-FATAL errors, a more specific indication of the error condition:



RTC = Real-Time Clock overrun: CPU did not acknowledge the RTC interrupt before two had occurred.

The other bits are reserved for details of Command errors, and are not implemented at this time.

The remaining 3 bytes are not yet defined, but are intended to provide details of the HPC's status when an illegal command is received.

Note: Except in the FATAL case, all 5 bytes provided by the HPC *must* be read by the CPU, regardless of the specific cause of the error.

Fatal Error Recovery:

When the HPC signals a IDIAG error with FATAL severity, the CPU may use the following procedure to recover:

1. Write the RESET command (A5 hex) to the HPC at address FFFC00.
2. By inspecting the UPIC register at address FD0000, wait for the HPC to read the command (the WRRDY bit will go low).
3. Wait an additional 25 μ s.

4. Read from address FFFE00. This will clear the OBUF register and reset the Read Ready status of the UPI port. The HPC will guarantee that a byte of data is present; it is not necessary to poll the UPIC register. This step is necessary because only a hardware reset will clear the Read Ready indication otherwise (HPC firmware cannot clear it).

5. Wait at least 80 μ s. This gives the HPC enough time to re-initialize the UPI port.

6. After Step 5 has been completed, the CPU may re-enable the Maskable Interrupt and start issuing commands. Since the HPC is still performing initialization, however, the first command may sit in the UPI IBUF register or a few milliseconds before the HPC starts to process it.

5.0 SOURCE LISTINGS AND COMMENTARY

5.1 HPC Firmware Guide

This section is intended to provide help in following the flow of the HPC firmware. Discussion of features already documented in Application Note AN-550 are abbreviated here; see that application note for details.

The firmware for the HPC is almost completely interrupt-driven. The main program's role is to poll mailboxes that are maintained by the interrupt service routines, and to send an interrupt to the CPU whenever a HPC interrupt routine requests one in its mailbox.

On reset, the HPC firmware begins at the label "start". However, the first routine appearing in ROM is the Fatal Error routine. This is done for ease of breakpointing, to keep this routine at a constant address as changes are made elsewhere in the firmware.

5.1.1 Fatal Error Routine

At the beginning of the ROM is a routine (label "hangup") that is called when a fatal error is detected by the HPC. This routine is identical to that documented in Application Note AN-550.

5.1.2 Initialization

At label "start", entered on a Reset signal or by the RESET-HPC command from the CPU, the HPC begins its internal initialization. It loads the PSW register (to select 1 Wait state), and then (at label "srfs"), it starts the Refresh clock pulses running for the dynamic RAM by initializing Timer T4 and starting it.

At "supi", the UPI port is initialized for transfers between the HPC and the CPU.

At label "sram", all RAM within the HPC is initialized to zero.

At "sskint", the stack pointer is initialized to point to the upper bank of on-chip RAM (at address 01C0). The address of the fatal error routine "hangup" is then pushed, so that it will be called if the stack underflows.

At "tminit", the timers T1-T3 are stopped and any interrupts pending from timers T0-T3 are cleared. This step arbitrarily initializes the UART baud rate to 9600, but this selection has no effect.

At "scent", the Centronics port is initialized and set up to appear busy to the external system.

At "suart", the HPC UART is initialized for serial data from the external system. The RS-232 DTR signal is arbitrarily set low, which generally means that the printer is not ready. The state of DTR is not actually valid until the first SELECT-UART command is received, which selects the handshaking mode.

At "sled", the LED control signals are initialized, and all LED indicators are turned off.

At "stmrs", all timers are loaded with their initial values, and timers T5-T7 are stopped and any interrupts pending from them are cleared.

At "slcd", the LCD display is initialized to a default contrast level of 5, then commands are sent to initialize it to 8-bit, 2-line mode, with the cursor visible and moving to the right by default. This section calls a subroutine "wrpn!" for each character; the subroutine simply writes the character in the accumulator out to the LCD display and waits for approximately 10 ms.

The program then continues to label "minit", which initializes the variables in the HPC's on-chip RAM to their proper contents.

At label "runsys", the necessary interrupts are enabled (from the timers, and from pin I3, which is the UPI port interrupt from the CPU), and the program exits to the Main Program at label "mainlp". Interrupts from the Centronics and UART ports are not enabled until the appropriate SELECT command is received.

5.1.3 Main Program (UPI Port Output to CPU)

The Main Program is the portion of the HPC firmware that runs with interrupts enabled. It consists of a scanning loop at label "mainlp" and a set of subroutines (explained below). It is responsible for interrupting the CPU and passing data to it; the HPC is allowed to write data to the CPU only after interrupting it. Unlike the simpler UPI/Front Panel interface described in Application Note AN-550, this main loop scans two separate variables in on-chip RAM that are set up by interrupt service routines: a word called "alert", and a byte called "bstat" (for "Buffer Status"). Both variables are used to determine whether any conditions exist that should cause an interrupt to the CPU.

The "alert" word contains one bit for each interrupt that the HPC can generate. If a bit is set (by an interrupt service routine), the Main Program jumps to an appropriate subroutine to notify the CPU. The subroutine checks whether the UPI interface's OBUF register is empty, and if not, it waits (by calling the subroutine "rdwait"). It then writes the vector number to the OBUF register. This has the effect of interrupting the CPU (because the pin \overline{URDRDY} goes low), and the CPU hardware reads the vector from the OBUF register.

If there is more information to give to the CPU, the HPC places it, one byte at a time, into the OBUF register, waiting each time for OBUF to be emptied by the CPU. This technique assumes that the CPU remains in the interrupt service routine until all data has been transferred: if the CPU were to return from interrupt service too early, the next byte of data given to it would cause another interrupt, with an incorrect vector.

(Note, however, that the CPU may be interrupted with a Non-Maskable interrupt from a separate source. This simply inserts a pause into the process of reading data from the HPC. Since the HPC is running its main program at this point, with interrupts still enabled, it will not lose data from its communication port under these circumstances.)

The "bstat" byte represents a special case involving the interrupt IDATA to the CPU. This byte shows the main program whether the data communication buffer (which holds data from the external system) is full enough to send its contents to the CPU. If so, the main program calls the subroutine "snddta", which interrupts the CPU, then sends one data byte containing the number of characters to be transferred (currently as many as 128 are possible), and then the characters themselves.

The CPU may, at any time, demand that the HPC transfer all characters that are within its communication buffer. (This is called a "flush" command, which sets one of the bits of the "alert" word, described above.) The HPC, in response, will empty the buffer to the CPU with a IDATA interrupt, even if only one character is left. If the buffer is completely empty, however, the flush command is ignored.

Subroutines called from the Main Program loop are:

- sndrtc: sends a Real-Time Clock interrupt to the CPU. No data is transferred; only the interrupt vector.
- sndlak: interrupts the CPU to acknowledge that a string of data (from a SEND-LCD command) has been written to the LCD display. No data is transferred for this interrupt.
- sndbtn: interrupts the CPU to inform it that a pushbutton has been pressed or released. A data byte is transferred from variable "swlsnt", which shows the new states of all the pushbuttons.
- sndfsh: performs a Flush operation. If there is data, it jumps to the "snddta" routine to send the contents of the buffer to the CPU. If there is no data, however, this subroutine simply returns without generating an interrupt.
- snddta: sends data from the communication buffer to the CPU. It may be entered for one of three reasons:
 1. the communication buffer is full enough that it must be sent automatically to the CPU.
 2. a Flush command has been received from the CPU. (The bit "aflush" in the ALERT word is set.)
 3. an error has been detected on a character received from the external system. This causes an internal Flush request, so that all good characters are sent to the CPU before the bad character is reported. This case is also different because it does not flush the entire buffer, but only up to the point of the error. The limit is held in the variable "fshlim".

The subroutine sends a "length" byte (from variable "numout", sampled from "numchr", which is maintained by the communication interrupt routines). This indicates how many characters will be transferred. The subroutine next sends the characters themselves. It then updates the buffer status variables in on-chip RAM, to indicate how many characters were removed.

Depending on other status of the selected communication port, this subroutine may re-enable communication on the port if it was stopped (for example, if the buffer was too full to accept more data until the "snddta" routine emptied it). This is done at label "sdstp".

- sndprm: interrupts the CPU because the INPUT PRIME signal on the Centronics parallel port was activated by the external system. No data is transferred by this interrupt.
- sndust: interrupts the CPU to report a change in UART status. This interrupt may also be triggered by the CPU using the TEST-UART command.

snderr: interrupts the CPU to inform it that a character with an error was received. The character and a byte containing error flags are transferred to the CPU.

snduak: interrupts the CPU in response to a SEND-UART command, to acknowledge that the requested character has been sent on the UART transmitter, and that it is ready to transmit another character.

sndiag: interrupts the CPU to inform it of a !DIAG interrupt condition, when it is of NOTE severity. (Other !DIAG conditions are handled at label "hangup".)

5.1.4 UPI Port Input from CPU (Interrupt I3)

This interrupt service routine, at label "upiw", accepts commands from the CPU. Apart from the existence of additional commands, the structure of this routine is identical to that of Application Note AN-550. We document here the labels and functions involved in this larger application.

Command Processing Routines

INITIALIZE	I3 interrupt labels:	State 1 = fcinit	State 3 = lcinlt
SELECT-CENT	I3 interrupt labels:	State 1 = fcsele	State 3 = lcselc
SELECT-UART	I3 interrupt labels:	State 1 = fcselu	State 3 = lcselu
FLUSH-BUF	I3 interrupt labels:	State 1 = fcflsh	State 3 = (none)
	At label "fcflsh", the "alert" word bit "aflush" is set, which requests that the main program flush the communication buffer.		
CPU-BUSY	I3 interrupt labels:	State 1 = fcbsy	State 3 = (none)
	At label "fcbsy", the buffer status byte "bstat" is set to indicate that the CPU is busy and cannot accept more data from the HPC. This disables the !DATA interrupt.		
CPU-NOT-BUSY	I3 interrupt labels:	State 1 = fcnbby	State 3 = (none)
	At label "fcnbby", the buffer status byte "bstat" is set to indicate that the CPU is ready to accept more data from the HPC. The !DATA interrupt is re-enabled.		
SET-IFC-BUSY	I3 interrupt labels:	State 1 = fcifby	State 3 = (none)
	At label "fcifby", the currently selected interface is set busy, in order to present an error indication.		
SET-CENT-STS	I3 interrupt labels:	State 1 = fcscst	State 3 = lcsrst
	At label "lcsrst", the Centronics Port status byte "cps" is loaded from the value supplied by the CPU, and the Centronics port control signals are updated to reflect these new settings. The subroutine "setcen" is used to set up the control signals, and it also pulses the Centronics ACK signal if appropriate.		
SET-CONTRAST	I3 interrupt labels:	State 1 = fcslcv	State 3 = lcslcv
	At label "lcslcv" (Set LCD Voltage), the LCD Contrast latch is loaded from the value supplied by the CPU.		
SEND-LCD	I3 interrupt labels:	State 1 = fcslcd	State 3 = lcslcd
	This command sends a string of up to eight bytes to the LCD display. Application Note AN-550 describes the implementation of this command in detail.		
SEND-LED	I3 interrupt labels:	State 1 = fcslcd	State 3 = lcsled
	At label "lcsled", the byte provided by the CPU is written to the LED latch.		
BEEP	I3 interrupt labels:	State 1 = fcbeep	State 3 = (none)
	This command sends a one-second beep tone to a speaker.		
SEND-UART	I3 interrupt labels:	State 1 = fcsndu	State 3 = lcsndu
	At label "lcsndu", the single argument (the character to be sent) is placed in variable "uschr", and the bit "schr" is set in variable "ups" (UART Port Status). By doing this, the character has been queued for transmission. The transmission is performed by the subroutine at label "setuar", which is also responsible for performing the XON/XOFF flow control protocol. If a character is already being sent (the transmitter interrupt is enabled), then this is the only action required, since the transmitter interrupt automatically invokes the "setuar" subroutine. However, if the transmitter is idle, this routine must itself call "setuar" to transmit the character.		
	The subroutine "setuar" itself calls another subroutine at label "uecsnd", which formats the character to be transmitted into the frame selected by the current UART framing mode. It then sends the character. Note that the UART framing mode applies to output as well as input characters.		
TEST-UART	I3 interrupt labels:	State 1 = fcusts	State 3 = (none)
	At label "fcusts", the HPC sets the "austat" bit of the ALERT word, requesting the Main Program to send a !UART-STATUS interrupt to the CPU.		

5.1.5 Centronics Communication

This task is triggered by each edge of the Centronics port STROBE signal. This signal is detected by the HPC on the I4 interrupt line. On the leading edge of STROBE, the character is input to the data communication buffer. This edge also sets the BUSY signal, by hardware action. On the trailing edge, the BUSY flag is affected by the HPC firmware. If the HPC is ready to receive more characters, the BUSY signal is cleared and the ACK signal is pulsed. If the HPC is not ready to receive more data, it leaves the BUSY signal high, which prevents the external system from sending more characters.

The Centronics port STROBE handler is at label "cenint". It first determines whether a falling or rising edge was detected on the STROBE signal. If the leading (falling) edge was detected, then it jumps to label "cstrbl"; otherwise it jumps to label "cstrbt" to process a trailing edge.

At label "cstrbl", the character is placed in the next available position of the communication buffer, if the buffer is not already full. (If it is already full, then it is processed as an error, as discussed below.) Then some tests are performed:

If the buffer is not full enough to pass data to the CPU, then the routine exits by jumping to label "cenlex", where it prepares to detect the trailing edge of STROBE. Otherwise, it sets the "pass" bit in the variable "bstat", which requests the main program to send data to the CPU, and then it continues.

If the buffer is not full enough to tell the external system to stop sending characters, then the routine exits by jumping to "cenlex". Otherwise, it sets the "stop" bit in variable "bstat", indicating that the external system has been stopped, and it also sets the "cbusy" flag in variable "cps", which will prevent the Centronics BUSY and ACK signals from being changed when the STROBE pulse ends. The routine continues.

If the buffer has become completely full, then the "full" bit in "bstat" is set, indicating that any more characters received will trigger an error. Character processing then continues at label "cenlex".

At "cenlex", the Centronics Control Latch is set (temporarily) to force the BUSY signal high, because it should not become low until the STROBE pulse ends. The I4 pin, which detects the STROBE signal, is then re-programmed to detect the trailing edge (rising edge at the Centronics connector, but falling edge at pin I4 due to an inverting buffer). If the trailing edge already has occurred, then this reprogramming will set another interrupt pending immediately. There is, however, a possibility that the strobe edge could occur simultaneously with the reprogramming, with unknown results. For this reason, the STROBE signal is sampled by the firmware, and if the pulse has already completed, then instead of returning from the interrupt it jumps immediately to interrupt routine "cstrbt", which processes the trailing edge.

The code at label "cstrbt" is entered whenever either a trailing edge interrupt is detected on pin I4 (STROBE), or the leading edge interrupt routine jumps to it. It reprograms the I4 pin to detect a leading edge again, clears the I4 interrupt

(which is automatically cleared only on interrupt service), then jumps to the "setcen" subroutine, which manipulates the BUSY and ACK signals appropriately, according to the contents of the "cps" variable and the selected ACK timing mode in variable "ackmd".

5.1.5.1 Centronics Error Handling

A buffer overrun error is processed at label "cenerr". This is the only kind of character error that can happen on a Centronics interface, and it would be due to an incorrect connection or a software error.

For internal firmware debugging purposes, the "cps" variable bit "cbusy" is again set to ensure that the Centronics interface will keep the BUSY signal set.

If an error is already waiting to be reported (bit "aerr" of variable "alert" is already set), then this is a "multiple error" condition, and cannot be fully reported. Instead, at label "cenmer", the bit "errorfl" in variable "errfgrs" is set. This variable is sent to the CPU when the error is reported. Also, the I4 interrupt is disabled, to prevent any further STROBE interrupts until a new SELECT-CENT command is received from the CPU.

If no error is waiting to be reported, then bit "aerr" of variable "alert" is set, requesting the main program to generate an IERROR interrupt to the CPU. Further data is provided to be passed to the CPU:

variable "errfgrs" is initialized to indicate only a buffer overrun error.

variable "errchr" is loaded with the character that was received and could not fit in the buffer.

Because the received character is reported with the error interrupt, and because no data is lost yet, the Centronics port is not disabled by this condition.

5.1.6 UART Communication

UART communication is performed by the UART interrupt routine at label "uarint". After pushing the required registers onto the stack, the routine determines which interface is selected. If it is the Centronics port, the only cause of the interrupt is the INPUT PRIME signal, and the HPC jumps to label "uarprm" (see Background Processing/Monitoring Tasks, below). If the UART port is selected, then it is due to either a receiver or a transmitter interrupt (and the INPUT PRIME is gated so that it cannot be presented).

5.1.6.1 UART Output

At label "uarout", a transmitter interrupt has been received. If the bit "icpu" in variable "ups" is set, this means that the character just transmitted was a character sent by a CPU SEND-UART command, and the CPU is notified by requesting the !ACK-UART interrupt from the Main Program.

The subroutine "setuar" is now called, to determine whether any more characters need to be sent, either for XON/XOFF handshaking or because the CPU has requested the HPC to send another character. If so, another character is sent by "setuar", and the UART transmitter interrupt remains enabled. If not, the "setuar" routine disables the transmitter interrupt.

5.1.6.2 UART Input

At label "uartin", an interrupt has been generated by the UART receiver. This means that a character is available to be placed into the Communication Buffer.

The first action taken by the HPC is to read the receiver status register ENUR (which contains the 9th data bit and the Data Overrun and Framing Error error flags), then it reads the character itself from the RBUF register. The ENUR register is saved temporarily in variable "enring" for future processing, but is also held in the Accumulator, which is used here to "accumulate" error flags. The HPC then prepares to check for a parity error.

Parity checking is not a hardware feature of the HPC's UART, so a bit-table lookup is performed using the "X,[B].b" addressing mode of the IFBIT instruction. This addressing mode is similar to NS32000 bit addressing, in that it allows one to address up to 64 kbits (addressed from the contents of the X register) from a base address given in the B register. By placing the character to be checked into the X register, and pointing the B register at a properly constructed table (labels "evntbl" and "oddtbl"), a parity error can be detected in a single IFBIT instruction (see for example label "u8dopr").

After loading the X and B registers, a multi-way branch is performed (jid), which branches to one of 8 labels depending on the character framing mode variable "uframe" (which is loaded by the SELECT-UART command). Each mode handles parity differently: labels "uiod8" and "uiev8" check for odd or even parity, respectively, including 9 character bits (8 data plus 1 parity) to make the test. Labels "uiod7" and "uiev7" include only 8 bits (7 data plus 1 parity). Label "nopar" handles the cases where no parity is included in the character frame. Also within these routines, a decision is made whether a Framing Error seen in the character is also a Break condition: if two consecutive characters are seen with framing errors with all zeroes in their parity and data fields, then the second character is reported as a Break character as well as having a framing error. If, at label "uinpok", no errors have been flagged in the Accumulator, the routine branches to label "uingd" to place the character into the Data Communication Buffer for the CPU. If errors have been discovered, then the character is instead reported to the CPU using the !DATA-ERR at label "uiner".

The "uingd" portion of this routine is very similar to the portion of the Centronics input routine that places characters into the buffer for the CPU. A different mechanism is used for flow control, of course, to stop the external system if the buffer becomes full.

At label "uiner", a check is made to determine whether the CPU has received the last character error reported. If not, this is a "multiple error" condition, handled at label "uinmce". If so, then this is reported as a new error at label "uin1ce". The error character and its error flags are provided to the Main Program in the mailboxes "errchr" and "errfgs", and the bit "aerr" in variable "alert" is set to request that a !DATA-ERR interrupt be sent to the CPU.

On a multiple-error condition, the new error flags are ORed with the old ones, handshaking is used to stop the external

host system from sending more characters, and the UART receiver is automatically disabled. The CPU must issue a new SELECT-UART command to re-enable it.

Another pair of routines report an error if the buffer overflows. This error is reported at label "uin1e" if no other error report is pending, or at label "uinme" if this is a multiple error condition. On a multiple error, an attempt is made to stop the external host system from sending characters, and the UART receiver is disabled until the CPU issues a SELECT-UART command. (A single error does not disable the receiver, because no data has been lost yet: the !DATA-ERR interrupt reports the character with the error report.)

5.1.7 Buffer Status Reporting

For internal debugging purposes, four unassigned signals from the LCD Contrast Latch are updated to show the status of the buffer. While the buffer is full enough to pass to the CPU, one bit of the latch (IC 25G, pin 12) is high. While the buffer is full enough that the external system should stop, pin 15 is high. While the CPU is not ready to receive data from the CPU, pin 16 is high. If a buffer overrun condition occurs, and data is lost, or if any fatal error occurs (with a hexadecimal code appearing on the LCD display), then pin 19 goes high. The code that handles these bits is flagged with the word "DEBUG" in the comment field.

5.1.8 Background Processing/Monitoring Tasks

These are tasks that are not triggered directly by CPU commands.

Real-Time Clock (T1) Timer T1 is loaded with a constant interval value which is used to interrupt the HPC at 10 ms intervals. When the Timer T1 interrupt occurs (labels "tmrint", "t1poll", "t1int"), and the real-time interrupt is enabled, the variable "rtccnt" is decremented to determine whether a !RTC interrupt should be issued to the CPU. If so, the bit "artc" in the "alert" word is set, requesting the main program to send a !RTC interrupt to the CPU. The main program, at label "sndrtc", interrupts the CPU. No other data is passed to the CPU.

At label "kbdchk" the panel pushbutton switches are also sampled. This process is described fully in Application Note AN-550.

At label "dsrchk", the state of the UART DSR flag is checked if the UART is selected and DSR monitoring mode has been requested by the CPU. If it has changed, this routine requests the Main Program to issue a !UART-STATUS

interrupt to the CPU. The UART receiver is also enabled and disabled by the state of this signal if DSR monitoring has been requested. (The CPU does not have to react to the interrupt for normal operation, but might wish to record its occurrence.)

At label "brkchk", if the UART is selected, and a BREAK has been detected, the UART data input pin is polled to determine whether the BREAK condition has ended. If a BREAK has ended, then this routine requests the Main Program to issue a !UART-STATUS interrupt to the CPU.

Centronics INPUT PRIME When the $\overline{\text{EXU}}$ pin on the HPC is activated, and the Centronics port is selected rather than the UART, the UART service routine (at label "uarprm") sets bit "aprime" in the "alert" variable, requesting the main program to send a !PRIME interrupt to the CPU. The Centronics port is internally flagged (in the "cps" variable) as being "busy", and the Centronics Control Latch is updated to set the BUSY signal high. The UART interrupt is then disabled until a SELECT-CENT command is received from the CPU. In the main program, the !PRIME interrupt is sent to the CPU at label "sndprm". No other data is sent.

5.2 HPC Firmware Listing

```

# Centronics Port input / checksum calculation / LCD output.
#   Accepts up to 1024 characters on Centronics port,
#   accumulates 8-bit checksum, and on receiving Ctrl-D,
#   displays checksum on LCD display.

```

```

.globl start,main
.globl dataint,rtcint,primeint
.globl lcdint
.globl swint,usttsint,errint,uwrint
.globl diaqint,badint

```

```

.set hpcctrl,0xFFFC00      # HPC Control/Status I/O location.
.set hpcdata,0xFFE00      # HPC Data I/O location.
.set hpcpoll,0xFD0000     # HPC Poll address (UPIC).

```

```

.set cr,0xD
.set lf,0xA
.set ctrlD,'D'-0x40

```

```
start:
```

```
    # Fill interrupt vector locations.
```

```

addr badint,vex          # Interrupt NMI. (Unimplemented)
addr dataint,vex+4      # Interrupt 0x10. Comm Buffer data.
addr rtcint,vex+8       # Interrupt 0x11. Real-Time Clock.
addr badint,vex+12      # Interrupt 0x12. (Unimplemented)
addr primeint,vex+16    # Interrupt 0x13. Centronics PRIME.
addr badint,vex+20      # Interrupt 0x14. (Unimplemented)
addr badint,vex+24      # Interrupt 0x15. (Unimplemented)
addr badint,vex+28      # Interrupt 0x16. (Unimplemented)
addr lcdint,vex+32      # Interrupt 0x17. LCD data written.
addr swint,vex+36       # Interrupt 0x18. Pushbutton event.
addr usttsint,vex+40    # Interrupt 0x19. UART Status change.
addr errint,vex+44      # Interrupt 0x1A. Error detected.
addr uwrint,vex+48      # Interrupt 0x1B. UART Write ack.
addr badint,vex+52      # Interrupt 0x1C. (Unimplemented)
addr diaqint,vex+56     # Interrupt 0x1D. Diagnostic.
addr badint,vex+60      # Interrupt 0x1E. (Unimplemented)
addr badint,vex+64      # Interrupt 0x1F. (Unimplemented)
addr badint,vex+68      # Interrupt 0x20. (Unimplemented)
addr badint,vex+72      # Interrupt 0x21. (Unimplemented)

```

```

movb $0,hpcctrl        # INITIALIZE command.
movb $100,hpcdata      # RTC value: once per second.

```

```

movb $0x0B,hpcctrl    # Turn on two LED's to show we're alive.
movb $0x06,hpcdata

```

```

movb $1,hpcctrl       # Select Centronics port.
movb $1,hpcdata       #   BUSY drops during ACK/ pulse.
movb $100,hpcdata     #   Accept 100 characters before passing
#                       #   buffer to CPU;
movb $120,hpcdata     #   Apply flow control if buffer has 120
#                       #   characters.

```

TL/DD/9977-6

```

run:      bispsrw $0x800      # Enable interrupts from HPC.

main:          # Main program starts here.

      movd    datoptr,r1     # Register R1 contains buffer out pointer.

mwait:  cmpd   datiptr,r1    # Wait here for a block to come in.
      bls    mwait

      movb   0(r1),r0       # Here, process character.
      cmpb  r0,$ctrlD      # if End of File, go type checksum.
      beq   typout

      addb  r0,ckdata
      addqd $1,r1
      br   mwait

typout:      # Send checksum out on LCDs.
      bicpsrw $0x800      # Disable interrupts.

      cbtbb  $0,poutflg    # Clear LCD output acknowledge flag.

      movb  $0xA,hpcctrl   # Send-LCD command.
      movb  $0x6,hpcdata
      movb  $3,hpcdata

      movb  $0x1,hpcdata   # Clear panel LCD's.

      movzbd ckdata,r0     # Send first hex character.
      lshd  $-4,r0
      movb  asctab[r0:b],r0
      movb  r0,hpcdata

      movzbd ckdata,r0     # Send second hex character.
      andb  $0xF,r0
      movb  asctab[r0:b],r0
      movb  r0,hpcdata

      bispsrw $0x800      # Re-enable interrupts.

pnlout:      tbtbb  $0,poutflg
      bfc   pnlout

      movqb  0,ckdata
      movd  $databuf,datiptr
      movd  datoptr,r1

      br   mwait # Close loop: infinite.

      ret   0 # End of main program.

maindat:     # Data for Main Program.

datiptr: .double databuf # Pointer to Data Buffer area.
datoptr: .double databuf # Pointer to Data Buffer area.
poutflg: .byte 1 # UART Output Ready.
ckdata: .byte 0 # Accum. checksum.
asctab: .byte '0','1','2','3','4','5','6','7'

```

TL/DD/9977-7

```

        .byte '8','9','a','b','c','d','e','f'
databuf: .blkb 1024 # Data buffer area.

# Start of Interrupt Service Routines.
# Invoked by ROM interrupt service. Registers R0..R2 are already
# saved, but no ENTER instruction has been performed yet.

dataint:      # Interrupt 0x10.  Comm Buffer ready.

        movzhd hpcdata,r0      # Get character count from HPC.
        movd   datiptr,r1

dataalp:      movb   hpcdata,0(r1) # Loop: get character from HPC,
        addgd  l,r1           # increment buffer address,
        acbd   -1,r0,dataalp  # decrement count and loop.

        movd   r1,datiptr
        ret    0

rtcint:      # Interrupt 0x11.  Real-Time Clock.

        movb   $4,hpcctrl     # Send Flush-Buf command to HPC.
        ret    0

primeint:    # Interrupt 0x13.  Centronics PRIME.

        movb   $1,hpcctrl
        movb   $1,hpcdata
        movb   $100,hpcdata
        movb   $120,hpcdata
        ret    0

lcdint:      # Interrupt 0x17.  LCD data written.

        sbitb  $0,poutflg
        ret    0

swint:      # Interrupt 0x18.  Pushbutton event.

        br     badint
        ret    0

usttsint:    # Interrupt 0x19.  UART Status change.

        br     badint
        ret    0

errint:      # Interrupt 0x1A.  Error detected.

        br     badint
        ret    0

uwrint:      # Interrupt 0x1B.  UART Write ack.

        br     badint
        ret    0

diagint:     # Interrupt 0x1D.  Diagnostic.

```

```
movb hpcdata,r0  
movb hpcdata,r0  
movb hpcdata,r0  
movb hpcdata,r0  
movb hpcdata,r0  
ret 0
```

badint: # Trap for unimplemented interrupts.

```
ret 0
```

TL/DD/9977-9

```

# UART Port input / checksum calculation / UART output.
#   Accepts up to 1024 characters on UART port,
#   accumulates 8-bit checksum, and on receiving Ctrl-D,
#   displays checksum by sending out on RS-232 port.

.globl start,main
.globl dataint,rtcint,primeint
.globl lcdint
.globl swint,usttsint,errrint,uwrint
.globl diagint,badint

.set   hpcctrl,0xFFFFC00    # HPC Control/Status I/O location.
.set   hpcdata,0xFFFFE00    # HPC Data I/O location.
.set   hpcpoll,0xFD0000     # HPC Poll address (UPIC).

.set   cr,0xD
.set   lf,0xA
.set   ctrlD,'D'-0x40

start:
                                # Fill interrupt vector locations.

addr   badint,vex              # Interrupt NMI.   (Unimplemented)
addr   dataint,vex+4           # Interrupt 0x10. Comm Buffer data.
addr   rtcint,vex+8            # Interrupt 0x11. Real-Time Clock.
addr   badint,vex+12           # Interrupt 0x12.
addr   primeint,vex+16         # Interrupt 0x13. Centronics PRIME.
addr   badint,vex+20           # Interrupt 0x14.
addr   badint,vex+24           # Interrupt 0x15.
addr   badint,vex+28           # Interrupt 0x16.
addr   lcdint,vex+32           # Interrupt 0x17. LCD data written.
addr   swint,vex+36            # Interrupt 0x18. Pushbutton event.
addr   usttsint,vex+40         # Interrupt 0x19. UART Status change.
addr   errrint,vex+44          # Interrupt 0x1A. Error detected.
addr   uwrint,vex+48           # Interrupt 0x1B. UART Write ack.
addr   badint,vex+52           # Interrupt 0x1C. (Unimplemented)
addr   diagint,vex+56          # Interrupt 0x1D. Diagnostic.
addr   badint,vex+60           # Interrupt 0x1E. (Unimplemented)
addr   badint,vex+64           # Interrupt 0x1F. (Unimplemented)
addr   badint,vex+68           # Interrupt 0x20. (Unimplemented)
addr   badint,vex+72           # Interrupt 0x21. (Unimplemented)

movb   $0,hpcctrl              # INITIALIZE command.
movb   $100,hpcdata            # RTC value: once per second.

movb   $0x0B,hpcctrl           # Turn on two LED's to show we're alive.
movb   $0x06,hpcdata

movb   $2,hpcctrl              # Select UART and set up parameters.
movb   $5,hpcdata              #   9600 baud,
movb   $2,hpcdata              #   8 bits, no parity,
movb   $0xA,hpcdata            #   XON/XOFF protocol, DTR always on.
movb   $100,hpcdata            #   Accept 100 characters before passing
                                #   buffer to CPU;
movb   $120,hpcdata            #   Apply flow control if buffer has 120

```

```

                                # characters.

run:
    bispsrw $0x800           # Enable interrupts from HPC.

main:                          # Main program starts here.

    movd    datoptr,r1       # Register R1 contains buffer out pointer.

mwait: cmpd    datiptr,r1    # Wait here for a block to come in.
      bls    mwait

    movb    0(r1),r0         # Here, process character.
    cmpb    r0,$ctrlD       # if End of File, go type checksum.
    beq     typout

    addb    r0,ckdata
    addqd   $1,r1
    br     mwait

typout:                         # Send checksum out on RS-232 port.

    movb    $cr,r0
    bsr     serout

    movb    $lf,r0
    bsr     serout

    movzbd  ckdata,r0
    lshd    $-4,r0
    movb    asctab[r0:b],r0
    bsr     serout

    movzbd  ckdata,r0
    andb    $0xF,r0
    movb    asctab[r0:b],r0
    bsr     serout

    movb    $cr,r0
    bsr     serout

    movb    $lf,r0
    bsr     serout

    movqb   0,ckdata
    movd    $databuf,datiptr
    movd    datoptr,r1

    br     mwait           # Close loop: infinite.

    ret     0              # End of main program.

serout:  tbitb  $0,uoutflg
      bfc     serout

      cbitb  $0,uoutflg    # Indicate UART not ready.

      bicpsrw $0x800      # Critical Sequence:
      movb    $0xD,hpcctrl # Give Send-UART command to HPC.
      movb    r0,hpcdata  # Give character to HPC.
      bispsrw $0x800      # End critical sequence.

```

TL/DD/9977-11


```

ret      0

maindat:    # Data for Main Program.

datiptr: .double databuf # Pointer to Data Buffer area.
datoptr: .double databuf # Pointer to Data Buffer area.
uoutflg: .byte 1        # UART Output Ready.
ckdata:  .byte 0        # Accum. checksum.
asctab:   .byte '0','1','2','3','4','5','6','7'
          .byte '8','9','a','b','c','d','e','f'
databuf: .blkb 1024    # Data buffer area.

# Start of Interrupt Service Routines.
# Invoked by ROM interrupt service. Registers R0..R2 are already
# saved, but no ENTER instruction has been performed yet.

dataint:   # Interrupt 0x10.  Comm Buffer ready.

movzbd    hpcdata,r0    # Get character count from HPC.
movd      datiptr,r1

datalp:   movb    hpcdata,0(r1) # Loop: get character from HPC,
addqd    1,r1           # increment buffer address,
acbd     -1,r0,datalp  # decrement count and loop.

movd     r1,datiptr
ret      0

rtcint:   # Interrupt 0x11.  Real-Time Clock.

movb     $4,hpcctrl    # Send Flush-Buf command to HPC.
ret      0

primeint: # Interrupt 0x13.  Centronics PRIME.

br       badint
ret      0

lcdint:   # Interrupt 0x17.  LCD data written.

br       badint
ret      0

swint:    # Interrupt 0x18.  Pushbutton event.

br       badint
ret      0

usttsint: # Interrupt 0x19.  UART Status change.

br       badint
ret      0

errint:   # Interrupt 0x1A.  Error detected.

br       badint

```

```
ret 0

uwrint:          # Interrupt 0x1B.  UART Write ack.

    sbitb $0,uoutflg
    ret 0

diagint:        # Interrupt 0x1D.  Diagnostic.
    movb hpcdata,r0
    movb hpcdata,r0
    movb hpcdata,r0
    movb hpcdata,r0
    movb hpcdata,r0
    ret 0

badint:         # Trap for unimplemented interrupts.

    ret 0
```

TL/DD/9977-13

```

.title CENTUART,'HPC FIRMWARE: CENTRONICS/UART PORTS'
;
; program centuart.asm version 1.0      05/22/88
; Copyright (C) 1988 by National Semiconductor Corp.
; (******)
; (*)
; (*) Copyright (c) 1988      by National Semiconductor Corporation (*)
; (*)
; (*) National Semiconductor Corporation (*)
; (*) 2900 Semiconductor Drive (*)
; (*) Santa Clara, California 95051 (*)
; (*) (*)
; (*) All rights reserved (*)
; (*) (*)
; (*) This software is furnished under a license and may be used (*)
; (*) and copied only in accordance with the terms of such license (*)
; (*) and with the inclusion of the above copyright notice. This (*)
; (*) software or any other copies thereof may not be provided or (*)
; (*) otherwise made available to any other person. No title to and (*)
; (*) ownership of the software is hereby transferred. (*)
; (*) (*)
; (*) The information in this software is subject to change without (*)
; (*) notice and should not be construed as a commitment by National (*)
; (*) Semiconductor Corporation. (*)
; (*) (*)
; (*) National Semiconductor Corporation assumes no responsibility (*)
; (*) for the use or reliability of its software on equipment (*)
; (*) configurations which are not supported by National (*)
; (*) Semiconductor Corporation. (*)
; (*) (*)
; (******)
;
; Derived from hpcupi.asm file. However, commands have
; been re-mapped (different code values), and so are not exactly
; upward compatible.
;
; Adds commands and interrupts to support input, buffering,
; handshaking and mode selection for an RS-232 port and
; a Centronics-style parallel port.
;
;
; .form 'Declarations: Register Addresses'

psw = x'C0:w ; PSW register
al = x'C8:b ; Low byte of Accumulator.
ah = x'C9:b ; High byte of Accumulator.
bl = x'CC:b ; Low byte of Register B.
bh = x'CD:b ; High byte of Register B.
xl = x'CE:b ; Low byte of Register X.
xh = x'CF:b ; High byte of Register X.

enir = x'D0:b
irpd = x'D2:b
ircd = x'D4:b
sio = x'D6:b
port1 = x'D8:b

```

```

obuf = x'E0:b ; (Low byte of PORTA.)
portah = x'E1:b ; High byte of PORTA.
portb = x'E2:w
portbl = x'E2:b ; Low byte of PORTB.
portbh = x'E3:b ; High byte of PORTB.
upic = x'E6:b
ibuf = x'F0:b ; (Low byte of DIRA.)
dirah = x'F1:b ; High byte of DIRA.
dirb = x'F2:w
dirbl = x'F2:b ; Low byte of DIRB.
dirbh = x'F3:b ; High byte of DIRB.
bfun = x'F4:w
bfunl = x'F4:b ; Low byte of BFUN.
bfunh = x'F5:b ; High byte of BFUN.

portd = x'0104:b
enu = x'0120:b
enui = x'0122:b
rbuf = x'0124:b
tbuf = x'0126:b
enur = x'0128:b

t4 = x'0140:w
r4 = x'0142:w
t5 = x'0144:w
r5 = x'0146:w
t6 = x'0148:w
r6 = x'014A:w
t7 = x'014C:w
r7 = x'014E:w
pwmode = x'0150:w
pwm dl = x'0150:b ; Low byte of PWMODE.
pwm dh = x'0151:b ; High byte of PWMODE.
portp = x'0152:w
portpl = x'0152:b ; Low byte of PORTP.
portph = x'0153:b ; High byte of PORTP.
eicon = x'015C:b

t1 = x'0182:w
r1 = x'0184:w
r2 = x'0186:w
t2 = x'0188:w
r3 = x'018A:w
t3 = x'018C:w
divby = x'018E:w
divbyl = x'018E:b ; Low byte of DIVBY.
divbyh = x'018F:b ; High byte of DIVBY.
tm mode = x'0190:w
tm dl = x'0190:b ; Low byte of TMODE.
tm dh = x'0191:b ; High byte of TMODE.
tOcon = x'0192:b

```

.form 'Declarations: Register Bit Positions

```

; Name      Position      Register(s)
; -----
gie = 0 ; enir
i2 = 2 ; enir, irpd, ircd

```

TL/DD/9977-15

```

i3      =      3      ; enir, irpd, ircd
i4      =      4      ; enir, irpd, ircd
tars    =      5      ; enir, irpd
uart    =      6      ; enir, irpd
ei       =      7      ; enir, irpd

dsr     =      7      ; porti only: poll UART DSR.

uvmode  =      1      ; ircd
uwdone  =      0      ; irpd

tbmt    =      0      ; enu
rbfl    =      1      ; enu
b8or9   =      4      ; enu
xbit9   =      5      ; enu
wakeup  =      2      ; enur
rbit9   =      3      ; enur
frterr  =      6      ; enur
doeerr  =      7      ; enur
eti     =      0      ; enui
eri     =      1      ; enui
xtclk   =      2      ; enui
xrclk   =      3      ; enui
b2stp   =      7      ; enui

wrrdy   =      0      ; upic
rdrdy   =      1      ; upic
la0     =      2      ; upic
upien   =      3      ; upic
b8or16  =      4      ; upic

t0tie   =      0      ; tmd1
t0pnd   =      1      ; tmd1
t0ack   =      3      ; tmd1
t1tie   =      4      ; tmd1
t1pnd   =      5      ; tmd1
t1stp   =      6      ; tmd1
t1ack   =      7      ; tmd1
t2tie   =      0      ; tmdh
t2pnd   =      1      ; tmdh
t2stp   =      2      ; tmdh
t2ack   =      3      ; tmdh
t3tie   =      4      ; tmdh
t3pnd   =      5      ; tmdh
t3stp   =      6      ; tmdh
t3ack   =      7      ; tmdh

t4tie   =      0      ; pvmd1
t4pnd   =      1      ; pvmd1
t4stp   =      2      ; pvmd1
t4ack   =      3      ; pvmd1
t5tie   =      4      ; pvmd1
t5pnd   =      5      ; pvmd1
t5stp   =      6      ; pvmd1
t5ack   =      7      ; pvmd1
t6tie   =      0      ; pvmdh
t6pnd   =      1      ; pvmdh
t6stp   =      2      ; pvmdh
t6ack   =      3      ; pvmdh
t7tie   =      4      ; pvmdh

```

```

t7pnd = 5 ; pmdh
t7stp = 6 ; pmdh
t7ack = 7 ; pmdh

t4out = 0 ; portpl
t4tfn = 3 ; portpl
t5out = 4 ; portpl
t5tfn = 7 ; portpl
t6out = 0 ; portph
t6tfn = 3 ; portph
t7out = 4 ; portph
t7tfn = 7 ; portph

cenclk = 0 ; portph (CCTLCLK signal).

txd = 0 ; portbl, dirbl, bfunl
dtr = 1 ; portbl, dirbl
pnlclk = 7 ; portbl, dirbl

lcvcclk = 1 ; portbh, dirbh
; ua0 would be 2 , but requires no setup.
uwrrdy = 3 ; portbh, dirbh, bfunh
cdata = 4 ; portbh (enables Centronics data to Port D).
astts = 5 ; portbh (enables panel switches to Port D).
ledclk = 6 ; portbh, dirbh
urdrdy = 7 ; portbh, dirbh, bfunh

; CONSTANTS
;
xon= x'11 ; XON character: Control-Q
xoff= x'13 ; XOFF character: Control-S

.form 'Space Declarations'

botad= x'40 ; First address in buffer.
topad= x'BF ; Last address in buffer.
bufsiz= topad-botad+1 ; Length of buffer.
;
.sect BUFFER,BASE,ABS=botad ; Data Communication Buffer.

.dsb bufsiz

.endsect

.sect DSECT,BASE,REL ; Basepage RAM variables (addresses 0000-00BF)

; WORD-ALIGNED
dummy: .dsw 1 ; x'00,01 ; Destroyed on reset (address 0).
.set upicsv,dummy ; Temporary image of UPIC register.
alert: .dsw 1 ; Alert status bits to main program:
; generate interrupts to CPU.
.set alerth,alert+1 ; Declare top byte of ALERT word.
cpuad: .dsw 1 ; Current address within CPU command buffer.
cpubuf: .dsw 4 ; Buffer for accepting command parameters from CPU.
lcdsix: .dsw 1 ; Pointer into LCD character string buffer.

;BYTE-ALIGNED

```

```

numchr:      .dsb 1 ; Number of characters currently in data buffer.
cadin: .dsb 1 ; Current input byte address in data buffer
           ; (first empty loc.).
cadout:      .dsb 1 ; Current output byte address in data buffer.
pascnt:      .dsb 1 ; Number of characters before data buffer full enough to
           ; transmit to CPU.
stpcnt:      .dsb 1 ; Number of characters before host system is told to stop
           ; transmitting.
numout:      .dsb 1 ; Number of data characters (total) being sent to CPU in
           ; current transfer.
cntout: .dsb 1 ; Number of data characters remaining to be sent to CPU in
           ; current transfer.
bstat: .dsb 1 ; Buffer Status byte.
cps: .dsb 1 ; Centronics Port Status byte
           ; (image of control signals).
ackmd: .dsb 1 ; Acknowledge Timing Mode: Position of ACK/ pulse edges
           ; on Centronics port relative to BUSY falling edge.
curcmd:      .dsb 1 ; Current command byte from CPU being processed.
numexp:      .dsb 1 ; Number of parameter bytes expected before command processing
           ; begins.
lcvs: .dsb 1 ; Image of LCD Voltage (Contrast) latch setting; needed with
           ; LCD RS (PAUX0) signal coming from this latch.
fshlim:      .dsb 1 ; Flush limit count: used to limit number of characters passed
           ; to CPU when an error report is pending.
errchr:      .dsb 1 ; Holds character on which an error was detected.
errfgs:      .dsb 1 ; Holds error flags associated with error character.
lcdfgs:      .dsb 1 ; Holds flag bits for characters sent to Panel LCD display.
lcdnum:      .dsb 1 ; Number of characters to be sent to LCD display.
lcdsf:       .dsb 1 ; Flag bits associated with characters in LCD String Buffer.
lcdsct:      .dsb 1 ; Counter for characters being sent to LCD display from String
           ; Buffer.
swlast:      .dsb 1 ; Last-sampled switch values.
swsnt:      .dsb 1 ; Last switch values sent to CPU.
beepct:      .dsb 1 ; Beep duration count. Counts occurrences of T0 interrupt.
uframe:      .dsb 1 ; Frame mode for UART.
uflow: .dsb 1 ; Flow control mode for UART.
ups: .dsb 1 ; UART Status byte.
uschr: .dsb 1 ; UART Send Character: from CPU.
uinchr:      .dsb 1 ; UART Input Character: character last received from UART.
enring:      .dsb 1 ; UART ENUR register image in memory.
rtcivl:      .dsb 1 ; Real-Time Clock Interval (units of 10 milliseconds).
rtccnt:      .dsb 1 ; Real-Time Clock Current Count (units of 10 milliseconds).
rtevs: .dsb 1 ; Events to check for on Timer T1 interrupts.
ustat: .dsb 1 ; UART status for CPU.
dsevc: .dsb 1 ; Diagnostic Interrupt: Severity Code.
derrc: .dsb 1 ; Diagnostic Interrupt: Error Code.
dbyte: .dsb 1 ; Diagnostic Interrupt: Error Byte.
dccmd: .dsb 1 ; Diagnostic Interrupt: Current Command.
dqual: .dsb 1 ; Diagnostic Interrupt: Qualifier (Command Status).

```

```

; * Addresses 0040-00BF are reserved for the Data Communication Buffer
; (128 bytes).

```

```

; BIT POSITIONS

```

```

           ; Bits in BSTAT byte (Data Communication Buffer Status):
pass= 0 ; Data is ready to be passed to the CPU.
passng= 1 ; Indicates that some of the data in the buffer is being
           ; passed to the CPU.
stop= 2 ; Indicates that host has been requested to stop transmitting.

```

```

cpubsy=      3      ; Indicates that CPU is not able to receive any more data.
ifcbusy=    4      ; Indicates that the interface is considered busy by CPU.
full= 5     ; Indicates that the interface is completely full. Any more
              ; characters will overflow it.

              ; Bits in CPS (Centronics Port Status byte)
cack= 0     ; ACK/ Strobe.
caux1= 1    ; AUXOUT1 Signal.
cbusy= 2    ; BUSY Signal.
cselect= 3  ; SELECT Signal.
ccall= 4    ; CALL Signal.
cfault= 5   ; FAULT/ Signal.
caux2= 6    ; AUXOUT2 Signal.
enprm= 7    ; 1 enables INPUT PRIME/ interrupt from Centronics port.

              ; Bits in ACKMD (Centronics Acknowledge Mode byte)
; (Bits 0 and 1 give timing relationship between BUSY and ACK/.)
clinmd= 2   ; 1 = Centronics Line Mode. Buffer limits must also both be 1.
; (Other bits unassigned.)

              ; ALERT status word (low-order byte) bits:
aflush = 0   ; Flush Data Buffer.
artc = 1    ; Real-Time Interrupt detected.
aprime = 3   ; INPUT PRIME detected from Centronics interface.
alcgak = 7   ; LCD Panel Write Acknowledge.
              ; ALERT status word (high-order byte, named alerth) bits:
abutton = 0  ; Pushbutton switch state change.
austat = 1   ; UART status change.
aerr = 2    ; Error detected (UART or buffer overflow).
auack = 3   ; UART output acknowledge: character sent.
adiag = 5   ; Diagnostic interrupt.
; (Other bits not defined.)

              ; ERRFGS error flags byte sent to CPU with !BAD-DATA interrupt:
doe= 7     ; Data Overrun Error on UART.
frm= 6     ; Framing Error on UART.
par= 5     ; Parity error on UART.
bufovf= 4  ; Buffer Overflow condition (flow control did not work).
errovf= 3  ; Error Overflow condition. Two or more errors occurred
              ; so close together that the first error could not be
              ; reported before the second error occurred. Details
              ; of the second error are lost.
brk= 2     ; Break condition detected in addition to Framing error.
; (Other bits not defined.)

              ; CURCMD byte: Current CPU command. The lower 5 bits contain a code
              ; in the range 0-10 (hex). The upper two bits contain
              ; further information about command collection:
cmdemp= 7  ; Bit 7 (MSB) of curcmd = 1 means that no command is being
              ; processed and curcmd byte is "empty".
getcmt= 6  ; Bit 6 of curcmd = 1 means that the count is being received
              ; for a variable-length command.

              ; LCVS byte: LCD Voltage (Contrast) Latch memory image.
              ; Contains voltage value in its least-significant 3 bits,
              ; RS signal to LCD controller in bit 3, and debugging
              ; information in its top 4 bits.
pnlrs= 3   ; Bit 3 is (inverted) RS signal to panel.

              ; UPS byte: Status of UART output and flow control.

```

TL/DD/9977-19


```

usel= 7      ; When set, means that UART port is selected.
mcmd= 6      ; Receiver disabled due to multiple character error.
brkmd= 5     ; BREAK signal has been detected and is still active; receiver
              ; disabled.
onebrk= 4    ; One character which is possibly a BREAK has been seen.
icpu= 3      ; When set, means that CPU should be informed of next UART
              ; transmitter interrupt.
schr= 2      ; Request to send a character from uschr location (from CPU).
cus= 1       ; Current UART status: 1 = stopped.
luss= 0      ; Last UART Status Sent: Indicates what the external system
              ; thinks the UART's status is.

```

```

; UFLOW byte: Modes for UART flow control.
flemp= 7     ; 1 = No flow control yet provided since reset.
;(intervening bits not defined.)
xonb= 3      ; 1 = XON/XOFF protocol mode selected.
dtrbl= 2     ; DTR Mode field: 00 = permanently low.
dtrb0= 1     ;           01 = permanently high.
              ;           10 = low when ready.
              ;           11 = high when ready.
dsrb= 0      ; 1 = characters received while DSR low will not be accepted.

```

```

; USTAT byte: Status of UART reported to CPU.
dsrflg= 0    ; State of DSR signal. 1 = Data Set Ready condition.
brkflg= 1    ; 1 = End of BREAK condition detected.

```

```

; RTEVS byte: Events to check for at 10-millisecond intervals.
; (T1 Underflows)
rtcenb= 0    ; 1 = Real-Time Clock interrupts enabled to CPU.
brkenb= 1    ; 1 = UART Break mode; report end of break.

```

```

.sect STACK, RAM16, REL      ; On-chip RAM in addresses 01C0-01FF.
stackb: .dsv 16             ; Space for 8 words beyond
              ; interrupt context.
avail: .dsv 12             ; Spare portion of this space.
lcdbuf: .dsv 4              ; LCD String Buffer.

```

```

.form 'Code Section'
.sect CSECT, ROM16, REL ; Code space. (On-chip ROM)

```

```

; Declarations of subroutines called by one-byte JSRP instruction.

```

```

.spt rdwait      ; Waits for CPU to read a value from UPI port.
.spt wrpnl      ; Writes to LCD panel (for initialization only).

```

```

; Program starts at label "start" on reset. This routine is the fatal
; error handler, located here for convenience in setting breakpoint.

```

```

hangup: rbit gie, enir      ; Fatal error: signal it and halt.
sbit 7, lcvs              ; Signal error on most-significant bit of
                          ; LCD Contrast Latch.
sbit pnlns, lcvs         ; Select command mode for LCD controller.
ld portah, lcvs          ; Place error on Port A for latch.
sbit lcvclk, portbh      ; Clock LCD Contrast Latch high,
rbit lcvclk, portbh      ; then low to load it.
sbit t6stp, pwmdh        ;
rbit t6tie, pwmdh        ; Set up Timer T6 for non-interrupt use.
nop
rbit t6pnd, pwmdh        ; Clear Pending bit.

```

```

pop      0.w          ; Get error address from stack.
ld       sp.w,#stackb ; In case of stack underflow, re-initialize SP.
ld       A,#x'01
jsrl    wrpnl        ; Clear LCD panel.
rbit    pn1rs,lcvs   ; Set up panel for data.
ld       portah,lcvs ; Place error on Port A for latch.
sbit    lcvc1k,portbh ; Clock LCD Contrast Latch high,
rbit    lcvc1k,portbh ; then low to load it.
ld       A,l.b       ; Process first character of return address.
swap    A
and     A,#x'0F
ld      A,hextab[A].b
jsrl    wrpnl        ; Display it on LCD panel.
ld      A,l.b       ; Process second character of return address.
and     A,#x'0F
ld      A,hextab[A].b
jsrl    wrpnl        ; Display it on LCD panel.
ld      A,O.b       ; Process third character of return address.
swap    A
and     A,#x'0F
ld      A,hextab[A].b
jsrl    wrpnl        ; Display it on LCD panel.
ld      A,O.b       ; Process last character of return address.
and     A,#x'0F
ld      A,hextab[A].b
jsrl    wrpnl        ; Display it on LCD panel.

hgupi:  ifbit    rdrdy,upic ; Check to see if OBUF register is full.
        ld      obuf,#wdiag ; If not, fill it with !DIAG vector
        ; continuously.
        ifbit    i3,irpd    ; Check for UPI data ready.
        jp      hgupi1
        jp      hgupi

hgupi1: ifeq     ibuf,#x'A5 ; Check for RESET command.
        jp      hgrst
        jp      hgupi2
hgrst:  ifbit    la0,upic
        jp      hgupi2
        jmpl    xreset      ; If so, then go reset the HPC.

        ; This is part of the outer loop, waiting for
        ; the RESET command.
hgupi2: ld      irpd,#x'F7 ; Clear the UWR detector,
        jp      hgupi     ; and keep looking. This is an
        ; infinite loop until RESET is seen.

hextab: .byte   '0','1','2','3','4','5','6','7'
        .byte   '8','9','A','B','C','D','E','F'

        .form   'Hardware Initialization'

start:  ld      psw.b,#x'08 ; Set one WAIT state.

srfsh: ; Start dynamic RAM refreshing,
        ; as quickly as possible.
sbit    t4out,portpl ; Trigger first refresh
        ; immediately.
sbit    t4stp,pwmd1  ; Stop timer T4 to
        ; allow loading,

```

TL/DD/9977-21

```

ld      t4,#8      ; then load it.
rbit   t4stp,pwmdl ; Start timer T4.
sbit   t4tfn,portpl ; Enable pulses out.
ld      r4,#8      ; Load R4.

sup1:
ld      upic,#x'18 ; Set up UPI port.
        ; 8-Bit UPI Mode
        ; enabled.

sbit   uvrrdy,bfunh ; Enable UWRRDY/ out.
sbit   uvrrdy,dirbh
ld      A,ibuf      ; Empty IBUF register,
        ; in case of false trigger.

sbit   urdrdy,bfunh ; Enable URDRDY/ out.
sbit   urdrdy,dirbh

        ; Set up UREAD/ interrupt.
sbit   i2,ircd      ; Detects rising edges.
ld      irpd,#x'FB ; Clear any false interrupt
        ; due to mode change.

        ; Set up UWRITE/ interrupt.
sbit   i3,ircd      ; Detects rising edges.
ld      irpd,#x'F7 ; Clear any false interrupt
        ; due to mode change.

sram:
        ; Clear all RAM locations.
        ; Clear Basepage bank:
ld      BK,#x'0000,#x'00BE ; Establish loop base and limit.
sram11:
clr     A
xs     A,[B+].w
jp     sram11

        ; Clear Non-Basepage bank:
ld      BK,#x'01C0,#x'01FE ; Establish loop base and limit.
sram12:
clr     A
xs     A,[B+].w
jp     sram12

sskint:
        ; Set up Stack and remove
        ; individual interrupt enables.
ld      sp.w,#stackb+2 ; Move stack to high
        ; bank of on-chip RAM.
ld      stackb.w,#hangup ; Safeguard against
        ; stack underflow.
ld      enir,#x'00      ; Disable interrupts
        ; individually.

tminit:
ld      t0con,#x'08
ld      tmnode,#x'4440 ; Stop timers T1, T2, T3.
ld      divby,#x'0055 ; UART set to 9600 Baud.
ld      tmnode,#x'CCC8 ; Clear and disable timer
        ; T0-T3 interrupts.

scent:
        ; Set up Centronics parallel
        ; port.
ld      dirah,#x'FF      ; Enable multiplexed outputs.
sbit   astts,portbh      ; Enable and remove ENASTTS/ signal.
sbit   astts,dirbh

```

```

sbit  cdata,portbh ; Enable and remove ENCDATA/ signal.
sbit  cdata,dirbh
ld    cps,#x'25 ; Set up Port A data for
                ; Centronics Control.
jsrl  setcen      ; Send to Centronics latch and to Busy flag.

                ; Set up I4 interrupt on
sbit  i4,ircd    ; CINTR/ (rising edge).
ld    irpd,#x'EF ; Clear any false interrupt
                ; caused by mode change.

suart:
                ; Set up RS-232 port.
sbit  txd,bfunl  ; Enable TXD output.
sbit  txd,dirbl
rbit  dtr,portbl ; Set up DTR signal. (State is arbitrary:
                ; low typically means not ready.)
sbit  dtr,dirbl ; Enable it as an output pin.
ld    enu,#x'0   ; 8-bit Mode.
ld    enur,#x'0 ; Clear Wake-Up Mode.
ld    enul,#x'80 ; Internal baud; 2 stop
                ; bits; no interrupts.

sled: ld portah,#x'FF ; Set up to turn off LED's.
rbit  ledclk,portbh ; Start with LEDCLK low,
sbit  ledclk,dirbh ; (enable output),
sbit  ledclk,portbh ; then high,
rbit  ledclk,portbh ; then low again.

stmr:
                ; Set up remaining timers.
                ; (T1-T3 already stopped
                ; and pending bits cleared
                ; at tinit above.)

ld    t1,#12287 ; T1 runs at 10-millisecond real-time interval.
ld    r1,#12287
                ; Timer remains stopped, and interrupt
                ; disabled, until INITIALIZE command.

ld    pwmode,#x'4440 ; Stop timers T5-T7.
nop   ; Wait for valid PND
nop   ; bits.
ld    pwmode,#x'CCC8 ; Clear and disable
                ; interrupts from all
                ; PWM timers.

ld    r6,#x'FFFF ; No modulus for LCD Display Ready timer.

ld    t7,#204 ; Set T7 to underflow at 6 KHz rate
ld    r7,#204 ; (= 3 KHz at pin).
rbit  t7tfn,portph ; Disable beep tone to panel speaker.
rbit  t7stp,pwmdh ; Start T7 running.

slcd:
                ; Set up LCD display.
                ; Requires use of timer T6, so
                ; appears after timer initialization.

                ; First, set up LCD contrast.
ld    lcvs,#x'0A ; Initialize memory image of LCD Voltage
                ; latch, containing RS (PAUX0) bit also.

```

```

ld      portah,lcvs      ; Arbitrary initial contrast level of 5,
                        ; and RS/ (PAUX0/) is high (= "command").
rbit    lcvcclk,portbh   ; Start with LCVCLK low,
sbit    lcvcclk,dirbh   ; (enable output)
sbit    lcvcclk,portbh   ; then high,
rbit    lcvcclk,portbh   ; then low to get it into LCV latch.

                        ; Initialize PNLCLK (Panel "E" signal).
sbit    pnlclk,portbl   ; Start with PNLCLK high
sbit    pnlclk,dirbl    ; (enable output).

                        ; Wait for worst-case command
                        ; execution time (4.9 ms, twice), in case
                        ; a panel command was triggered while
                        ; PNLCLK was floating.
sbit    t6ack,pwmdh     ; Clear T6 PND bit.
ld      t6,#13000      ; Set T6 to twice 4.9 milliseconds.
rbit    t6stp,pwmdh     ; Start timer T6.
lcdipl: ifbit    t6pnd,pwmdh ; Wait for T6 PND bit
                        ; to be set.

jp      lcdgol
jp      lcdipl
lcdgol: sbit    t6stp,pwmdh ; Stop timer T6.
sbit    t6ack,pwmdh     ; Clear T6 PND bit.

                        ; Reset Panel controller (per Hitachi HD44780
                        ; User's Manual).

                        ; (Panel RS signal was set
                        ; in LCD Contrast initialization above,
                        ; so no change needed here to
                        ; flag these as a commands.)

ld      A,#x'38        ; Send "8-Bit Mode, 2 Lines" command: one;
jsrl   wrpnl
ld      A,#x'38        ; two;
jsrl   wrpnl
ld      A,#x'38        ; three;
jsrl   wrpnl
ld      A,#x'38        ; four times.
jsrl   wrpnl
ld      A,#x'08        ; Disable display.
jsrl   wrpnl
ld      A,#x'01        ; Clear display RAM.
jsrl   wrpnl

                        ; Initial default mode settings.

ld      A,#x'06        ; Set mode to move cursor to the right, no
jsrl   wrpnl          ; automatic shifting of display.
ld      A,#x'0E        ; Enable display: non-blinking cursor mode.
jsrl   wrpnl

;      CONTINUES TO MAIN PROGRAM INITIALIZATION
      .form 'Main Program Initialization'

init:
                        ; Once-only initializations.

```

```

ld    curcmd,#x'80    ; Current Command: top bit set means "none".
ld    cpuad,#cpubuf  ; Set CPU command index to beginning of buffer.
ld    numexp,#8      ; Arbitrary starting value.

                                ; Arbitrary set of initialization values for variables,
                                ; in effect until receipt of the first INITIALIZE
                                ; command.

ld    numchr,#0      ; Clear count of characters received.
ld    cadin,#botad   ; Next character in from comm port goes to
                                ; first byte of buffer.
ld    cadout,#botad  ; Next port data character out (to CPU)
                                ; comes from first byte of buffer.
ld    numout,#0      ; No characters being sent to CPU.
ld    cntout,#0      ; No characters being sent to CPU.
ld    pascnt,#125    ; Send to CPU when 125 characters received.
ld    stpcnt,#126    ; Stop host when 126 characters received.
ld    bstat,#0       ; Set buffer ready to receive.
ld    alert,#0       ; No events pending.
ld    ackmd,#1       ; BUSY will fall during ACK/ pulse.
ld    errchr,#55     ; Arbitrary fill for error character.
ld    errfgs,#0      ; Clear error detail flags.
ld    ulow,#x'80     ; Set UART flow control mode byte empty.

runsys:                          ; Enable interrupts, start timers and go to main loop.

sbit  tmsr,enir        ; Enable timer interrupts. (Done here
                                ; to allow certain commands without an
                                ; INITIALIZE command first.)
sbit  13,enir          ; Enable CPU Command interrupt.
sbit  gie,enir         ; Enable interrupt system.

.form 'Main Scan Loop'

; Declarations

vdata = x'10    ; CPU DATA vector number.
vrtc  = x'11    ; Real-Time Clock vector number.
vprime = x'13   ; Centronics INPUT PRIME signal.
vlcdak = x'17   ; Acknowledge finished writing to LCD panel.
vbutton = x'18  ; Pushbutton status change: a button pressed or
                                ; released.
vustat = x'19   ; Change in UART DSR signal, or end of BREAK.
verr   = x'1A   ; Character received with error from UART, or gross
                                ; error condition in buffering or flow control on
                                ; either port.
vuack  = x'1B   ; UART output acknowledge: character sent.
vdiag  = x'1D   ; Diagnostic Interrupt.

mainlp:
                                ; Error Vectors for unimplemented or
                                ; unexpected interrupts.

.ipt  1,hangup    ; NMI: never expected.
.ipt  2,hangup    ; UPI READ READY: never expected.
.ipt  7,hangup    ; EI: never expected.

chkdta:

```

TL/DD/9977-25

```

ld      A,bstat      ; Test state of buffer.
and     A,#x'09      ; Check PASS and CPUBUSY bits.
ifeq    A,#x'01      ; If PASS and not CPUBUSY,
jsrl    snddta       ; then go send a block of data to CPU.

chkalt: ifeq    alert.w,#x'00 ; Check for alert conditions.
        jmp    chkrsp      ; If none, go check for response ready.

        ifbit   artc,alert.b ; Check for RTC interrupt request.
        jsrl    sndrtc     ; If so, then send Real-Time Clock Interrupt.

        ifbit   aprime,alert.b ; Check for Centronics Input Prime signal.
        jsrl    sndprm     ; If so, send Input Prime interrupt.

        ifbit   alcdak,alert.b ; Check for LCD Panel write done.
        jsrl    sndlak     ; If so, then send LCD Acknowledge interrupt.

        ifbit   aflush,alert.b ; Check for Flush Buffer request.
        jsrl    sndfsh     ; If so, then send data in buffer to CPU.

        ifbit   abutton,alerth.b ; Check for a pushbutton change.
        jsrl    sndbtn     ; If so, then report the change to the CPU.

        ifbit   austat,alerth.b ; Check for a UART status change.
        jsrl    sndust     ; If so, then report the change to the CPU.

        ifbit   aerr,alerth.b ; Check for a data error condition.
        jp      cherr
        jp      nocher
cherr:  ifbit   cpubsy,bstat ; Suppress if CPU busy. (CPU needs to
        jp      nocher      ; receive flushed characters first.)
        ifgt   fshlim,#0
        jsrl    sndfsh     ; If a flush is still needed, then do it first.
        jsrl    snderr     ; If so, then report the error to the CPU.
nocher: ; (This line deliberately empty.)

        ifbit   auack,alerth.b ; Check for UART output done.
        jsrl    snduak     ; If so, then send UART-ACKNOWLEDGE interrupt.

        ifbit   adiag,alerth.b ; Check for Diagnostic Interrupt.
        jsrl    snddiag    ; If so, then send interrupt and data.

chkrsp: jmp    chkdta      ; No "responses" defined yet; just close loop.

        .form 'Main: Send Real-Time Clock Interrupt'

; No data transfer; just trigger interrupt and continue.

sndrtc:
        rbit   artc,alert.b ; Clear ALERT bit.
        jsrl    rdwait     ; Check that UPI interface is ready.
        ; If not, loop until it is.

        ld     obuf,#vrtc   ; Load Real-Time Clock vector into OBUF for CPU.
        ret                    ; Return to main loop.

```

; No data transfer; just trigger interrupt and continue.

sndlak:

```

rbit   alcdak,alert.b ; Clear ALERT bit.
jsrl   rdwait          ; Check that UPI interface is ready.
                        ; If not, loop until it is.

ld     obuf,#vlcdak    ; Load LCD-Acknowledge vector into OBUF for CPU.
ret                                         ; Return to main loop.

.form  'Main: Send Pushbutton Status to CPU'
```

sndbtn:

```

jsrl   rdwait          ; Check that UPI interface is ready.
                        ; If not, loop until it is.

ld     obuf,#vbutton   ; Load BUTTON-DATA vector into OBUF for CPU.

jsrl   rdwait          ; Check that UPI interface is ready.
                        ; If not, loop until it is.

rbit   gie,enir        ; *** Begin Indivisible Sequence ***
ld     obuf,swisnt     ; Load Pushbutton Data Byte into OBUF for CPU.
rbit   abutton,alerth.b ; Clear ALERT bit.
sbit   gie,enir        ; *** End Indivisible Sequence ***
ret                                         ; Return to main loop.

.form  'Main: Send Data from Data Buffer to CPU'
```

; Trashes A, B, K (limit), and C flag. May trash X in future.

; Buffer Flush request serviced here.

sndfsh:

```

rbit   aflush,alert.b ; Reset Flush request.
ifeq   numchr,#0       ; If no characters to send, just return,
ret                                         ; else go to Send Data routine.
jmpl   snddta
```

; Automatic Pass condition serviced here.

snddta:

```

ifbit  aerr,alerth.b  ; Check for a communication or buffer error.
jp     chkflm         ; If so, there is a limit on the number of
                        ; characters to send. Investigate further.
jp     snddl          ; Else, go ahead and perform automatic pass.
```

chkflm:

```

ifeq   fshlim,#0      ; Here, a flush limit is in effect due to an
ret                                         ; error condition. Check that the limit is
                        ; non-zero before initiating the pass. If
                        ; zero, then simply return without passing.
```

snddl: jsrl

```

rdwait          ; Check that UPI interface is ready.
                ; If not, loop until it is.
```

```

ld     obuf,#vdata    ; Load DATA vector into OBUF for CPU.
```

```

jsrl   rdwait          ; Check that UPI interface is ready
                        ; (CPU has acknowledged DATA interrupt).
                        ; If not, loop until it is.
```

TL/DD/9977-27


```

rbit   gie,enir           ; Indivisible operation: disable interrupts
                               ; momentarily.
sbit   passng,bstat       ; Indicate data being passed to CPU.
ld     numout,numchr      ; Sample number of characters in buffer.
                               ; This becomes the number of characters to
                               ; transfer,
ifbit  aerr,alerth.b     ; unless there is a flush limit in effect,
ld     numout,fshlim     ; in which case that limit is used.
ld     fshlim,#0         ; Any flush limit is set to zero at this point,
                               ; disabling any data passing until the error
                               ; condition is reported.
                               ; (This does not need to be conditional.)
sbit   gie,enir           ; End indivisible operation: re-enable
                               ; interrupts.
ld     obuf,numout       ; Give number of characters to CPU.
ld     cntout,numout     ; Copy number of characters to temporary
                               ; count location.

ld     B,cadout          ; Initialize for loop below.
ld     K,#topad         ; Establish buffer limit.

sndd1p:                               ; Loop to send characters from data buffer to CPU.

lds    A,[B+].b          ; Load from next byte in buffer, and increment
                               ; address pointer in B.
jp     sndd4             ; If skip occurs (incremented past end
ld     B,#botad         ; of buffer), reset pointer to top of buffer.

sndd4: jsrl rdwait       ; Check that UPI interface is ready.
                               ; If not, loop until it is.

st     A,obuf           ; Give character to CPU.
decsz cntout            ; Check if last character.
jp     sndd1p           ; No: Loop.

                               ; Yes: Update pointers and buffer status.
ld     cadout,B.b       ; Update current pointer address in memory.

rbit   gie,enir         ; *** Begin Indivisible Sequence. ***
and    bstat,#x'FC      ; Clear PASS and PASSING flags.

rbit   pass+4,lcvs      ; (DEBUG: Update PASS in LCD Contrast latch.)
ld     portah,lcvs
sbit   lcvcclk,portbh
rbit   lcvcclk,portbh

sc                                           ; (Set carry for subtraction.)
subc   numchr,numout    ; Adjust number of characters in buffer to
                               ; reflect those just removed.
ld     A,#bufsiz       ; Check whether the buffer is any longer
ifgt   A,numchr        ; completely full.
rbit   full,bstat      ; No: remove FULL indication (if set).
ifgt   A,numchr        ; (DEBUG: update FULL for LCV latch.)
rbit   7,lcvs

ifbit  stop,bstat      ; Check whether host was stopped.
jp     sdstp           ; Yes: continue,
jmpl   sdencl          ; No: terminate indivisible sequence and
                               ; return to main loop.

sdstp: ifgt stpcnt,numchr ; Check whether number of characters is

```

```

                                ; now less than "Stop" value to host.
    jp      sdstpl
    jmpl   sdsend                ; If not, then return to main loop.

sdstpl:  rbit   stop,bstat      ; Clear "Stop Host" flag.
        rbit   5,lcvs

                                ; Check which port to enable for more data.
        ifbit  usel,ups        ; Check if UART is selected.
        jmpl   sdsts          ; If so, go set up flow control.
        ifbit  enprm,cps      ; Check if Centronics port is selected.
        jmpl   sdcsts        ; If so, go set up Centronics BUSY.
        jmpl   sdsend        ; Otherwise, do nothing more and return.

sdcssts: ifbit  clinad,ackmd    ; Check if in Centronics Line Mode. If so,
        jmpl   sdsend          ; the CPU itself must command the ACK action.
        ld     A,bstat        ; Test whether data communication with
                                ; host should be allowed to continue.
        and    A,#x'3C        ; Bits involved are STOP, CPUBSY, IFCBSY and
                                ; FULL.
        ifeq   A,#x'00        ; If no stop conditions are in effect,
        rbit   cbusy,cps      ; clear the BUSY indication in CPS
                                ; (Centronics Port Status) byte in memory.
        ifbit  14,ircd        ; If not between the two interrupt services
                                ; of a Centronics strobe, then
        jsrl   setcen         ; call Centronics port control setup routine,
                                ; to generate ACK/ pulse and clear BUSY.
                                ; (If this sequence does occur between the
                                ; leading and trailing edge interrupts for
                                ; STROBE/, then the trailing edge routine
                                ; will pulse ACK/ when it is allowed to run.)

        jmpl   sdsend

sdusts:  rbit   cus,ups        ; Set UART not busy.
        jsrl   dtron          ; Set DTR handshake appropriately.
        ifbit  eti,enui       ; Check if a UART transmitter interrupt will
                                ; be occurring.
        jmpl   sdsend        ; If so, then no further action is required.
        ifbit  xonb,uflow     ; Otherwise, if XON protocol is in effect,
        jsrl   setuar        ; then check and perform flow control.
        jmpl   sdsend        ; Then exit to main program.

sdsend:  ld     portah,lcvs    ; (DEBUG: Update LCV latch.)
        sbit  lcvcclk,portbh
        rbit  lcvcclk,portbh
        sbit  gie,enir        ; *** End Indivisible Sequence. ***

        ret                  ; Return to main program loop.

        .form 'Main: Send Input Prime interrupt to CPU'

sndprm:  ; Send INPUT PRIME interrupt to CPU.
        rbit  aprime>alert.b  ; Clear ALERT bit.
        jsrl  rdwait          ; Check that UPI interface is ready.
                                ; If not, loop until it is.

        ld   obuf,#vprime    ; Load PRIME vector into OBUF for CPU.
        ret                  ; Return to main program loop.

```

TL/DD/9977-29

```

.form 'Main: Report a UART DSR change or END OF BREAK'
sndust:
  jsrl rdwait ; Check that UPI interface is ready.
            ; If not, loop until it is.

  ld obuf,#vustat ; Load UART-STATUS vector into OBUF for CPU.

  jsrl rdwait ; Check that UPI interface is ready.
            ; If not, loop until it is.

  rbit gie,enir ; * INDIVISIBLE SEQUENCE *
  rbit austat,alerth.b ; Clear ALERT bit.
  ld obuf,ustat ; Load UART Status Byte into OBUF for CPU.
  rbit brkflg,ustat ; Clear END OF BREAK indication.
  sbit gie,enir ; * END INDIVISIBLE SEQUENCE *
  ret ; Return to main loop.

.form 'Main: Report a Data Error Condition to CPU'
snderr:
  ; Send DATA-ERR interrupt to CPU.
  rbit aerr,alerth.b ; Clear ALERT bit.
  jsrl rdwait ; Check that UPI interface is ready.
            ; If not, wait until it is.

  ld obuf,#verr ; Load DATA-ERR vector into OBUF for CPU.
  jsrl rdwait ; Check that UPI interface is ready.
            ; If not, wait until it is.

  ld obuf,errchr ; Give CPU the offending character.
  jsrl rdwait ; Check that UPI interface is ready.
            ; If not, wait until it is.

  ld obuf,errfgs ; Give CPU the error flags.
  ret ; Return to main program loop.

.form 'Main: Send UART Acknowledge interrupt to CPU'
snduack:
  ; Send ACK-UART interrupt to CPU.
  rbit auack,alerth.b ; Clear ALERT bit.
  jsrl rdwait ; Check that UPI interface is ready.
            ; If not, loop until it is.

  ld obuf,#vuack ; Load ACK-UART vector into OBUF for CPU.
  ret ; Return to main program loop.

.form 'Main: Send Diagnostic Interrupt to CPU'
snddiag:
  jsrl rdwait ; Wait for UPI interface ready.
  ld obuf,#vdiag ; Load vector into OBUF for CPU.
  jsrl rdwait ; Wait for UPI interface ready.
  rbit gie,enir ; *** Begin Indivisible Sequence ***
  ld obuf,dsevc ; Transfer Severity Code.
  ld dsevc,#0 ; Clear it.
  ld A,derrc ; Get Error Code.
  ld derrc,#0 ; Clear it.
  rbit adiag,alerth.b ; Clear ALERT bit.
  sbit gie,enir ; *** End Indivisible Sequence ***

```

```

jsrl  rdwait      ; Wait for UPI interface ready.
st     A,obuf     ; Transfer Error Code.
jsrl  rdwait      ; Wait for UPI interface ready.
      ; Remaining bytes will have meaning only for
      ; command errors.
ld     obuf,dbyte  ; Transfer Byte Received.
jsrl  rdwait      ; Wait for UPI interface ready.
ld     obuf,dccmd  ; Transfer Current Command.
jsrl  rdwait      ; Wait for UPI interface ready.
ld     obuf,dqual  ; Transfer Command Count.
ret    ; Return to main program loop.

.form  'UPI (I3) Interrupt: Data from CPU'

.ipt   3,upivr    ; Declare upivr as vector for Interrupt 3.

upivr: ; Write Strobe received from CPU.
push   A         ; Save Context
push   psw

ld     upicsv.b,upic ; Save UPIC register image for LA0 bit test.

ifbit  cmdemp,curcmd ; If expecting first byte of a command,
jmpl   firstc      ; then go process it as such.

ld     A,ibuf      ; If not, input it for entry into cpubuf.

ifeq   A,#x'A5     ; Check for RESET command.
jp     lcrst

ifbit  la0,upicsv.b ; Check for command argument written to proper
      ; address.
jp     lcord      ; If so, go process as a normal argument.
jsrl  hangup      ; If not, process as a FATAL error, generating
      ; !DIAG interrupt.

lcrst: ifbit  la0,upic ; Continue checking for a RESET command.
      jp     lcord
      jmpl   xreset ; If so, go reset the HPC.

lcard: x     A,[cpuad].b ; If not, place it in next available cpubuf
      ; entry.

inc     cpuad
decsz  numexp
jmpl   upwret      ; If not final byte of command, then return.

lastc: ld     A,curcmd ; Else, process current command.
ifbit  getcnt,A.b    ; Check if extended collection is being made.
jp     lastcl       ; If not, then:
sbit   cmdemp,curcmd ; Set command slot available again.
ld     cpuad,#cpubuf ; Reset CPU buffer pointer to beginning.

lastcl: and   A,#x'1F ; Mask off flag bits.
      shl   A         ; Scale by two, and then
      .odd
      jidw ; jump based on command value.
      .ptw  lcinitt,lcseic,lcseiu,illc
      .ptw  illc,illc,illc,illc ; (All these are one-byte commands.)
      .ptw  lcsrst,lcslcv,lcslcd,lcsled
      .ptw  illc,lcsndu,illc,illc
      .ptw  illc,illc

```

; Process INITIALIZE Command.

```

lcinit:      ld      rtevs,#x'01      ; Enable only Real-Time Clock interrupts, but
            ifeq    cpubuf.b,#0      ; disable them again if
            rbit    rtcenb,rtevs     ; the command argument is zero.
            ld      rtcivi,cpubuf.b  ; Put argument into Real-Time
            ; Clock interval.
            ld      rtccnt,cpubuf.b  ; Put argument into Real-Time
            ; Clock count.
            sbit    tltie,tmdl1      ; Enable Timer T1 interrupt, if not already
            ; enabled.
            rbit    tlstp,tmdl1      ; Start timer, if not already running.

            jsrl    lcibuf           ; Initialize buffer parameters.
            ld      alert.w,#0       ; Set no events pending.
            ld      ackmd,#1         ; BUSY will fall during ACK/ pulse.
            ld      errchr,#55       ; Arbitrary fill for error character.
            ld      errfgs,#0        ; Clear error detail flags.
            ld      swlast,#0        ; Set up initial switch values.
            ld      swlsnt,#0        ; (Both current and last sent)

```

; Reset Centronics port: Busy

```

incent:      ld      cps,#x'25       ; Initialize Centronics port status byte
            ; in memory. (Busy, and PRIME interrupt
            ; disabled; otherwise normal.)
            jsrl    setcen           ; Send to Centronics Control Latch.

```

; Reset UART port: Busy

```

inuart:      and     enui,#x'FC       ; Disable UART by clearing enables on
            ; UART-generated interrupts (except EKUI/,
            ; which is connected to INPUT PRIME/.)
            ld      ups,#x'03        ; Flag UART as busy and not selected.
            ld      A,rbuf           ; Clear out spurious characters.
            ld      A,enur           ; Clear out spurious error flags.
            jmp    upwret            ; Return.

```

```

lcibuf:      ; Internal subroutine to initialize buffer status.
            ; Called also from SELECT commands.
            ld      numchr,#0        ; Clear count of characters received.
            ld      cadin,#botad     ; Next character in from comm port goes to
            ; first byte of buffer.
            ld      cadout,#botad    ; Next port data character out (to CPU)
            ; comes from first byte of buffer.
            ld      numout,#0        ; No characters being sent to CPU.
            ld      cntout,#0        ; No characters being sent to CPU.
            ld      bstat,#0        ; Set buffer ready to receive.
            and     lcvs,#x'0F       ; (DEBUG: Initialize LCV latch high bits.)
            ld      portah,lcvs
            sbit    lcvc1k,portbh
            rbit    lcvc1k,portbh
            ret                      ; Return.

```

; Process SELECT-CENT command.

```

lcselc:      and     enui,#x'FC       ; Disable UART by clearing enables on

```

```

                                ; UART-generated interrupts (except EXUI/,
                                ; which is connected to INPUT PRIME/.)
rbit   usel,ups                ; Flag UART not selected.
ifbit  flemp,uflow            ; If valid UART mode exists,
jp     lcsecl
jsrl   dtroff                 ; use it to set DTR to "not ready" state.

lcsecl:  ld    ackmd,cpubuf.b ; Accept ACK/ mode from command buffer.
         ld    pascnt,cpubuf+1.b ; Put "Buffer Pass" value into
                                ; the PASCNT slot.
         ld    stpcnt,cpubuf+2.b ; Put "Host Stop" value into
                                ; the STPCNT slot.
         jsrl  lcibuf         ; Initialize buffer parameters.

primlp:  ifbit  uart,irpd      ; Check to see if INPUT PRIME/ interrupt is
         jp     primlp        ; still asserted. If so, wait here.

         sbit  14,ircd        ; Set up STROBE detector to see leading edge.
         ld    irpd,#x'EF     ; Clear any spurious interrupt triggered by
                                ; polarity change.
         sbit  14,enir        ; Enable interrupts on I4 (STROBE).
         sbit  uart,enir      ; Enable INPUT PRIME/ interrupt (through
                                ; UART vector).
         ld    cps,#x'A9      ; Set Centronics interface byte not busy,
                                ; selected, and all status bits normal.
         jsrl  setcen        ; Clears BUSY signal and generates ACK/ pulse
                                ; according to current mode in ACKMD.
         jmp   upvret        ; Return.

                                ; Process SELECT-UART command.

lcsele:  ld    A,divby.b      ; Process UART baud selection.
         and   A,#x'0F        ; Strip out old baud rate selector.
         st    A,cpubuf+7.b   ; Save (in unused area of the command buffer),
                                ; and start processing new value.
         ifgt  cpubuf.b,#x'08 ; Check if out of range.
         jsrl  hangup
         ld    A,#10
         sc
         subc  A,cpubuf.b     ; Convert to DIVBY field format.
         swap A               ; Place value in correct field.
         or   A,cpubuf+7.b    ; OR with Microwire rate field.
         st    A,divby.b     ; Place back in DIVBY register.

         ld    uframe,cpubuf+1.b ; Get requested frame format.
         and   uframe,#x'07    ; Discard unused bits.
         sbit  b8or9,enu       ; Set 9-bit mode for 8-bit data plus parity.
         ifgt  uframe,#1      ; If 7-bit plus parity, or 8-bit without parity,
         rbit  b8or9,enu       ; then change this setting to 8-bit mode.

         rbit  b2stp,enu       ; Initialize to one Stop bit.
         ifeq  uframe,#3      ; Test for number of Stop bits requested,
         sbit  b2stp,enu       ; and set up UART hardware accordingly.
         ifgt  uframe,#5
         sbit  b2stp,enu

         ld    A,cpubuf+2.b    ; Set up handshaking mode. This also clears
         and   A,#x'0F        ; the FLEMP bit automatically.
         st    A,uflow

```

TL/DD/9977-33

```

ld    pascnt,cpubuf+3.b    ; Put "Buffer Pass" value into
                        ; the PASCNT slot.
ld    stpcnt,cpubuf+4.b    ; Put "Host Stop" value into
                        ; the STPCNT slot.
jsrl  lcibuf                ; Initialize buffer parameters.

ld    cps,#x'25            ; Set up Port A to disable and de-select
                        ; Centronics port, and disable
                        ; INPUT PRIME interrupt.

rbit  clinmd,ackmd        ; Clear the Centronics Line Mode bit.
jsrl  setcen              ; Send to Centronics latch and to Busy flag.
rbit  14,enir             ; Disable Centronics STROBE interrupt.
ld    A,rbuf              ; Clear any pending character before selection.
ld    A,enur              ; Clear any error indications before selection.
sbit  eri,enui           ; Enable receiver interrupt.
rbit  et1,enui           ; Disable transmitter interrupt.
ld    ups,#x'80           ; Set UART port selected, not busy, and
                        ; no characters being sent or waiting to be
                        ; sent.

ld    ustat,#x'01        ; Set DSR ready(will trigger interrupt if not).
sbit  uart,enir          ; Enable UART interrupt.

ifbit  dtrb0,uflow      ; Initialize DTR pin according to new mode.
jp     lcslu1
rbit  dtr,portbl
jp     lcslu2
lcslu1:  sbit  dtr,portbl
lcslu2:

      jmp    upwret        ; Return.

                        ; Process SET-CENT-STS Command.

lcsbst:  ld    cps,cpubuf.b    ; Load Centronics Port Status from byte
                        ; provided by CPU.
      jsrl  setcen          ; Perform ACK/ if new status calls for it.
      jmp    upwret

                        ; Process SET-CONTRAST Command.

lcslcsv: ld    A,cpubuf.b    ; Load LCD Voltage latch (Contrast) from byte
                        ; supplied by CPU.
      comp  A,#x'07        ; (3-bit value is in complemented form.)
      and  A,#x'07        ; Use only lower three bits.
      and  lcvs,#x'F8     ; Clear field in memory image.
      or   lcvs,A.b       ; Merge new field into image.
      ld   portah,lcvs    ; Place on Port A (input to latch).
      sbit  lcvclock,portbh ; Clock latch.
      rbit  lcvclock,portbh
      jmp    upwret

                        ; Process SEND-LCD Command.

lcslcd:  ifbit  getcnt,curcmd    ; Check for first or second collection
      jmp    lcslc1            ; phase.

```

```

lcsic2:                                ; Second phase: begins execution of the LCI
                                        ; command.
ld   lcdbuf.v,cpubuf.v                 ; Copy CPU buffer to LCD string buffer.
ld   lcdbuf+2.v,cpubuf+2.v
ld   lcdbuf+4.v,cpubuf+4.v
ld   lcdbuf+6.v,cpubuf+6.v
ld   lcdsct,lcdnum                      ; Move number of characters to string
                                        ; count byte
inc  lcdsct                             ; (incremented by one because of
                                        ; extra interrupt occurring after
                                        ; last character has been sent).
ld   lcdsix,#lcdbuf                    ; Set string pointer to first byte.
ld   lcdsfg,lcdfgs                      ; Move flag bits to string location.

ld   r6,#x'FFFF                         ; Set up R6 and T6 to trigger string
ld   t6,#0                               ; transfer.
sbit t6tie,pwmdh                         ; Enable timer T6 interrupt.
rbit t6stp,pwmdh                         ; Start timer to trigger (immediate)
                                        ; interrupt from timer T6.

jmpl upvret

lcsicl:                                ; First phase: Prepare to collect up to 8
                                        ; more bytes of command.
ld   lcdfgs,cpubuf.b                    ; Get flag bits supplied by CPU.
ld   lcdnum,cpubuf+1.b                  ; Get character count from CPU.

ld   numexp,lcdnum                       ; Request another collection of
                                        ; data from the CPU (the string of
                                        ; data for the panel).
ld   cpuad,#cpubuf                       ; Reset CPU collection pointer to start
                                        ; of command buffer.
rbit getcnt,curcmd                       ; Declare that it will be the final
                                        ; collection.

jmpl upvret

                                        ; Process SEND-LED Command.

lcsled:    ld   A,cpubuf.b                ; Load LED latch from byte supplied by CPU.
comp      A                                ; (Data goes to LED's in complemented form.)
st        A,portah                        ; Place new value on Port A (input to latch).
sbit     ledclk,portbh                    ; Clock latch.
rbit     ledclk,portbh
jmpl     upvret

                                        ; Process SEND-UART Command.

lcsndu:    ld   uschr,cpubuf.b            ; Queue this character,
sbit     schr,ups                          ; and request transmission at next
                                        ; transmitter interrupt.
ifbit    eti,enui                          ; Check to see if another character is
jmpl     upvret                             ; already being sent (transmitter interrupt
                                        ; enabled).
jsrl     setuar                             ; If not, then call flow control routine to
                                        ; send it.
jmpl     upvret                             ; Return.

```

TL/DD/9977-35


```

.form 'Processing of First Byte of Command (Code)'
; One-byte commands are processed in this section.
; Longer commands are scheduled for collection of
; remaining bytes, and are processed in routines
; above.

firstc:   ld      A,ibuf      ; Get command from UPI port.
ifbit    la0,upicsv.b      ; Check for out-of-sequence condition
; (argument instead of command).
jsrl     hangup           ; If so, process as a FATAL error (previous
; command was too short).

; Processing of RESET command.

ifeq     A,#x'A5         ; Check for RESET command.
jp       xreset
jp       fcord

; This code is entered whenever a RESET
; command is received.

xreset:  ld      obuf,#vdiag ; Present dummy value for CPU,
; (in case a value was already in OBUF),
; and wait for it to be read by CPU.
jsrl     rdwait
ld       A,#0           ; Initialize registers.
st       A,upic.b
st       A,ibuf.w      ; (Actually all of DIRA.)
st       A,dirb.w
st       A,bfun.w
st       A,ircd.b
st       A,portp.w
st       A,sp.w        ; Then, through RESET vector,
st       A,psv.w
ret

; jump to start of program.

; Here, process an ordinary command (not RESET).

fcord:   and     A,#x'1F     ; Use only least-significant 5 bits.
ifgt     A,#x'11         ; Check for command out of range.
jmpl     il1c
st       A,curcmd      ; Save as current command.

shl     A                ; Scale by two, and then
.odd
jidw     ; jump based on command value.
.ptw    fcinit,fcselc,fcselu,il1c
.ptw    fcflsh,fccbsy,fccnby,fcifby
.ptw    fcscst,fcslcv,fcslcd,fcslcd
.ptw    fcbeep,fcndu,fcusts,il1c
.ptw    il1c,il1c

fcinit:  ld      numexp,#1   ; First byte of INITIALIZE command.
; Expects 1 more byte (RTC interval).
jmpl     upwret           ; Return.

fcselc:  ld      numexp,#3   ; First byte of SELECT-CENTRONICS command.

```

```

                                ; Expects 3 more bytes (ACK-Mode, Pass-Count,
                                ; Stop-Count).
jmpl    upwret                ; Return.

fcselu:    ld    numexp,#5      ; First byte of SELECT-UART command.
                                ; Expects 5 more bytes (baud, frame,
                                ; handshake, Pass-Count, Stop-Count)
jmpl    upwret                ; Return.

                                ; Processing of one-byte FLUSH-BUF command.
fcflsh:    sbit  aflush,alert.b ; Set flush request bit in ALERT byte.
sbit     cmdemp,curcmd         ; Set command byte empty (end of command).
jmpl     upwret

                                ; Processing of one-byte CPU-BUSY command.
fccbsy:    sbit  cpubusy,bstat  ; Set CPU Busy bit in BSTAT byte.
sbit     6,lcvs                ; (DEBUG: set also CPU Busy bit in LCV latch.)
ld       portah,lcvs
sbit     lcvclk,portbh
rbit     lcvclk,portbh
sbit     cmdemp,curcmd         ; Set command byte empty (end of command).
jmpl     upwret

                                ; Processing of one-byte CPU-NOT-BUSY command.
fccnby:    rbit  cpubusy,bstat  ; Reset CPU Busy bit in BSTAT byte.
sbit     6,lcvs                ; (DEBUG: reset also CPU Busy bit in LCV latch.)
ld       portah,lcvs
sbit     lcvclk,portbh
rbit     lcvclk,portbh
sbit     cmdemp,curcmd         ; Set command byte empty (end of command).
jmpl     upwret

fcifby:    ; Processing of one-byte SET-IFC-BUSY command.
                                ; This command (one byte) sets the interface busy
                                ; immediately, to stop characters from the external
                                ; system.
sbit     cmdemp,curcmd         ; Set command byte empty (end of command).
ifbit   usel,ups               ; Check if UART is selected.
jmpl    fciby                  ; If so, go set up flow control.
ifbit   enpr,cps               ; Check if Centronics port is selected.
jmpl    fcibyc                 ; If so, go set up Centronics BUSY status.
jsrl    hangup                 ; Otherwise, error. Stop.

fciby:    ; Set UART port busy.
sbit     cus,ups               ; Set UART input port status busy.
jsrl    dtroff                 ; Set DTR handshake appropriately.
ifbit   eti,enui               ; Check if UART transmitter busy.
jp      fcibyl                 ; If so, flow control will happen
                                ; automatically.
ifbit   xonb,uflow             ; If not, then if XON mode is selected,
jsrl    setuar                 ; invoke flow control routine.

fcibyl:    jmp    upwret

                                ; Set Centronics port busy.
fcibyc:    sbit  ifcbsy,bstat  ; Set Interface Busy bit in BSTAT byte.
sbit     cbusy,cps             ; Set BUSY bit in Centronics Port Status byte.
jsrl    setcen                 ; Change Centronics port control latch
                                ; accordingly.
sbit     cmdemp,curcmd         ; Set command byte empty (end of command).
jmpl     upwret

```

```

fcsrst:          ; First byte of SET-CENT-STS command.
              ld      numexp,#1      ; Set up to expect one more byte.
              jmpl    upvret

fcslcv:          ; First byte of SET-CONTRAST command.
              ld      numexp,#1      ; Set up to expect one more byte.
              jmpl    upvret

fcslcd:          ; First byte of SEND-LCD command.
              ld      numexp,#2      ; Set up to expect one more byte.
              sbit    getcnt,curcmd   ; Note extended collection mode in Current
              jmpl    upvret         ; Command byte.

fcsled:          ; First byte of SEND-LED command.
              ld      numexp,#1      ; Send to LED's: Set up to expect one more byte.
              jmpl    upvret

fcbEEP:          ; Process one-byte BEEP command.
              sbit    cmdemp,curcmd   ; No arguments; set CURCMD byte empty.
              sbit    t7tfn,portph   ; Enable beep tone to panel speaker.
              sbit    t0tie,tmdl     ; Enable Timer T0 interrupt.
              ld      beepct,#19     ; Initialize duration count (approximately
              jmpl    upvret         ; 1 second, in units of Timer T0 overflows).

fcsndu:          ; First byte of SEND-UART command.
              ld      numexp,#1      ; Send to UART: Set up to expect one more byte.
              jmpl    upvret

fcusts:          ; Process one-byte TEST-UART command.
              sbit    cmdemp,curcmd   ; No arguments; set CURCMD byte empty.
              sbit    austat,alerth.b ; Force UART Status interrupt.
              jmpl    upvret

illic:  jsrl    hangup              ; Process illegal command codes.

upvret:          ; Return from UPI Write interrupt.
              ; Restore Context
              pop     psw
              pop     A
              reti

.form 'Timer Interrupt Handler'

.ipt 5,tmrint    ; Declare entry point for Timer Interrupt.

tmrint:  push   A              ; Save context.
         push   B
         push   psw

tlpoll:  ifbit  tlpnd,tmdl     ; Poll for Timer T1 interrupt (Real-Time Clock).
         jmpl  tlint          ; If set, go service it.

t6poll:  ifbit  t6pnd,pwdh     ; Poll for Timer T6 interrupt (LCD Panel Timing
         jmpl  t6int          ; Interrupt).

```

```

t0poll:    ifbit    t0pnd,tmmdl    ; Poll for Timer T0 interrupt (Beep Duration).
           jp      t0pdg          ; If set, check the Enable bit; T0 is not
           jp      t0notp        ; always enabled to interrupt when it runs.
t0pdg:    ifbit    t0tie,tmmdl    ; If enable is also set, then go service T0.
           jmpl   t0int
t0notp:                                     ; (This label is deliberately here.)

noint:    jsrl    hangup          ; Error: no legal timer interrupt pending.

           .form 'Timer T1 Interrupt Service Routine'

tlint:    sbit    t1ack,tmmdl    ; Acknowledge T1 interrupt.
           ifbit  rtcenb,rtevs   ; Check if RTC interrupts are enabled.
           jp      t1int1
           jmpl   kbdchk         ; If not, then go check other events.
t1int1:    decsz   rtcnt         ; Decrement interval value.
           jmpl   kbdchk         ; If interval has not elapsed, then go check
                                   ; for other events.
           ld     rtcnt,rtccivl  ; Reload counter value for next interval.
           ifbit  artc,alert.b   ; Check if CPU has received previous interrupt
           jp      t1rerr        ; request; report error if not.
           sbit  artc,alert.b   ; Set Real-Time Interrupt request to main
           jp      kbdchk        ; program.
t1rerr:    sbit    0,dsevc       ; Signal NOTE severity.
           sbit  7,derrc        ; Signal multiple-RTC error.
           sbit  adia,alerth.b  ; Request !DIAG interrupt from main program.

kbdchk:                                     ; Check keyboard switches.
rbit      astts,portbh         ; Enable pushbutton data to Port D.
ld        A,portd              ; Sample pushbutton switches.
sbit      astts,portbh         ; Disable pushbutton data to Port D.
xor       A,#x'FF              ; Complement low-order 8 bits of A.
x         A,swlast             ; Exchange with last sample.
ifeq     A,swlast              ; Check if the data is stable (same as last
                                   ; sample).
           jp      kbint1
           jmpl   dsrchk        ; If not, go check other events.

kbint1:    ifeq     A,swlsnt     ; Check if the data differs from the last
                                   ; pattern sent to the CPU.
           jmpl   dsrchk        ; If not, go check other events.

           st     A,swlsnt      ; Place new pattern in "last sent" location.
           sbit  abutton,alerth.b ; Request "BUTTON-DATA" interrupt to CPU.

dsrchk:                                     ; Check for status of DSR signal if mode selected.
           ifbit  usel,ups       ; Check if UART is selected.
           jp      dsr0
           jmpl   tmochk        ; If not, skip both DSR and BREAK checking.
dsr0:    ifbit  dsrb,uflow      ; Check if DSR input should be checked.
           jp      dsrl
           jmpl   brkchk
dsrl:    ld     A,#x'01         ; Initialize Accumulator to check DSR.
           ifbit  dsr,porti     ; Check current state of DSR pin.
           rbit  0,A            ; Clear LSB of A if DSR pin set.
           st     A,B           ; Register B holds DSR state (1 = DSR Ready).
           ifbit  dsrflg,ustat  ; Check last DSR state given to CPU.
           xor   A,#x'01        ; Toggle LSB of A if set.
           ifbit  0,A           ; If LSB of A is still set, then must send
           jp      dsr2         ; UART-STATUS interrupt to CPU.

```

```

    jmpl   brkchk           ; Else, go check BREAK status.
dsr2:  rbit   dsrflg,ustat   ; Report new state of DSR to CPU.
       ifbit  0,B.b
       sbit   dsrflg,ustat
       sbit   austat,alerth.b ; Request main program to generate !UART-STATUS.

       ifbit  0,B.b           ; Now, enable or disable UART receiver based on
       jp     dsron           ; new DSR state.
dsroff: rbit   eri,enui     ; If DSR is now inactive, disable receiver
       jmpl   brkchk         ; interrupts.
dsron:  ld    A,ups         ; If DSR is now active, check to see whether
       and   A,#x'60       ; receiver may be re-enabled: must test
       ifgt  A,#x'00       ; for BREAK condition and Multiple Character
       jmpl   brkchk         ; Error condition, which disable the receiver
       sbit   eri,enui     ; until a SELECT-UART command. If not
                           ; permanently disabled then re-enable it here.
       ld    A,rbuf        ; Also remove any garbage characters and error
       ld    A,enui        ; indications seen while DSR was inactive.

brkchk: ifbit  brkmd,ups     ; Check whether BREAK has been detected.
       jp     brkmdl
       jmpl   tmochk        ; Go check for other events if not.
brkmdl: ifbit  txd,portbl   ; Check UART data input pin.
       jp     brkmd2       ; If set, BREAK pulse is done.
       jmpl   tmochk        ; Otherwise, go check for other events.
brkmd2: rbit   brkmd,ups    ; Clear BREAK mode in UART Port Status byte.
       sbit   brkflg,ustat ; Set END OF BREAK bit in UART status to CPU.
       sbit   austat,alerth.b ; Request main program to generate !UART-STATUS.

tmochk: ; *** Insert other RTC events here. ***

       jmpl   tarret       ; Return from Timer T1 interrupt.

       .form 'Timer T6 Interrupt Service Routine'

                           ; Timer T6 interrupt routine: sends characters from
                           ; LCD String Buffer to the panel.
t6int: sbit   t6stp,pwmdh   ; Stop timer T6.
       sbit   t6ack,pwmdh   ; Acknowledge T6 interrupt.

       decsz  lcdsct       ; Decrement LCD character count.
       jmpl   t6nxtc       ; If not done, go send another character.

       sbit   alcdak,alert.b ; If done, request main program to send LCD
                           ; Acknowledge interrupt to CPU.
       jmpl   tarret

t6nxtc: ld    A,lcdsfg     ; Get flags byte (for panel RS signal).
       shr   A             ; Shift right, LSB into carry.
       st   A,lcdsfg       ; Store shifted value back.
       sbit   pnrs,lcvs    ; Determine proper state for RS signal from
       ifc   pnrs,lcvs    ; current character's flag (= flag inverted).
       rbit   pnrs,lcvs
       ld    portah,lcvs   ; Send new RS value to LCD Voltage (LCV) latch.
       sbit   lcvclk,portbh ; Clock the latch. RS signal is now valid.
       rbit   lcvclk,portbh

       ld    A,[lcdsix].b ; Get next LCD character from string buffer.
       inc  lcdsix         ; Increment character pointer.
       compl A            ; Complement character, then

```

```

st      A,portah      ; place it on Port A for LCD display.
rbit   pn1clk,portb1 ; Clock it into panel.
sbit   pn1clk,portb1
comp   A              ; Restore A to uncomplemented form for
                        ; test performed below.

ld      t6,#148       ; Set up normal delay time in timer T6
                        ; (120 microseconds).
ifgt   A,#x'03        ; Check whether the longer delay
jp      t6nxt2        ; (4.9 milliseconds) is necessary.
                        ; This happens if RS=0 and the byte sent to
ifnc   ;               ; the panel is a value of hex 03 or less.
ld      t6,#6022      ; If so, change timer to 4.9 milliseconds.

t6nxt2: rbit   t6stp,pwadh ; Start Timer T6 to time out the character.
        jmp   tarrret    ; Return from the interrupt.

        .form 'Timer T0 Interrupt Service Routine'

t0int:  ; Count duration of beep tone. Restore beep signal
        ; to zero and re-enable switch sampling interrupt
        ; when done.
sbit   t0ack,tmd1     ; Acknowledge interrupt from Timer T0.
decsz  beepct         ; Check whether beep time has finished.
jmp    tarrret        ; No: return from interrupt.
rbit   t0tie,tmd1     ; Yes: disable Timer T0 interrupts and
                        ; continue.
and    portph,#x'0F   ; Disable speaker output.
jmp    tarrret        ; Return from interrupt.

                        ; Common return for timer interrupt service routines.
tarrret: pop   psw     ; Restore context.
        pop   B
        pop   A
        reti

        .form 'Centronics Port Interrupt Handler'
;
; Centronics Port Interrupt Handler
; (Pin I4 rising edge)
;
; Note that cadin is an 8-bit quantity; buffer must be
; contiguous within the basepage area.
;

        .ipt 4,cenint

cenint: push   psw     ; Save context.
        push  A
        push  B
        push  K

        ; Decide whether to process leading or trailing edge interrupt.
ifbit  i4,ircd       ; Check polarity of detector.
jmp    cstrbl        ; Leading edge (rising on I4 pin).
jmp    cstrbt        ; Trailing edge (falling on I4 pin).

cstrbl: ; STROBE/ leading edge service routine.

```

TL/DD/9977-41

```

ld      K,#topad      ; Reg. K gets buffer top address.
sbit   astts,portbh  ; Make sure pushbutton buffer is off.
rbit   cdata,portbh  ; Enable Centronics data to Port D.

                ; Test whether there is room for another byte
                ; in the data buffer.
ifbit  full,bstat    ; If FULL bit set,
jmpl   cenerr        ; process this character as an error
                ; (Buffer Overflow).

ld      B,cadin       ; Get current buffer input address.
ld      A,portd       ; Get character.
xs     A,[B+].b       ; Store in table.
jp     cen0           ; If skip,
ld      B,#botad      ; then wrap input pointer to beginning
cen0:  ld      cadin,bl.b ; of buffer; else just increment it.

cen1:  inc     numchr   ; Increment number of characters.
ifgt   pascnt,numchr  ; Check if buffer full enough to send.
jmpl   cenlex        ; No: end of service.

sbit   pass,bstat    ; Yes: indicate buffer ready to pass.
sbit   4,lcvs        ; (DEBUG: report status in LCD Contrast latch.)
ld     portah,lcvs
sbit   lcvclk,portbh
rbit   lcvclk,portbh

ifgt   stpcnt,numchr ; Check if buffer too full for more
                ; host characters.
jmpl   cenlex        ; No: end of service.

sbit   cbusy,cps     ; Yes: set Centronics port status busy.
sbit   stop,bstat   ; set Buffer Status as "STOPPED".
sbit   5,lcvs       ; (DEBUG: report status in LCD Contrast latch.)
ld     portah,lcvs
sbit   lcvclk,portbh
rbit   lcvclk,portbh

ifeq   numchr,#bufsiz ; Check if buffer completely full.
sbit   full,bstat   ; Yes: set condition.

jmpl   cenlex        ; Update Centronics latch and quit.

cenerr:                ; Error handler: invoked if BUSY flag fails to stop
                ; host processor and the HPC's data buffer overflows
                ; as a result.
sbit   cbusy,cps     ; Set busy indication in Centronics Port
                ; Status byte (to keep BUSY asserted to host
                ; when ENCDATA/ signal is removed later).
                ; This should not be necessary except in case
                ; of an internal error in this program.

sbit   7,lcvs        ; (DEBUG: report error in LCD Contrast latch.)
ld     portah,lcvs
sbit   lcvclk,portbh
rbit   lcvclk,portbh

ifbit  aerr,alerth.b ; If an error has already been posted,
jp     cenmer        ; handle as a multiple error.
jmpl   cenler        ; Else, report single error.

```

```

cenmer:      sbit   bufovf,errfgs   ; OR in the buffer overflow condition.
            sbit   errovf,errfgs   ; Update error conditions byte to also report
            rbit   14,enir         ; Disable STROBE interrupt until re-initialized
            ; by CPU.
            jmp    cenlex         ; Return from the interrupt.

cenler:      sbit   aerr,alerth.b   ; Signal an error.
            ld     errfgs,#x'10     ; Report buffer overflow as reason.
            ld     errchr,portd     ; Place character in ERRCHR slot for report to
            ; CPU.
            ld     fshlim,numchr    ; Establish limit on future flushes.
            jmp    cenlex         ; Return from the interrupt.

cenlex:      ; Exit from Centronics STROBE/ leading edge.
            ld     A,cps            ; Prepare to keep BUSY active when ENCDATA/
            sbit   cbusy,A.b       ; is removed.
            st     A,portah        ; Send CPS byte (with BUSY set) to Centronics
            ; status latch.
            sbit   cenclk,portph   ; (Pulse latch strobe.)
            rbit   cenclk,portph
            sbit   cdata,portbh   ; Remove Centronics data enable; loads BUSY
            ; signal with a "1".
            rbit   14,ircd        ; Set I4 strobe pin to trigger on STROBE/
            ; trailing edge.
            ifbit  14,porti       ; Check if strobe has already gone away.
            jmp    cenend        ; If not, just return (no ACK/ pulse).
            ; The "cstrbt" routine will be activated then
            ; whenever STROBE/ goes away, by means of the
            ; I4 interrupt.
            jmp    cstrbt        ; If so, there is a very small possibility
            ; that the interrupt request may have been
            ; lost due to it changing while the polarity
            ; bit in IRCD was being changed above.
            ; Jump to trailing edge service routine
            ; directly from here.

cstrbt:      ; Centronics STROBE/ trailing edge.

            sbit   14,ircd        ; Set up for leading edge detection again.
            ld     irpd,#x'EF     ; Clear interrupt I4, in case the leading edge
            ; routine came directly here. (No hardware
            ; clear of the request occurs in that case.)
            jmp    cenupd        ; Go update Centronics port, with ACK/ pulse
            ; if necessary.

;           Return from interrupt.

;           ; With Centronics Port update.
cenupd:      jsrl   setcen        ; Update Centronics Control signals
            ; from CPS byte.

;           ; Without Centronics Port update.
cenend:      pop    K            ; Restore context from stack and return from
            ; Centronics interrupt.

            pop    B
            pop    A

```

TL/DD/9977-43


```

pop      psw
reti    ; Return from Centronics interrupt.

;
; Subroutine SETCEN.
; Sets up Centronics Port control signals according to CPS byte.
; Generates ACK signal (if called for) according to current
; Centronics timing mode (in ACKMD byte).
; Trashes Accumulator.

setcen:  rbit   cdata,portbh ; Start with ENCDATA/ low, regardless
        ; of previous state.

        ifbit  cbusy,cps    ; Check if BUSY flag should stay set.
        jmp    noack       ; If so, no ACK/ pulse.

        ld     A,ackmd     ; Get ACK/ mode,
        and   A,#x'03     ; and extract the timing field.
        jid   aab,aba,baa ; Branch based on ACK/ timing mode.
        .pt

aab:     ld     portah,cps  ; BUSY low after ACK/ pulse.
        rbit  cack,portah  ; ACK/ falling edge.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        sbit  cack,portah  ; ACK/ rising edge.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        sbit  cdata,portbh ; Load BUSY flag.
        ret

aba:     ld     portah,cps  ; BUSY low during ACK/ pulse.
        rbit  cack,portah  ; ACK/ falling edge.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        sbit  cdata,portbh ; Load BUSY flag.
        sbit  cack,portah  ; ACK/ rising edge.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        ret

baa:     ld     portah,cps  ; BUSY low before ACK/ pulse.
        sbit  cdata,portbh ; Load BUSY flag.
        rbit  cack,portah  ; ACK/ falling edge.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        sbit  cack,portah  ; ACK/ rising edge.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        ret

noack:   ld     portah,cps  ; BUSY high: Set Centronics latch.
        sbit  cenclk,portph ; Pulse CCTLCLK to load latch.
        rbit  cenclk,portph
        sbit  cdata,portbh ; Load Centronics BUSY signal (high).
        ret

        .form 'UART and Input Prime Interrupt Handler'
        .ipt  6,uarint     ; UART Interrupt Vector

```

```

; This interrupt can indicate any of three conditions:
; 1) A character has been sent, and the transmitter
;    is again ready (label "uarout").
; 2) A character has been received (label "uartin").
; 3) A Centronics INPUT PRIME event has been detected
;    (label "uarprm").

uarint:    push    psw
           push    A
           push    B
           push    K
           push    X

           ifbit  usel,ups    ; Check if UART selected.
           jmpl   uarchr     ; If so, go process a character interrupt.
           ifbit  enprm,cps   ; Check if PRIME interrupt enabled
           jmpl   uarprm     ; from Centronics port. If so,
                           ; this means that the Centronics port
                           ; is selected, and it must be a PRIME
                           ; event.
           jsrl   hangup     ; Else, there is an error. Stop.

uarchr:    ifbit  rbfl,enu    ; Check for Receiver interrupt.
           jmpl   uartin     ; Go process input character if so.
           ifbit  tbmt,enu    ; Check for Transmitter interrupt.
           jmpl   uarout     ; Go process output interrupt if so.
           jsrl   hangup     ; Else, there is an error. Stop.

           .form 'UART Output Routine'

uarout:    ; Here, the interrupt is because a character has just
           ; been sent and the transmitter buffer is now empty.
           ifbit  icpu,ups    ; Check if the CPU needs to be informed.
           jmpl   uicpu
           jmpl   unicpu
uicpu:     sbit   auack,alerrh.b ; Request main program to interrupt CPU for
                           ; UART acknowledge.
           rbit   icpu,ups    ; Reset "Interrupt CPU" status on UART.
           jmpl   unicpu     ; Continue processing of interrupt.

unicpu:    ifbit  xonb,uflow  ; If XON mode selected,
           jsrl   setuar     ; check UART handshake status and take any
                           ; appropriate action.
           jmpl   uarret     ; Return.

           .form 'UART Input Routine'

uartin:    ; UART data input routine.

           ld     A,enur     ; Get image of error flags and RBIT9.
           ld     uinchr,rbuf ; Get character.
           st     A,enring   ; Save image of ENUR for further processing.
                           ; Check for hardware-detected errors.
           and    A,#x'CO    ; Mask for error bits (Overrun/Framing).

           ld     X,uinchr   ; Prepare for parity check.

```

TL/DD/9977-45

```

ld      B,#evntbl      ; Initialize B to point to Even Parity table.
x       A,uframe       ; Parity processing depends on selected
                        ; frame format, so branch to proper
                        ; parity processing routine.
jid
.pt     uiod8,uiev8,unopar,unopar
.pt     uiod7,uiev7,uiod7,uiev7

                        ; Processing for 8-bit characters with parity.
uiod8: ld      B,#oddtbl ; For odd processing, change parity table base.
uiev8: x       A,uframe  ; Recover cumulative errors in accumulator.
        ifbit  frm,A.b   ; Check for BREAK condition:  if framing error,
        jp     ufer8
        jp     u8nbrk
ufer8: ifgt   uinchr,#0  ; and data field is all zeroes,
        jp     u8nbrk
        ifbit  rbit9,enring ; and 9th bit also zero,
        jp     u8nbrk
        ifbit  onebrk,ups ; then check if this is the second
        jp     u82brk    ; consecutive BREAK.
        sbit  onebrk,ups ; If not, then flag only the framing error,
        jp     u8dopr    ; and do not report break status yet.
u82brk: sbit   brk,A.b   ; If so, then set Break bit in error image and
        rbit  eri,enui  ; disable UART receiver until re-selected.
        sbit  brkmd,ups ; Also show receiver disabled in UPS byte.
u8nbrk: rbit   onebrk,ups
u8dopr: ifbit  X,[B].b  ; Check parity of 8-bit character. Set "par"
        sbit  par,A.b   ; bit of Accumulator if it would be incorrect
                        ; without parity bit.
        ifbit  rbit9,enring ; Check parity bit for 8-bit character. Toggle
        xor   A,#x'20   ; parity error indication if set.
uinpok: ifeq   A,#x'00  ; Branch based on presence of error.
        jmpl  uingd
        jmpl  uinerc

                        ; Processing for 7-bit characters with parity.
uiod7: ld      B,#oddtbl ; For odd processing, change parity table base.
uiev7: x       A,uframe  ; Recover cumulative errors in accumulator.
        ifbit  frm,A.b   ; Check for BREAK condition:  if framing error,
        jp     ufer7
        jp     u7nbrk
ufer7: ifgt   uinchr,#0  ; and data field is all zeroes (incl. parity),
        jp     u7nbrk
        ifbit  onebrk,ups
        jp     u72brk
        sbit  onebrk,ups
        jp     u7dopr
u72brk: sbit   brk,A.b   ; then set Break bit in error image and
        rbit  eri,enui  ; disable receiver.
        sbit  brkmd,ups ; Also show receiver disabled in UPS byte.
u7nbrk: rbit   onebrk,ups
u7dopr: rbit   7,uinchr  ; Seven-bit data:  clear parity bit in memory.
        ifbit  X,[B].b  ; Perform bit-table lookup:  1 means error.
        jp     uipe7
        jmpl  uinpok
uipe7: sbit   par,A.b   ; Set parity error indication in A.
        jmpl  uinerc

                        ; For 8-bit character frames with no parity:
unopar: x       A,uframe ; Restore frame value to UFRAME, and continue
                        ; (no parity check in these modes).

```

```

        ifbit   frm,A.b           ; Check for BREAK condition:  if framing error,
        jp      uferr
        jp      unbrk
uferr:  ifgt   uinchr,#0         ; and data field is all zeroes (incl. parity),
        jmp    unbrk
        ifbit   onebrk,ups       ; then BREAK condition:  if previous character
        jmp    un2brk
        sbit    onebrk,ups       ; was not a BREAK, then just note this one.
        jp      unobrck
un2brk: sbit    brk,A.b         ; If it was, then set Break bit in error image
        rbit    eri,enui        ; and disable receiver.
        sbit    brkmd,ups       ; Also show receiver disabled in UPS byte.
unbrk:  rbit    onebrk,ups
unobrck: jmp    uinpk

uingd:                ; Here, a good character was received.  Start buffer
                    ; processing.
        ld     A,uinchr         ; Get character again.
        ld     K,#topad        ; Reg. K gets buffer top address.

                    ; Test whether there is room for another byte
                    ; in the data buffer.
        ifbit   full,bstat      ; If FULL bit set,
        jmp    uinerrf         ; process this character as an error
                    ; (Buffer Overflow).

        ld     B,cadin         ; Get current buffer input address.
        xs     A,[B+].b        ; Store character in table.
        jp     uin0            ; If skip,
        ld     B,#botad        ; then wrap input pointer to beginning
uin0:   ld     cadin,bl.b      ; of buffer; else just increment it.

uin1:   inc    numchr          ; Increment number of characters.
        ifgt   pascnt,numchr    ; Check if buffer full enough to send.
        jmp    uinex          ; No: end of service.

        sbit   pass,bstat      ; Yes: indicate buffer ready to pass.
        sbit   4,lcvs          ; (DEBUG: report status in LCD Contrast latch.)
        ld     portah,lcvs
        sbit   lcvclk,portbh
        rbit   lcvclk,portbh

        ifgt   stpcnt,numchr    ; Check if buffer too full for more
                    ; host characters.
        jmp    uinex          ; No: end of service.

        sbit   cus,ups         ; Yes: set UART input port status busy.
        sbit   stop,bstat      ; set Buffer Status as "STOPPED".
        jsrl   dtroff         ; set DTR handshake appropriately.
        ifbit   eti,enui       ; check if UART transmitter busy.
        jp     uin2
        ifbit   xonb,uflow     ; if not, then if XON mode selected,
        jsrl   setuar          ; then invoke flow control routine.
                    ; (otherwise it will happen on next
                    ; UART transmitter interrupt
                    ; automatically).

uin2:   sbit   5,lcvs          ; (DEBUG: report status in LCD Contrast latch.)
        ld     portah,lcvs
        sbit   lcvclk,portbh

```

TL/DD/9977-47

```

rbit    lcvclk,portbh

ifeq    numchr,#bufsiz ; Check if buffer completely full.
sbit    full,bstat    ; Yes: set condition.

jmpl    uinex

uinerc:                                ; Character error handler.
ifbit   aerr,alerth.b ; If an error has already been posted,
jp      uinmce        ; handle as a multiple error.
jmpl    uinlce        ; Else, report single error.

uinmce:    sbit    errovf,errfgs ; Update error conditions byte to also report
; a lost error.
or       errfgs,A.b ; OR in the errors from this character.
sbit    cus,ups    ; Yes: set UART input port status busy.
ifbit   eti,enui   ; check if UART transmitter busy.
jp      uinmce2
ifbit   xonb,uflow ; if not, then if XON mode selected,
jsrl    setuar     ; then invoke flow control routine.
; (otherwise it will happen on next
; UART transmitter interrupt
; automatically).
uinmce2:  jsrl    dtroff ; Remove DTR handshake if flow mode requires it.
rbit    eri,enui   ; Disable UART input interrupt until
; re-initialized by CPU.
sbit    mcead,ups  ; Also flag receiver disabled in UPS byte.
jmpl    uinex     ; Return from the interrupt.

uinlce:
sbit    aerr,alerth.b ; Request CPU interrupt from main program.
st      A,errfgs    ; Report error flags from Accumulator.
ld      errchr,uinchr ; Report error character.
ld      fshlim,numchr ; Establish limit on future flushes.
jmpl    uinex     ; Return from the interrupt.

uinerf:                                ; FULL error handler: invoked if HPC's data buffer
; overflows.

sbit    7,lcvs     ; (DEBUG: report error in LCD Contrast latch.)
ld      portah,lcvs
sbit    lcvclk,portbh
rbit    lcvclk,portbh

ifbit   aerr,alerth.b ; If an error has already been posted,
jp      uinmef       ; handle as a multiple error.
jmpl    uinlef       ; Else, report single error.

uinmef:    sbit    bufovf,errfgs ; Signal buffer overflow as another error.
sbit    errovf,errfgs ; Update error conditions byte to also report
; a lost error.
sbit    cus,ups    ; Set UART input port status busy.
rbit    luss,ups   ; (This is done to force flow control action.)
ifbit   eti,enui   ; Check if UART transmitter busy.
jp      uinme2
ifbit   xonb,uflow ; If not, then if XON mode selected,
jsrl    setuar     ; then invoke flow control routine.
; (otherwise it will happen on next
; UART transmitter interrupt automatically).
uinme2:    jsrl    dtroff ; Remove DTR handshake if flow mode needs it.

```

```

rbit   eri,enui       ; Disable UART input interrupt until
                ; re-initialized by CPU.
sbit   mcmd,ups      ; Also flag receiver disabled in UPS byte.
jmp    uinex         ; Return from the interrupt.

uinlef:  sbit   aerr,alerth.b ; Signal an error.
        ld     errfgs,#x'10   ; Report buffer overflow as reason.
        ld     errchr,uinchr  ; Place character in ERRCHR slot for report to
                ; CPU.
        ld     fshlim,numchr  ; Establish limit on future flushes.
        sbit   cus,ups       ; Set UART input port status busy.
        rbit   luss,ups      ; (This is done to force flow control action.)
        ifbit  eti,enui      ; Check if UART transmitter busy.
        jp     uinlf2
        ifbit  xonb,uflow    ; If not, then if XON mode selected,
        jsrl  setuar         ; then invoke flow control routine.
                ; (otherwise it will happen on next
                ; UART transmitter interrupt automatically).
uinlf2:  jsrl  dtroff        ; Remove DTR handshake if flow mode needs it.
        jmp    uinex         ; Return from the interrupt.

uinex:   ; Exit from UART input character processing.
        jmp    uarret        ; Return.

        ; Parity Bit Lookup Table

evntbl:  .byte  X'96,X'69,X'69,X'96,X'69,X'96,X'96,X'69
        .byte  X'69,X'96,X'96,X'69,X'96,X'69,X'69,X'96
oddtbl:  .byte  X'69,X'96,X'96,X'69,X'96,X'69,X'69,X'96
        .byte  X'96,X'69,X'69,X'96,X'69,X'96,X'96,X'69
        .byte  X'96,X'69,X'69,X'96,X'69,X'96,X'96,X'69
        .byte  X'69,X'96,X'96,X'69,X'96,X'69,X'69,X'96
;
; A one in the table means incorrect parity for the mode,
; the mode being expressed as the base address (evntbl or oddtbl).
        .form  'Centronics INPUT PRIME'

        ; Centronics INPUT PRIME service.
uarprm:  sbit   aprime,alert.b ; Set PRIME bit in Alert mailbox to Main prog.
        sbit   cbusy,cps      ; Set BUSY bit in Centronics status byte.
        jsrl  setcen         ; Go set up Centronics port itself.
        rbit   uart,enir      ; Disable interrupt until it goes away.
        jmp    uarret        ; Return.

uarret:  pop     X            ; Common return from UART interrupt.
        pop    K
        pop    B
        pop    A
        pop    psw
        reti

        .form  'Subroutine to Wait for OBUF Empty'

; RDWAIT subroutine: waits until the CPU has read a byte from the
; UPI interface.

rdwait:  ifbit  rdrdy,upic    ; Check to see if OBUF register is full.

```

TL/DD/9977-49

```

ret
Jp rdwait

.form 'Write to Panel Subroutine'
; Write Panel subroutine.
; Used only at initialization or to report a
; fatal protocol error, since it performs
; the timing delay using timer T6 without interrupts.
; (Panel RS signal must be set up previously in the
; LCV latch by the calling routine.)

wrpnl: comp A ; Complement value for bus.
st A,portah ; Put value on panel bus.
rbit pnlclk,portbl ; Set Panel Clock low,
sbit pnlclk,portbl ; then high again;
; pulse width approx.
; 1.2 microsec.
; Wait for another
; 4.9 milliseconds (twice).
ld t6,#13000 ; Twice 4.9 milliseconds.
rbit t6stp,pwdh ; Start timer T6.
wrplp: ifbit t6pnd,pwdh ; Wait for PND to be set.
Jp wrpgo
Jp wrplp
wrpgo: sbit t6stp,pwdh ; Stop timer T6.
sbit t6ack,pwdh ; Clear T6 PND bit.
ret ; Return from subroutine.

.form 'Set up UART flow control/output'

setuar: ; Subroutine SETUAR: checks status of UART output
; section, and initiates a transfer if needed.
ld A,ups ; Check if UART handshake status needs update.
and A,#x'03
shl A
.odd
jidv
.ptw usmat,usnmat,usnmat,usmat
; Here, UART status last sent does not match
; current status. Needs flow control action.

usnmat:
ifbit cus,ups
jmpl ustop
ugo: ld X,#xon ; Get XON (Control-Q) code.
jsrl uecsnd ; Format it and send.
rbit luss,ups
jmpl sturet ; Return.
ustop: ld X,#xoff ; Get XOFF (Control-S) code.
jsrl uecsnd ; Format it and send.
sbit luss,ups
jmpl sturet ; Return.

usmat: ; No flow control needed. Check if CPU character is
; waiting to be sent.
ifbit schr,ups
jmpl uscpc

```

```

unopnd:                ; Here, no characters pending to be sent.  Turn off
                        ; transmitter interrupt and return.
    rbit   eti,enui    ; Turn off transmitter interrupts.
    jmp    sturet     ; Return.

uscpc:                ; Here, a character is waiting to be sent from CPU.
    ld     X,uschr     ; Get character.
    jsrl  uecsnd      ; Format character for current frame and send.
    rbit   schr,ups    ; Remove character send request.
    sbit   icpu,ups    ; Set CPU interrupt request on completion.
    jmp    sturet     ; Return.

sturet:               ret                ; Return from subroutine.

    .form  'Format and transmit UART character'

uecsnd:                ; Subroutine to encode a character according to the
                        ; currently-selected frame format and send it.
                        ; Character is passed in Register X.
    ld     B,#evntbl
    rbit   xbit9,enu
    ld     A,uframe    ; Jump based on frame format.
    jmp    A
    .pt    su8odd,su8evn,su8,su8
    .pt    su7odd,su7evn,su7odd,su7evn

su8odd:                ld     B,#oddtbl
su8evn:                ifbit  X,[B].b
    sbit   xbit9,enu
    ld     tbuf,X.b
    sbit   eti,enui
    ret

su7odd:                ld     B,#oddtbl
su7evn:                ifbit  X,[B].b
    xor    X.b,#x'80   ; Toggle parity to ignore bad top bit.
    ld     tbuf,X.b
    sbit   eti,enui
    ret

su8:                   ld     tbuf,X.b
    sbit   eti,enui
    ret

    .form  'DTR Handshake Routines'

dtroff:                ; Subroutine DTROFF - Sets printer not ready using DTR.
    ifbit  dtrbl,uflow ; Action taken depends on UFLOW mode.
    jp     doff        ; If DTR is in a permanent state, return.
    ret
doff:                  ifbit  dtrb0,uflow
    jp     d2off
    sbit   dtr,portbl  ; For low-active DTR mode.
    ret
d2off:                rbit   dtr,portbl  ; For high-active DTR mode.
    ret

    ; Subroutine DTRON - Sets printer ready using DTR.
dtron:                ifbit  dtrbl,uflow ; Action taken depends on UFLOW mode.
    jp     dton        ; If DTR is in a permanent state, return.

```

TL/DD/9977-51


```
ret
dton: ifbit dtrb0,uflow
      jp    d2on
      rbit dtr,portbl ; For low-active DTR mode.
      ret
d2on: sbit dtr,portbl ; For high-active DTR mode.
      ret

      .end start
```

TL/DD/9977-52

Interfacing A Serial EEPROM to the National HPC16083

National Semiconductor
Application Note 552
Brian Marley



ABSTRACT

This application note describes how to interface the HPC16083 High-Performance microController to a MICROWIRE™ serial EEPROM (Electrically Erasable Programmable Read-Only Memory) device. The technique uses interrupt-driven scheduling from one of the eight on-chip timers, and so can run in the "background", sharing the HPC gracefully with other control applications running at the same time. Source code is included.

1.0 INTRODUCTION

It is often the case in control-oriented applications that a piece of equipment, on being installed, must be set up with certain semi-permanent configuration mode settings. In the past, jumpers and switches have been the methods used, but in recent years these have been largely supplanted by EEPROM devices, which hold more information and are not prone to mechanical problems. In addition, the presence of an EEPROM allows certain information about the status of the equipment (for example, in printers, a page or character count for monitoring the "age" of the cartridge or print head) to be stored to assist in maintenance.

The most cost-effective type of EEPROM device is one with a serial interface, such as the 256-bit NMC9306 (COP494) or the 1024-bit NMC9345 (COP495). These reside in an

8-pin DIP package, and require only four connections (besides V_{CC} and Ground). These connections are provided by the HPC family of High-Performance Microcontrollers, on a serial port called the MICROWIRE/PLUSTM Interface.

Because one of the HPC's strong suits is Concurrent Control applications (applications in which several control tasks are executing simultaneously, scheduled by interrupts), the code given in this exercise is written to be completely interrupt-driven as well. Instead of timing events with software loops, interrupts from HPC Timer T5 are used both to signal the end of each MICROWIRE transfer and to time the ERASE and WRITE pulse durations for the EEPROM.

2.0 CONNECTIONS AND COMMANDS

The connection between the HPC and the EEPROM device is a completely traditional MICROWIRE connection, as shown in Figure 1. The SI (Serial Input), SO (Serial Output) and SK (Serial Clock) signals of the HPC connect directly to the DO, DI and SK pins of the EEPROM, respectively. The EEPROM's required Chip Select signal (CS: active high) could come from any port bit of the HPC, but the P1 pin of Port P was chosen because Port P pins present zeroes on reset (instead of floating), and this will automatically deselect the EEPROM.

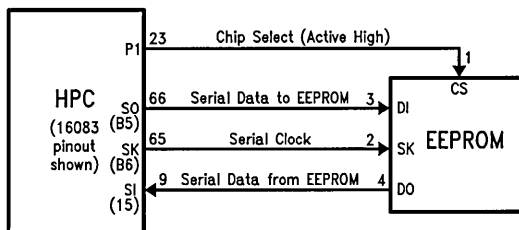


FIGURE 1. MICROWIRE/PLUS Connections

TL/DD/9978-1

To communicate with the EEPROM, the signal CS (pin P1) is set high, and then each 8-bit serial transfer is triggered by writing a value to the HPC's eight-bit SIO register, which is effectively just a shift register. The data placed into the SIO register is shifted out, most-significant bit first, and eight clock pulses are presented on the SK pin corresponding to each shift. Serial data is simultaneously accepted from the SI pin, and at the end of the eight clock pulses the SIO register has been changed to reflect the value presented by the EEPROM (if any). The timing involved in a single MICROWIRE transfer is shown in Figure 2.

While reading from the EEPROM, the value written to SIO doesn't matter, since it is ignored by the EEPROM. The CS signal must be active throughout a command (which may involve more than one eight-bit transfer), and it must be set inactive between commands for at least one microsecond. Also, the time between an ERASE or WRITE command and the following command (as measured by the amount of time the CS signal remains low between them) determines the length of the corresponding ERASE or WRITE pulse within the EEPROM chip. These pulse widths have strict limits which, if exceeded, can damage some EEPROMs.

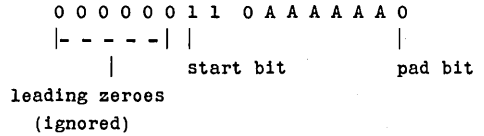
EEPROM commands are 8-bit values. However, they must start with an additional "1" bit (the Start bit), and READ commands require a trailing "pad" bit, to provide timing

control for the access. Since HPC MICROWIRE transfers must consist of integral numbers of 8-bit transfers, at least two such transfers must be used per command.

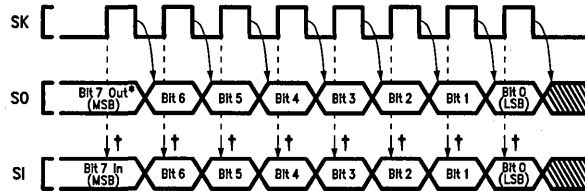
Note that the formats shown below (with 6 address bits) support an EEPROM with up to 1K bits (64 16-bit words). To use a 256-bit EEPROM, one would not specify an address greater than binary 001111, because the two most-significant address bits are ignored by the EEPROM.

2.1 Read Commands

Reading a 16-bit word from the EEPROM is accomplished with a single READ command. For the READ command, the format is:



where the bits marked "A" constitute the address of the EEPROM word to be accessed. These two command transfers are followed by two additional 8-bit transfers, in which the 16 bits of data from the addressed EEPROM word are read by the HPC (most significant bit first).



TL/DD/9978-2

*This bit becomes valid immediately when the transmitting device loads its SIO register. The HPC guarantees it to be valid for at least 1 full SK period before the rising edge of the first SK pulse presented.

† Arrows indicate points at which SI is sampled.

FIGURE 2. MICROWIRE/PLUS Transfer

Master presents eight pulses on SK pin; each pulse transfers one bit in and out.

2.2 Write Commands

To write data into the EEPROM, a sequence of commands is entered:

```

an EWEN command (Erase/Write Enable):
    0 0 0 0 0 0 0 1   0 0 1 1 1 0 0 0 0
an ERASE command:
    0 0 0 0 0 0 0 1   1 1 A A A A A A
    ("A" = Address bits,
     most-significant bit first)
a pause of 16 to 25 milliseconds, with CS
low,
a WRITE command:
    0 0 0 0 0 0 0 1   0 1 A A A A A A
    D D D D D D D D   D D D D D D D D
("A" = Address bits,
 "D" = Data bits,
 most-significant bit first)
a pause of 16 to 25 milliseconds, with CS
low,
and, finally, an EWDS command (Erase/Write
Disable):
    0 0 0 0 0 0 0 1   0 0 0 0 0 0 0 0
  
```

3.0 LISTING AND COMMENTARY

The listing provided shows three necessary segments of a program to access the EEPROM device:

- 1) initialization of the MICROWIRE/PLUS port on the HPC,
- 2) two program fragments of a Main Program which would initiate a Read or a Write operation,
- 3) an interrupt service routine (attached to Timer T5) which actually performs the transfers.

3.1 Initialization

On receiving a Reset signal, the HPC begins execution at the label "start". It loads the PSW register (to select 1 Wait state), and then removes all interrupt enables.

At label "sram", all RAM within the HPC is initialized to zero.

At "suwire", the MICROWIRE/PLUS interface pins are initialized. The MICROWIRE/PLUS interface is then set to the CKI/128 bit rate (125 KHz clocking at 16 MHz crystal frequency). The internal interface is not completely cleared by the Reset signal, so the firmware must set it up and wait (at label "suwlp") for the interface to become ready. Once this has been done, a byte of all zeroes is sent to the EEPROM to terminate any Write operation that might have been in progress when the Reset was received.

At "tminit", the timers T1–T7 are stopped and any interrupts pending from timers T0–T7 are cleared. The individual timer interrupt enables are then cleared.

The program then continues to label "minit", which initializes the variables in the HPC's on-chip RAM to their proper contents.

At label "runsys", the necessary interrupt is enabled (from the timers), and execution continues to the body of the Main Program.

There follow now two fragments of illustrative main program code which can be used to trigger the process of reading and writing the EEPROM.

3.2 Reading

The main program and interrupt routines given here enable reading from one to eight bytes from the EEPROM, starting at the beginning of any word.

At label "rnvr", an EEPROM READ command is constructed from the EEPROM starting address and placed in the variable "nvr cmd". The number of bytes to be transferred is placed in the variable "nvr num". Control is then transferred to the label "nvr x", where Timer T5 is set up to generate scheduling interrupts for reading data from the EEPROM.

The variable "nvr s" indicates the state of an EEPROM access from one interrupt to another: its top bit ("nvravl") shows whether the EEPROM is already being used, bit 6 ("nvrwr") shows whether it is being written or read, and the low-order 4 bits hold a state number, which is used to transfer control to the appropriate code within the Timer T5 interrupt service routine.

On each Timer T5 interrupt (see labels "tmrint", "t5poll", "t5int"), the timer is stopped, a check is made to determine whether the EEPROM is being read or written (T5 interrupts are used for both), and then a multiway branch (jidw) is performed based on the state number in the variable "nvrs". The state number is incremented on each interrupt. On a Read transfer, five states are entered, at the following labels:

- t5rd0 activates the chip select to the EEPROM and initiates the MICROWIRE transfer to send the first byte of a READ command. Timer T5 is started to time out the MICROWIRE transfer.
- t5rd1 sends the second byte of the READ command. Timer T5 is started to time out the MICROWIRE transfer.
- t5rd2 initiates the MICROWIRE transfer to read the first byte of data from the current EEPROM word. Timer T5 is started to time out the MICROWIRE transfer.
- t5rd3 accepts the first byte of the data into the high-order byte of the variable "nword", and initiates the transfer to read the second byte of the current EEPROM word. Timer T5 is started to time out the MICROWIRE transfer.
- t5rd4 accepts the second byte from the EEPROM into the low-order byte of the variable "nword", and then moves the word into the EEPROM string buffer, called "nvrbuf", using a pointer called "nvrptr". It then checks whether the requested number of bytes has been read (by decrementing the "nvrnum" variable). If so, it leaves Timer T5 stopped, disables its interrupt and returns. This would also be the proper place to set a semaphore flag to acknowledge to the main program that the reading is complete. (Code for this is not included here; it would vary from system to system.) If the requested number of bytes has not yet been read, it increments the address field of the READ command in "nvrcmd", resets the state field in "nvrs" to zero, leaves Timer T5 interrupts enabled, and jumps directly to the "t5rd0" routine to continue.

3.3 Writing

At label "wnvr", an EEPROM ERASE command is constructed from the word address supplied by the CPU. The 16-bit value to be written is placed in the variable "nword". As in the READ-NVR command above, the "nvrs" variable is initialized to select the first state of an EEPROM write operation, and Timer T5 is used to provide the interrupts

that schedule the steps. There are 13 states involved in writing a word to the EEPROM, at the following labels:

- t5wr0 activates the chip select signal to the EEPROM, and sends the first byte of an EWEN command to enable ERASE and WRITE commands. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr1 sends the second byte of the EWEN command. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr2 removes the chip select signal briefly (to signal the beginning of a new command), then sends the first byte of an ERASE command. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr3 sends the second byte of the ERASE command, from the variable "nvrcomd". Timer T5 is started to time out the MICROWIRE transfer.
- t5wr4 removes the chip select signal, then sets up the Timer T5 interval to 20 milliseconds, to time the duration of the EEPROM's internal Erase pulse.
- t5wr5 (entered 20 milliseconds after "t5wr4") re-asserts the chip select signal to the EEPROM, and transfers the first byte of a WRITE command. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr6 alters the command in "nvrcmd" to a WRITE command, then transfers it as the second command byte to the EEPROM. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr7 transfers the first byte of data to be written. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr8 transfers the second byte of data to be written. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr9 removes the chip select signal, then sets up the Timer T5 interval to 20 milliseconds, to time the duration of the EEPROM's internal Write pulse.
- t5wr10 (entered 20 milliseconds after "t5wr9") re-asserts the chip select signal to the EEPROM, and transfers the first byte of an EWDS command (Erase/Write Disable). Timer T5 is started to time out the MICROWIRE transfer.
- t5wr11 transfers the second byte of the EWDS command. Timer T5 is started to time out the MICROWIRE transfer.
- t5wr12 removes the chip select signal to the EEPROM, keeps Timer T5 stopped, disables its interrupt, and returns. This would also be the proper place to set a semaphore flag to acknowledge to the main program that the writing is complete. (Code for this is not included here; it would vary from system to system.)

3.4 Source Listing

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345

EEPROM

03-May-88 10:53
PAGE 1

```

1          .title  EEPROM,'HPC-Based Driver for NMC9306/9345'
2
3          ; This code is written to drive either the 256-bit NMC9306 (COP494)
4          ; or the 1024-bit NMC9345 (COP495) MICROWIRE(tm) EEPROM.
5
6          ; NOTE: Timing values assume that the HPC is running at 16MHz
7          ;        crystal frequency.  For correct programming pulse
8          ;        widths, one should not deviate far from this without
9          ;        adjusting the timing constant below.
10
11 4E1F      TIMCON =      19999      ; 20000 counts at 1 usec = 20 msec.
12          ;                ; Timing constant for ERASE and WRITE
13          ;                ; pulse widths.
14

```

TL/DD/9978-3

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Declarations: Register Addresses

EEPROM

03-May-88 10:53
PAGE 2

```

15          .form  'Declarations: Register Addresses'
16
17 00C0      psw      =      x'C0:w      ; PSW register
18 00C8      al       =      x'C8:b      ; Low byte of Accumulator.
19 00C9      ah       =      x'C9:b      ; High byte of Accumulator.
20 00CC      bl       =      x'CC:b      ; Low byte of Register B.
21 00CD      bh       =      x'CD:b      ; High byte of Register B.
22 00CE      xl       =      x'CE:b      ; Low byte of Register X.
23 00CF      xh       =      x'CF:b      ; High Byte of Register X.
24
25 00D0      enir     =      x'D0:b
26 00D2      irpd     =      x'D2:b
27 00D4      ircd     =      x'D4:b
28 00D6      sio      =      x'D6:b
29 00D8      porti   =      x'D8:b
30 00E0      obuf     =      x'E0:b      ; (Low byte of PORTA.)
31 00E1      portah  =      x'E1:b      ; High byte of PORTA.
32 00E2      portb   =      x'E2:w
33 00E2      portbl  =      x'E2:b      ; Low byte of PORTB.
34 00E3      portbh  =      x'E3:b      ; High byte of PORTB.
35 00E6      upic    =      x'E6:b
36 00F0      ibuf     =      x'F0:b      ; (Low byte of DIRA.)
37 00F1      dirah   =      x'F1:b      ; High byte of DIRA.
38 00F2      dirb   =      x'F2:w
39 00F2      dirbl  =      x'F2:b      ; Low byte of DIRB.
40 00F3      dirbh  =      x'F3:b      ; High byte of DIRB.
41 00F4      bfun    =      x'F4:w
42 00F4      bfunl   =      x'F4:b      ; Low byte of BFUN.
43 00F5      bfunh   =      x'F5:b      ; High byte of BFUN.
44
45 0104      portd   =      x'104:b
46 0120      enu     =      x'120:b
47 0122      enul    =      x'122:b
48 0124      rbuf    =      x'124:b
49 0126      tbuf    =      x'126:b
50 0128      enur    =      x'128:b
51
52 0140      t4      =      x'140:w
53 0142      r4      =      x'142:w
54 0144      t5      =      x'144:w

```

TL/DD/9978-4

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
 HPC-Based Driver for NMC9306/9345
 Declarations: Register Addresses

EEPROM

03-May-88 10:53
 PAGE 3

```

55 0146          r5      =      x'0146:w
56 0148          t6      =      x'0148:w
57 014A          r6      =      x'014A:w
58 014C          t7      =      x'014C:w
59 014E          r7      =      x'014E:w
60 0150          pmode   =      x'0150:w
61 0150          pmdl    =      x'0150:b ; Low byte of PMODE.
62 0151          pmdh    =      x'0151:b ; High byte of PMODE.
63 0152          portp   =      x'0152:w
64 0152          portpl  =      x'0152:b ; Low byte of PORTP.
65 0153          portph  =      x'0153:b ; High byte of PORTP.
66 015C          eicon   =      x'015C:w
67
68 0182          t1      =      x'0182:w
69 0184          r1      =      x'0184:w
70 0186          r2      =      x'0186:w
71 0188          t2      =      x'0188:w
72 018A          r3      =      x'018A:w
73 018C          t3      =      x'018C:w
74 018E          divby   =      x'018E:w
75 018E          divbyl  =      x'018E:b ; Low byte of DIVBY.
76 018F          divbyh  =      x'018F:b ; High byte of DIVBY.
77 0190          tmmode  =      x'0190:w
78 0190          tmdl    =      x'0190:b ; Low byte of TMMODE.
79 0191          tmdh    =      x'0191:b ; High byte of TMMODE.
80 0192          tβcon   =      x'0192:b
81
82

```

TL/DD/9978-5

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
 HPC-Based Driver for NMC9306/9345
 Declarations: Bit Positions

EEPROM

03-May-88 10:53
 PAGE 4

```

83                                     .form 'Declarations: Bit Positions'
84
85                                     ; Name      Position      Register(s)
86                                     ; -----
87
88 0000          gie      =      0      ; enir
89 0000          i0       =      0      ; port only
90 0002          i2       =      2      ; enir, irpd, ircd
91 0003          i3       =      3      ; enir, irpd, ircd
92 0004          i4       =      4      ; enir, irpd, ircd
93 0005          tmrs     =      5      ; enir, irpd
94 0006          uart     =      6      ; enir, irpd
95 0007          ei       =      7      ; enir, irpd
96
97 0001          uwmode   =      1      ; ircd
98 0000          uwdone   =      0      ; irpd
99
100 0000          tbmt    =      0      ; enu
101 0001          rbfl    =      1      ; enu
102 0004          b8or9   =      4      ; enu
103 0005          xbit9   =      5      ; enu
104 0002          wakeup  =      2      ; enur
105 0003          rbit9   =      3      ; enur
106 0006          fmerr   =      6      ; enur
107 0007          doeerr  =      7      ; enur
108 0000          etl     =      0      ; enui
109 0001          eri     =      1      ; enui
110 0002          xtclk   =      2      ; enui
111 0003          xrclk   =      3      ; enui
112 0007          b2stp   =      7      ; enui
113
114 0000          wrrdy   =      0      ; upic
115 0001          rdrdy   =      1      ; upic
116 0002          la0     =      2      ; upic
117 0003          upien   =      3      ; upic
118 0004          b8or16  =      4      ; upic
119
120 0000          t0tie   =      0      ; tmdl
121 0001          t0pnd   =      1      ; tmdl
122 0003          t0ack   =      3      ; tmdl

```

TL/DD/9978-6

NSC ASMNPC, Version E2 (Nov 02 15:51 1987)
NPC-Based Driver for NMC9306/9345

EEPROM

03-May-88 10:53
PAGE 5

Declarations: Bit Positions

```

123 0004      t1tie = 4      ; tmdl
124 0005      t1pnd = 5      ; tmdl
125 0006      t1stp = 6      ; tmdl
126 0007      t1ack = 7      ; tmdl
127 0000      t2tie = 0      ; tmdch
128 0001      t2pnd = 1      ; tmdch
129 0002      t2stp = 2      ; tmdch
130 0003      t2ack = 3      ; tmdch
131 0004      t3tie = 4      ; tmdch
132 0005      t3pnd = 5      ; tmdch
133 0006      t3stp = 6      ; tmdch
134 0007      t3ack = 7      ; tmdch
135
136 0000      t4tie = 0      ; pmdl
137 0001      t4pnd = 1      ; pmdl
138 0002      t4stp = 2      ; pmdl
139 0003      t4ack = 3      ; pmdl
140 0004      t5tie = 4      ; pmdl
141 0005      t5pnd = 5      ; pmdl
142 0006      t5stp = 6      ; pmdl
143 0007      t5ack = 7      ; pmdl
144 0000      t6tie = 0      ; pmdch
145 0001      t6pnd = 1      ; pmdch
146 0002      t6stp = 2      ; pmdch
147 0003      t6ack = 3      ; pmdch
148 0004      t7tie = 4      ; pmdch
149 0005      t7pnd = 5      ; pmdch
150 0006      t7stp = 6      ; pmdch
151 0007      t7ack = 7      ; pmdch
152
153 0000      t4out = 0      ; portpl
154 0003      t4tfn = 3      ; portpl
155 0004      t5out = 4      ; portpl
156 0007      t5tfn = 7      ; portpl
157 0000      t6out = 0      ; portph
158 0003      t6tfn = 3      ; portph
159 0004      t7out = 4      ; portph
160 0007      t7tfn = 7      ; portph
161
162 0000      eipol = 0      ; eicon

```

TL/DD/9978-7

NSC ASMNPC, Version E2 (Nov 02 15:51 1987)
NPC-Based Driver for NMC9306/9345

EEPROM

03-May-88 10:53
PAGE 6

Declarations: Bit Positions

```

163 0001      eimode = 1      ; eicon
164 0002      eiack = 2      ; eicon
165
166 0000      txd = 0      ; portbl, dirbl, bfunl
167 0003      t2in = 3      ; portbl, dirbl
168 0005      so = 5      ; portbl, dirbl, bfunl
169 0006      sk = 6      ; portbl, dirbl, bfunl
170
171

```

TL/DD/9978-8

NSC ASMHP, Version E2 (Nov 02 15:51 1987)
 HPC-Based Driver for MMC9306/9345
 Space Declarations

EEPROM

03-May-88 10:53
 PAGE 7

```

172          .form 'Space Declarations'
173 0000          .sect DSECT, BASE, REL
174
175          ;WORD-ALIGNED VARIABLES
176
177 0000          stackb: .dsw 16          ; Space for 16 words.
178 0020          nvrbuf: .dsw 4          ; EEPROM String Buffer.
179 0028          nvrptr: .dsw 1         ; Pointer into EEPROM Data buffer.
180 002A          nvword: .dsw 1        ; Scratch location for gathering EEPROM data as words.
181
182          ;BYTE-ALIGNED VARIABLES
183
184 002C          nvrcmd: .dsb 1         ; Current EEPROM command.
185 002D          nvrnum: .dsb 1        ; Byte count for current EEPROM Read command.
186 002E          nvrs: .dsb 1         ; EEPROM status byte: phase number for sequencing MICROWIRE
187                                     ; transfers.
188
189          ;BIT DEFINITIONS
190
191          ; NVRS byte: Status of EEPROM MICROWIRE transfers.
192          ; Contains phase (step number) of current EEPROM command
193          ; in low-order 4 bits. Top two bits are as follows:
194 0007          nvravl= 7            ; When set, indicates that no EEPROM command is in progress.
195 0006          nvrwr= 6            ; 0 means an EEPROM Read is in progress; 1 means EEPROM Write.
196
197

```

TL/DD/9978-9

NSC ASMHP, Version E2 (Nov 02 15:51 1987)
 HPC-Based Driver for MMC9306/9345
 Code Section

EEPROM

03-May-88 10:53
 PAGE 8

```

198          .form 'Code Section'
199 0000          .sect CSECT, ROM16, REL ; Code space.
200
201 0000 870008C0          start: ld    psw, #x'08          ; Set one WAIT state.
202 0004 970000          ld    enir, #x'00          ; Disable interrupts
203                                     ; individually.
204
205 0007          sram:          ; Clear all RAM locations.
206                                     ; Basepage bank:
207 0007 8000BE          ld    BK, #x'0000, #x'00BE          ; Establish loop base and limit.
208 000A 00          sraml1: clr  A, [B*].w
209 000B E1          xs
210 000C 62          jp    sraml1
211
212                                     ; Non-basepage bank:
213 000D A701C01FE          sraml2: ld  BK, #x'01C0, #x'01FE          ; Establish loop base and limit.
214 0012 00          clr  A, [B*].w
215 0013 E1          xs
216 0014 62          jp    sraml2
217
218 0015          suwire:          ; MICROWIRE setup.
219                                     ; (EEPROM is automatically
220                                     ; deselected on reset, since
221                                     ; Port P is cleared.)
222
223 0015 96F40D          sbit  so, bfunl          ; Enable SO output.
224 0018 96F20D          sbit  so, dirbl
225 001B 96E21E          rbit  sk, portbl          ; Set up SK output.
226 001E 96F20E          sbit  sk, dirbl
227 0021 96F40E          sbit  sk, bfunl
228 0024 96D409          sbit  umode, ircd          ; Set Master Mode.
229 0027 872225018EAB          ld    divby, #x'2225          ; Set MICROWIRE frequency.
230
231 002D 96D210          suwlp: ifbit  uwdone, irpd          ; Wait until MICROWIRE
232 0030 41          jp    snvr1          ; interface ready (UWDONE
233 0031 64          jp    suwlp          ; bit set).
234
235 0032          snvr1:          ; Cancel any EEPROM Write in progress:
236 0032 8601520C          sbit  t5out, portpl          ; Set EEPROM Chip Select active.
237 0036 970006          ld    sio, #0          ; Send a byte of zeroes.

```

TL/DD/9978-10

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for MMC9306/9345
Code Section

EEPROM

03-May-88 10:53
PAGE 9

```

238 0039 960210      suwlp1:  ifbit  uwdone,irpd  ; Wait until MICROWIRE
239 003C 41          jp      snvr2   ; interface ready (UWDONE
240 003D 64          jp      suwlp1  ; bit set).
241 003E B601521C    snvr2:  rbit   t5out,portpl ; Remove EEPROM Chip Select.
242
243 0042 830801928B  tminit: ld      t0con,#x'08
244 0047 8744400190AB ld      tmmode,#x'4440 ; Stop timers T1, T2, T3.
245 004D 8355018EAB  ld      divby,#x'0055 ; MICROWIRE frequency set
246                          ; to CKI/128.
247 0052 87CCC80190AB ld      tmmode,#x'CCC8 ; Clear and disable timer
248                          ; T0-T3 interrupts.
249
250 0058 8744440150AB ld      pwmode,#x'4444 ; Stop timers T4-T7.
251 005E 40          nop                     ; Wait for Pending bits to
252 005F 40          nop                     ; trickle through before clearing them.
253 0060 87CCCC0150AB ld      pwmode,#x'CCCC ; Clear and disable
254                          ; interrupts from all
255                          ; PWM timers.
256
257 0066 87FFFF0146AB ld      r5,#x'FFFF ; No modulus for EEPROM timer.
258

```

TL/DD/9978-11

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for MMC9306/9345
Main Program Initialization

EEPROM

03-May-88 10:53
PAGE 10

```

259                          .form  'Main Program Initialization'
260
261 006C          R      minit:  ld      nvrs,#x'80 ; Set EEPROM available.
262 006C 97802E    R      ld      nvrptr,#nvrbuf ; Set EEPROM pointer to start of buffer.
263 006F B7002028
264
265 0073          runsys:      ; Enable timer interrupts, and go to main.
266
267 0073 96000D    sbit   tmrs,enir ; Enable timer interrupts. (Done here
268                          ; to allow engine commands without an
269                          ; INITIALIZE command first.)
270 0076 960008    sbit   gie,enir  ; Enable interrupt system.
271
272

```

TL/DD/9978-12

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Main Program Fragments

EEPROM

03-May-88 10:53
PAGE 11

```

273                .form 'Main Program Fragments'
274
275                ; These values are declared as constants; more typically they would be
276                ; contained within variables. Note that the pound-sign character must
277                ; then be deleted in the instructions referencing them.
278
279 0000            nvradr = 0          ; EEPROM address: change to suit your application.
280 ABCD           nvrdata = x'ABCD   ; Written data: change to suit.
281 0004           nvrbyt = 4         ; Number of bytes to read (1-8): change to suit.
282
283                ; Read Fragment: reads up to 4 words (8 bytes) from EEPROM.
284
285 0079 0000            rnv:  ld  A,#nvradr      ; Get NVR starting address.
286 0078 993F            and  A,#x'3F          ; Truncate to legal limit.
287 007D E7             shl  A                 ; Create NVR READ command.
288 007E 882C            R    st  A,nvrcmd      ; Place it in memory.
289 0080 9004            ld  A,nvrbyt         ; Get number of bytes requested.
290 0082 882D            R    st  A,nvrnum     ; Save byte count in memory.
291 0084 97002E        R    ld  nvrbuf,#0     ; Set up NVR access status flags:
292                                ; Read transfer in progress, first phase.
293 0087 87002028      R    ld  nvrptr,#nvrbuf   ; Reset buffer pointer to beginning.
294 0088 4E             jmpl nvrax            ; Go start up transfer.
295
296
297                ; Write Fragment: writes one word to EEPROM.
298
299 008C 87ABCD2A      R    wnvr: ld  nvrword,#nvrdata ; Get data word.
300 0090 9000            ld  A,#nvradr      ; Get EEPROM address.
301 0092 993F            and  A,#x'3F          ; Mask it for proper range.
302 0094 882C            R    st  A,nvrcmd      ; Store it in Command byte in memory.
303                                ; (Opcode = 00 at this point.)
304 0096 97402E        R    ld  nvrbuf,#x'40     ; Set up NVR access status flags:
305                                ; Write transfer in progress, first phase.
306 0099 40             jmpl nvrax            ; Go start up transfer.
307
308
309                ; Common routine, performed by both READ and WRITE.
310
311 009A            nvrax:                ; Start interrupts from Timer T5 to schedule
312                                ; accesses to EEPROM.

```

TL/DD/9978-13

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Main Program Fragments

EEPROM

03-May-88 10:53
PAGE 12

```

313 009A 87FFFF0146AB ld  r5,#x'FFFF ; Interrupts are not repetitive; give R5 a
314                                ; high value.
315 00A0 83000144AB    ld  t5,#0      ; Set Timer T5 to interrupt (almost)
316                                ; immediately when started.
317 00A5 8601500C      sbit t5tie,pwmdl ; Enable interrupt from Timer T5.
318 00A9 8601501E      rbit t5stp,pwmdl ; Start Timer T5.
319
320                ; *** One could replace the following instruction with one that
321                ; *** looks for an appropriate semaphore bit to be set, indicating
322                ; *** that the requested operation has been completed. See other
323                ; *** comments beginning with "****".
324
325 00AD 60             jp  .           ; Stops HPC, except for interrupt service.
326
327                ; END OF MAIN PROGRAM FRAGMENTS.
328

```

TL/DD/9978-14

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 13

```

329                                     .form 'Timer Interrupt Handler'
330
331                                     ; The Timer T5 interrupt service routine does all the work. Each
332                                     ; interrupt sequences the next step of the READ or WRITE
333                                     ; operation in progress.
334
335 FFF4 AE00 R .ipt 5,tmrnt ; Declare entry point for Timer Interrupt.
336
337 00AE AFCB tmrnt: push A ; Save context.
338 00B0 AFC0 push ps.w ;
339
340 00B2 B6015015 t5poll: ifbit t5pnd,pwmdl ; Poll for Timer T5 interrupt (EEPROM Timing
341 00B6 41 jmpl t5int ; Interrupt).
342
343 00B7 60 jp . ; Otherwise, error. Stop HPC.
344
345 00B8 B601500E t5int: sbit t5stp,pwmdl ; Stop Timer T5.
346 00BC B601500F sbit t5ack,pwmdl ; Clear interrupt request. (Doing this
347 ; immediately is acceptable here.)
348 00C0 962E16 R ifbit nvrwr,nvrs ; Check whether Read or Write operation is
349 ; is in progress.
350 00C3 9483 jmpl t5wr ; If Write, go perform
351 ; Enable/Erase/Write/Disable operation.
352 00C5 t5rd: ld A,nvrs ; Else, program is reading from EEPROM.
353 00C7 892E R inc nvrs ; Get phase info.
354 00C9 990F R and A,#x'0F ; Increment memory value for next T5 interrupt.
355 00CB E7 shl A ; Extract phase number.
356 00CC 40 ; Jump based on this number.
357 00CD .odd
358 00CE EC jidw
359 00CE 0A00 .ptw t5rd0,t5rd1,t5rd2,t5rd3,t5rd4
360 00D0 1800
361 00D2 2800
362 00D4 3500
363 00D6 4500
364
365 00D8 B601520C t5rd0: sbit t5out,portpl ; Set chip select signal to EEPROM.
366 00DC 970306 ld sio,#x'03 ; Send first part of NVR Read command.
367 ; Format is: 1/10/A5-A0/B ,

```

TL/DD/9978-15

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 14

```

364                                     ; where first bit is start bit (always '1'),
365                                     ; next two bits are operation (10=read),
366                                     ; next 6 bits are EEPROM address,
367                                     ; last bit is "padding" for access time.
368                                     ; This phase sends top two bits of command.
369 00DF 835A0144AB ld t5,#00 ; Set up for interrupt after MICROWIRE transfer.
370 00E4 B601501E rbit t5stp,pwmdl ; Start Timer T5.
371 00E8 B40151 jmpl tmrret ; Return from interrupt.
372
373 00EB 8C2CD6 R t5rd1: ld sio,nvrcmd ; Send second part of NVR Read command (bottom
374 ; eight bits).
375 00EE 835A0144AB ld t5,#00 ; Set up for interrupt after MICROWIRE transfer.
376 00F3 B601501E rbit t5stp,pwmdl ; Start Timer T5.
377 00F7 B40142 jmpl tmrret ; Return from interrupt.
378
379 00FA 9700D6 t5rd2: ld sio,#0 ; Start reading MSB of EEPROM data.
380 00FD 835A0144AB ld t5,#00 ; Set up for interrupt after MICROWIRE transfer.
381 0102 B601501E rbit t5stp,pwmdl ; Start Timer T5.
382 0106 B40133 jmpl tmrret ; Return from interrupt.
383
384 0109 8CD62B R t5rd3: ld nword+1.b,sio ; Accept MSB of EEPROM data to word buffer.
385 010C 9700D6 ld sio,#0 ; Start reading LSB of EEPROM data.
386 010F 835A0144AB ld t5,#00 ; Set up for interrupt after MICROWIRE transfer.
387 0114 B601501E rbit t5stp,pwmdl ; Start Timer T5.
388 0118 B40121 jmpl tmrret ; Return from interrupt.
389
390 011B 8CD62A R t5rd4: ld nword.b,sio ; Accept LSB of EEPROM data to word buffer.
391 011E B601521C rbit t5out,portpl ; Remove EEPROM chip select signal.
392 0122 A82A R A,nword ; Get EEPROM data word.
393 0124 AD28AB R st A,[nvrptr].w ; Store in EEPROM buffer for CPU.
394 0127 A928 R nvrptr ; Increment EEPROM buffer pointer once.
395 0129 8A2D R decsz nvrnum ; Check whether both bytes of the word were
396 ; requested.
397 012B 41 jp t5rdh ; Yes: continue.
398 012C 45 jp t5rddn ; No: done with reading.
399 012D A928 R t5rdh: inc nvrptr ; Increment EEPROM buffer pointer a second time
400 ; (to signal that a whole word was input to
401 ; buffer).
402 012F 8A2D R decsz nvrnum ; Check whether done.
403 0131 4A jp t5rnxt ; No: Initiate another Read command.

```

TL/DD/9978-16

```

404                                     ; Yes: Terminate and pass data to CPU.
405 0132 B601501C      t5rddn: rbit    t5tie,pwmdl    ; Disable Timer T5 interrupts.
406 0136 962E0F      R      sbit      nvravl,nvrs    ; Set NVR available for more commands.
407                                     ;*** Here you'll want to set a semaphore bit saying that the READ
408                                     ;*** transfer is done.
409 0139 840100      R      jmpl     tmrret        ; Return from interrupt.
410
411 013C      t5rnxt:                                     ; Here, more data needs to be read from the
412                                     ; EEPROM. Initiate another read cycle.
413 013C 97002E      R      ld       nvrs,#x'00    ; Set up new transfer phase = 0.
414 013F 892C      R      inc      nvrcmd        ; Increment address field of NVR command.
415 0141 892C      R      inc      nvrcmd        ; (Two increments are needed: field starts
416                                     ; in Bit 1.)
417 0143 962C1F      R      rbit     7,nvrcmd      ; Prevent increments from altering operation
418                                     ; field. This allows addresses to roll over.
419 0146 9581      R      jmpl     t5rd          ; Rather than triggering a Timer T5 interrupt,
420                                     ; just jump to T5 Read interrupt service again.
421
422
423 0148      t5wr:                                     ; EEPROM Write sequence starts here.
424 0148 882E      R      ld       A,nvrs        ; Get phase info.
425 014A 892E      R      inc      nvrs        ; Increment memory value for next T5 interrupt.
426 014C 990F      R      and     A,#x'0F      ; Extract phase number.
427 014E E7      R      shl     A           ; Jump based on this number.
428 014F          R      .odd
429 014F EC      R      jidw
430 0150 1A00      R      .ptw   t5wr0,t5wr1,t5wr2,t5wr3
431 0152 2A00      R      .ptw
432 0154 3600      R      .ptw   t5wr4,t5wr5,t5wr6,t5wr7
433 0156 4B00      R      .ptw
434 0158 5B00      R      .ptw   t5wr8,t5wr9,t5wr10,t5wr11
435 0162 A000      R      .ptw
436 0164 AE00      R      .ptw
437 0166 BD00      R      .ptw
438 0168 C800      R      .ptw   t5wr12
439
440

```

```

435 016A B601520C      t5wr0: sbit   t5out,portpl  ; Set chip select signal to EEPROM.
436 016E 9701D6        ld          sio,#x'01    ; Send start bit of EWEN command.
437 0171 835A0144AB   ld          t5,#90      ; Set up for interrupt at end of MICROWIRE
438                                     ; transfer.
439 0176 B601501E      rbit       t5stp,pwmdl   ; Start timer T5.
440 017A 94C0          jmpl       tmrret       ; Return from interrupt.
441
442 017C 9730D6        t5wr1: ld          sio,#x'30    ; Send body of EWEN command.
443 017F 835A0144AB   ld          t5,#90      ; Set up for interrupt at end of MICROWIRE
444                                     ; transfer.
445 0184 B601501E      rbit       t5stp,pwmdl   ; Start timer T5.
446 0188 94B2          jmpl       tmrret       ; Return from interrupt.
447
448 018A B601521C      t5wr2: rbit       t5out,portpl  ; Remove EEPROM select momentarily to signal
449 018E 40            nop                    ; end of EWEN command, then:
450 018F B601520C      sbit       t5out,portpl  ;
451 0193 9701D6        ld          sio,#x'01    ; Send Start Bit for ERASE command.
452 0196 835A0144AB   ld          t5,#90      ; Set up for interrupt at end of MICROWIRE
453                                     ; transfer.
454 0198 B601501E      rbit       t5stp,pwmdl   ; Start timer T5.
455 019F 949B          jmpl       tmrret       ; Return from interrupt.
456
457 01A1 82C02CDA      R t5wr3: or          nvrCmd,#x'C0  ; Change NVR Command byte to ERASE command.
458 01A5 8C2CD6        R          ld          sio,nvrCmd  ; Send to EEPROM.
459 01A8 835A0144AB   ld          t5,#90      ; Set up for interrupt at end of MICROWIRE
460                                     ; transfer.
461 01AD B601501E      rbit       t5stp,pwmdl   ; Start timer T5.
462 01B1 9489          jmpl       tmrret       ; Return from interrupt.
463
464 01B3 B601521C      t5wr4: rbit       t5out,portpl  ; Remove EEPROM chip select signal, starting
465                                     ; ERASE pulse inside EEPROM.
466 01B7 874E1F0144AB ld          t5,#TIMCON   ; Set up for delay of 20
467                                     ; milliseconds (erase pulse width).
468 01BD B601501E      rbit       t5stp,pwmdl   ; Start timer T5.
469 01C1 9479          jmpl       tmrret       ; Return from interrupt.
470
471 01C3 B601520C      t5wr5: sbit       t5out,portpl  ; Set EEPROM chip select signal again, ending
472                                     ; the ERASE pulse inside EEPROM.
473 01C7 9701D6        ld          sio,#x'01    ; Send Start bit for Write command.
474 01CA 835A0144AB   ld          t5,#90      ; Set up for interrupt at end of MICROWIRE

```

TL/DD/9978-18

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 17

```

475                                     ; transfer.
476 01CF B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
477 01D3 9467                    jmpl  tmrret    ; Return from interrupt.
478
479 01D5 962C1F                    R t5wr6: rbit  7,nvrcmd  ; Create WRITE command in NVR Command byte.
480 01D8 8C2CD6                    R        ld    sio,nvrcmd ; Send to EEPROM.
481 01DB 835A0144AB                ld    t5,#90    ; Set up for interrupt at end of MICROWIRE
482                                     ; transfer.
483 01E0 B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
484 01E4 9456                    jmpl  tmrret    ; Return from interrupt.
485
486 01E6 8C2BD6                    R t5wr7: ld    sio,nvword+1.b ; Send MSB of data to EEPROM.
487 01E9 835A0144AB                ld    t5,#90    ; Set up for interrupt at end of MICROWIRE
488                                     ; transfer.
489 01EE B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
490 01F2 9448                    jmpl  tmrret    ; Return from interrupt.
491
492 01F4 8C2AD6                    R t5wr8: ld    sio,nvword.b  ; Send LSB of data to EEPROM.
493 01F7 835A0144AB                ld    t5,#90    ; Set up for interrupt at end of MICROWIRE
494                                     ; transfer.
495 01FC B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
496 0200 943A                    jmpl  tmrret    ; Return from interrupt.
497
498 0202 B601521C                t5wr9: rbit  t5out,portpl ; Remove EEPROM chip select, starting Write
499                                     ; pulse within EEPROM.
500 0206 874E1F0144AB                ld    t5,#TIMCON ; Set up for delay of 20
501                                     ; milliseconds (write pulse width).
502 020C B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
503 0210 942A                    jmpl  tmrret    ; Return from interrupt.
504
505 0212 B601520C                t5wr10: sbit  t5out,portpl ; Set EEPROM chip select signal, ending Write
506                                     ; pulse within EEPROM.
507 0216 9701D6                    ld    sio,#x'01 ; Send Start bit for EWDS command (Disable
508                                     ; Write/Erase).
509 0219 835A0144AB                ld    t5,#90    ; Set up for interrupt at end of MICROWIRE
510                                     ; transfer.
511 021E B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
512 0222 59                    jmpl  tmrret    ; Return from interrupt.
513
514 0223 9700D6                t5wr11: ld    sio,#x'00 ; Send body of EWDS command.

```

TL/DD/0978-19

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 18

```

515 0226 835A0144AB                ld    t5,#90    ; Set up for interrupt at end of MICROWIRE
516                                     ; transfer.
517 022B B601501E                rbit  t5stp,pwmdl  ; Start timer T5.
518 022F 4C                    jmpl  tmrret    ; Return from interrupt.
519
520 0230 B601521C                t5wr12: rbit  t5out,portpl ; Remove EEPROM chip select signal.
521 0234 B601501C                R        rbit  t5tie,pwmdl  ; Disable Timer T5 interrupts.
522 0238 962E0F                    R        sbit  nvravl,nvrs  ; Set EEPROM Available.
523                                     ;*** Here you'll want to set a semaphore bit saying that the WRITE
524                                     ;*** transfer is done.
525 023B 40                    jmpl  tmrret
526
527 023C 3FC0                tmrret: pop  psw.w      ; Restore context.
528 023E 3FC8                pop    A
529 0240 3E                    reti
530
531 0241                    .end  start

```

TL/DD/0978-20

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 19

```

ah      00C9 Abs Byte
al      00C8 Abs Byte
b2stp   0007 Abs Null
b8or16  0004 Abs Null
b8or9   0004 Abs Null
bfun     00F4 Abs Word
bfunh   00F5 Abs Byte
bfunl   00F4 Abs Byte
bh       00CD Abs Byte
bl       00CC Abs Byte
dirah   00F1 Abs Byte
dirb    00F2 Abs Word
dirbh   00F3 Abs Byte
dirbl   00F2 Abs Byte
divby   018E Abs Word
divbyh  018F Abs Byte
divbyl  018E Abs Byte
doeerr  0007 Abs Null
ei       0007 Abs Null
etack   0002 Abs Null
eicon   015C Abs Word
eimode  0001 Abs Null
eipol   0000 Abs Null
enir    00D0 Abs Byte
enu     0120 Abs Byte
enur    0122 Abs Byte
enur    0120 Abs Byte
eri     0001 Abs Null
eti     0000 Abs Null
frmerr  0000 Abs Null
gie     0000 Abs Null
i0      0000 Abs Null
i2      0002 Abs Null
i3      0003 Abs Null
i4      0004 Abs Null
ibuf    00F0 Abs Byte
ircd    00D4 Abs Byte
irpd    00D2 Abs Byte
lap     0002 Abs Null
minit   006C Rel Null ROM16

```

TL/DD/9978-21

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 20

```

nvradr  0000 Abs Null
nvravl  0007 Abs Null
nvrbuf  0020 Rel Word BASE
nvrbyt  0004 Abs Null
nvrcmd  002C Rel Byte BASE
nvrdata ABCD Abs Null
nvrnum  002D Rel Byte BASE
nvrptr  002B Rel Word BASE
nvrs    002E Rel Byte BASE
nvrwr   0006 Abs Null
nvrx    009A Rel Null ROM16
nword   002A Rel Word BASE
obuf    00E0 Abs Byte
portah  00E1 Abs Byte
portb   00E2 Abs Word
portbh  00E3 Abs Byte
portbl  00E2 Abs Byte
portd   0104 Abs Byte
porti   000B Abs Byte
portp   0152 Abs Word
portph  0153 Abs Byte
portpl  0152 Abs Byte
psw     00C0 Abs Word
pwmch   0151 Abs Byte
pwmcl   0150 Abs Byte
pwmode  0150 Abs Word
r1      0184 Abs Word
r2      0186 Abs Word
r3      018A Abs Word
r4      0142 Abs Word
r5      0146 Abs Word
r6      014A Abs Word
r7      014E Abs Word
rbfl    0001 Abs Null
rbit9   0003 Abs Null
rbuf    0124 Abs Byte
rdrdy   0001 Abs Null
rnvr    0079 Rel Null ROM16
runsys  0073 Rel Null ROM16
sio     0006 Abs Byte

```

TL/DD/9978-22

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for MMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 21

```

sk          0006 Abs Null
snvr1      0032 Rel Null ROM16
snvr2      003E Rel Null ROM16
so         0005 Abs Null
sram       0007 Rel Null ROM16
sraml1     000A Rel Null ROM16
sraml2     0012 Rel Null ROM16
stackb     0000 Rel Word BASE
start      0000 Rel Null ROM16
suwire     0015 Rel Null ROM16
suwlp     002D Rel Null ROM16
suwlp1     0039 Rel Null ROM16
TIMCON     4E1F Abs Null
t0ack      0003 Abs Null
t0con      0192 Abs Byte
t0pnd      0001 Abs Null
t0tie      0000 Abs Null
t1         0182 Abs Word
t1ack      0007 Abs Null
t1pnd      0005 Abs Null
t1stp      0006 Abs Null
t1tie      0004 Abs Null
t2         0188 Abs Word
t2ack      0003 Abs Null
t2in       0003 Abs Null
t2pnd      0001 Abs Null
t2stp      0002 Abs Null
t2tie      0000 Abs Null
t3         018C Abs Word
t3ack      0007 Abs Null
t3pnd      0005 Abs Null
t3stp      0006 Abs Null
t3tie      0004 Abs Null
t4         0140 Abs Word
t4ack      0003 Abs Null
t4out      0000 Abs Null
t4pnd      0001 Abs Null
t4stp      0002 Abs Null
t4tfn      0003 Abs Null
t4tie      0000 Abs Null

```

TL/DD/9978-23

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for MMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 22

```

t5         0144 Abs Word
t5ack      0007 Abs Null
t5int      0008 Rel Null ROM16
t5out      0004 Abs Null
t5pnd      0005 Abs Null
t5poll     0002 Rel Null ROM16
t5rd       00C5 Rel Null ROM16
t5rd0      00D8 Rel Null ROM16
t5rd1      00EB Rel Null ROM16
t5rd2      00FA Rel Null ROM16
t5rd3      0109 Rel Null ROM16
t5rd4      011B Rel Null ROM16
t5rddn     0132 Rel Null ROM16
t5rdh      012D Rel Null ROM16
t5rnxt     013C Rel Null ROM16
t5stp      0006 Abs Null
t5tfn      0007 Abs Null
t5tie      0004 Abs Null
t5wr       0148 Rel Null ROM16
t5wr0      016A Rel Null ROM16
t5wr1      017C Rel Null ROM16
t5wr10     0212 Rel Null ROM16
t5wr11     0223 Rel Null ROM16
t5wr12     0230 Rel Null ROM16
t5wr2      018A Rel Null ROM16
t5wr3      01A1 Rel Null ROM16
t5wr4      01B3 Rel Null ROM16
t5wr5      01C3 Rel Null ROM16
t5wr6      01D5 Rel Null ROM16
t5wr7      01E6 Rel Null ROM16
t5wr8      01F4 Rel Null ROM16
t5wr9      0202 Rel Null ROM16
t6         0148 Abs Word
t6ack      0003 Abs Null
t6out      0000 Abs Null
t6pnd      0001 Abs Null
t6stp      0002 Abs Null
t6tfn      0003 Abs Null
t6tie      0000 Abs Null
t7         014C Abs Word

```

TL/DD/9978-24

NSC ASMHPC, Version E2 (Nov 02 15:51 1987)
HPC-Based Driver for NMC9306/9345
Timer Interrupt Handler

EEPROM

03-May-88 10:53
PAGE 23

```
t7ack 0007 Abs Null
t7out 0004 Abs Null
t7pnd 0005 Abs Null
t7stp 0006 Abs Null
t7tfn 0007 Abs Null
t7tie 0004 Abs Null
tbmt 0000 Abs Null
tbuf 0126 Abs Byte
tminit 0042 Rel Null ROM16
tmndh 0191 Abs Byte
tmndl 0190 Abs Byte
tmnode 0190 Abs Word
tmrint 00AE Rel Null ROM16
tmrret 023C Rel Null ROM16
tmrs 0005 Abs Null
txd 0000 Abs Null
uart 0006 Abs Null
upic 00E6 Abs Byte
upien 0003 Abs Null
uwdone 0000 Abs Null
uwmode 0001 Abs Null
wakeup 0002 Abs Null
wnvr 008C Rel Null ROM16
wrrdy 0000 Abs Null
xbit9 0005 Abs Null
xh 00CF Abs Byte
xl 00CE Abs Byte
xrclk 0003 Abs Null
xtclk 0002 Abs Null
```

**** Errors: 0, Warnings: 0

TL/DD/9978-25

I²C-Bus—Interface with HPC

National Semiconductor
Application Note 561
Hubert Utz



INTRODUCTION

There are many applications in which microcontrollers are used as a central processor. These systems are designed with the following aspects:

- reduce and minimize system costs
- provide system flexibility
- simple connections to other peripheral devices
(no high speed requirements)

A serial bus structure fulfills the above subjects.

The National Semiconductor microcontroller family provides the MICROWIRE/PLUSTM interface as a synchronous serial line to communicate with peripherals.

Another important serial bus is the I²C-Bus (Inter IC-Bus) which was developed by Valvo/Philips. It is mainly used in the customer area. This article describes a simple I²C-Bus interface with National's microcontroller family HPC 16xxx and two different software routines to work the interface:

- a. Softwarepolling
- b. Using the MICROWIRE™ shift register

THE I²C-Bus

The I²C-Bus is a bidirectional two line serial communication bus. The two wires, SDA (serial data) and SCL (serial clock) carry information between the different devices connected to the bus.

The devices can operate either as a receiver or a transmitter, depending on their functions.

The I²C-Bus also supports multimaster mode. Each device has its own 7-bit address.

This address consists commonly of a fixed hardwired part (4 Bits chip intern) and a variable address part (3 Pins of the device).

The I²C-Bus is based on the following definitions:

- TRANSMITTER: the device which sends the data to the serial data line
- RECEIVER: the device which receives the data from the serial data line
- MASTER: the device which starts a transfer, supplies the clock signals and terminates a current transfer cycle
- SLAVE: the device which is addressed by the master
- MULTIMASTER: more than one device can get the master to control the serial data bus and the serial clock bus
- ARBITRATION: if more than one device simultaneously tries to control the bus, a simple arbitration procedure takes place, so that only one device can get the master
- SYNCHRONIZATION: procedure to synchronize the clock signals of two or more devices (slow slaves)

The maximum transmission rate is 100 kbit/s.

The maximum number of devices connected to the bus is limited by the maximum bus capacitance of 400 pF (typical device capacitance 10 pF).

Start-and Stop Conditions

The bus is not busy if both data- and clock lines remain HIGH because there are only two lines available, the start- and stop conditions have special timing definitions between these two lines:

- start conditions: HIGH-to-LOW transition of the data line, while the clock line is in a HIGH state.
- stop conditions: LOW-to-HIGH transition of the data line, while the clock line is in a HIGH state.

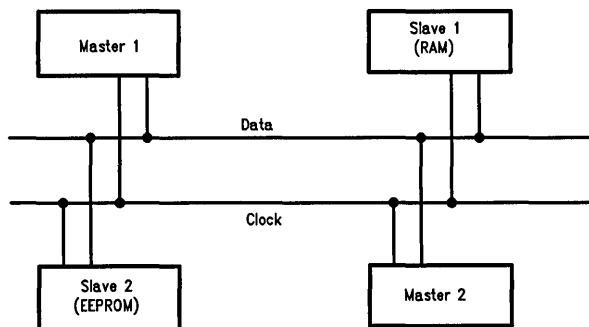
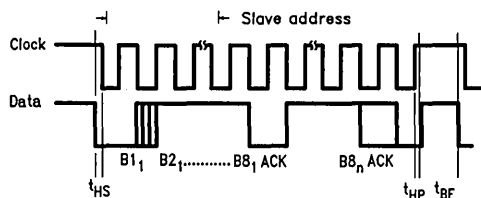


FIGURE 1. I²C-Bus Configurations

TL/DD/10080-1



TL/DD/10080-2

Startcondition

- Clock-, Dataline high (Bus free)
- change Dataline from high to low level
- after $t_{HS} \text{ Min} = 4 \mu\text{s}$ the master supplies the clock

Acknowledge

- transmitting Device releases the Dataline
- the receiving Device pulls the Dataline low during ACK-clock if there is no error
- if there is no ACK the master will generate a Stopcondition to abort the transfer

Stopcondition

- clockline goes high
- after $t_{HP} \text{ Min} = 4.7 \mu\text{s}$ datalines goes high
- the master remains the Data-, Clockline high
- next Startcondition after $t_{FB} \text{ Min} = 4.7 \mu\text{s}$ possible

FIGURE 2. I²C-Bus Timing

The master always generates the start and stop conditions. After the start condition the bus is in the busy state. The bus becomes free after the stop condition.

DATA BIT TRANSFER

After a start condition 'S' one databit is transferred during each clock pulse. The data must be stable during the HIGH-period of the clock. The data line can only change when the clock line is at a LOW level.

Normally each data transfer is done with 8 data bits and 1 acknowledge bit (byte format with acknowledge).

ACKNOWLEDGE

Each data transfer needs to be acknowledged. The master generates the acknowledge clock pulse. The transmitter releases the data line (SDA = HIGH) during the acknowledge clock pulse. If there was no error detected, the receiver will pull down the SDA-line during the HIGH period of the acknowledge clock pulse.

If a slave receiver is not able to acknowledge, the slave will keep the SDA line HIGH and the master can then generate a STOP condition to abort the transfer.

If a master receiver keeps the SDA line HIGH, during the acknowledge clock pulse the master signals the end of data transmission and the slave transmitter release the data line to allow the master to generate a STOP- condition.

ARBITRATION

Only in multi master systems.

If more than one device could be master and more than one wants to access the bus, an arbitration procedure takes place: if a master transmits a HIGH level and another master transmits a LOW level the master with the LOW level will get the bus and the other master will release the bus and the clockline immediately and switches to the slave receiver mode. This arbitration could carry on through many bits (address bits and data bits are used for arbitration).

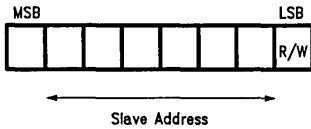
FORMATS

There are three data transfer formats supported:

- master transmitter writes to slave receiver; no direction change
- master reads immediate after sending the address byte
- combined format with multiple read or write transfers (see ...)

ADDRESSING

The 7-bit address of an I²C device and the direction of the following data is coded in the first byte after the start condition:

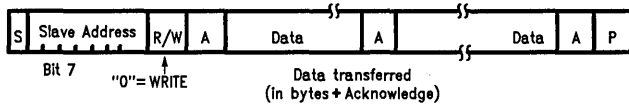


TL/DD/10080-3

A "0" on the least significant bit means that the master will write information to the selected Slave address device; a "1" means that the master will read data from the slave.

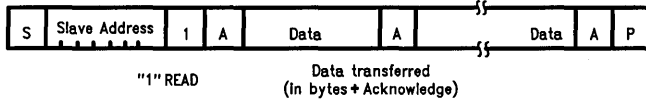
Some slave addresses are reserved for future use. These are all addresses with the bit combinations 1111XXX and 0000XXX. The address 00000000 is used for a general call address, for example to initialize all I²C devices (refer to I²C bus specification for detailed information).

—Master Transmits to Slave, No Direction Change



TL/DD/10080-4

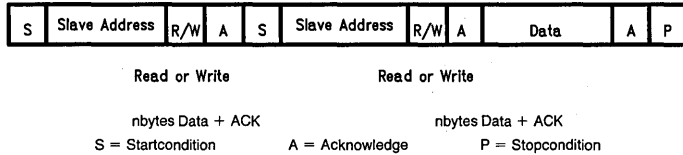
—Master Reads Slave Immediately after First Byte



TL/DD/10080-5

The master becomes a master receiver after first ACK

—Combined Formats



TL/DD/10080-6

FIGURE 3. I²C-Bus Transfer Formats

TIMING

The master can generate a maximum clock frequency of 100 kHz. The minimum LOW period is defined with 4.17 μ s, the minimum HIGH period width is 4 μ s, the maximum rise

time on SDA and SCL is 1 μ s and the maximum fall time on SDA and SCL is 300 ns.

Figure 4 shows the detailed timing requirements.

Symbol	Parameter	Min	Max	Units
f _{SCL}	SCL Clock Frequency	0	100	kHz
t _{BUF}	Time the Bus Must Be Free before a New Transmission Can Start	4.7		μ s
t _{HD} : STA	Hold Time Start Condition. After This Period the First Clock Pulse Is Generated	4.0		μ s
t _{LOW}	The LOW Period of the Clock	4.7		μ s
t _{SU} : STA	Setup Time for Start Condition (Only Relevant for a Repeated Start Condition)	4.7		μ s
t _{HD} : DAT	Hold Time DATA for CBUS Compatible Masters (See Also NOTE, Section 8.1.3.) for I ² C Device	5 0*		μ s μ s
t _{SU} : DAT	Setup Time Data	250		ns
t _R	Rise Time of Both SDA and SCL Lines		1	μ s
t _F	Fall Time of Both SDA and SCL Lines		300	ns
t _{SU} :STO	Setup Time for Stop Condition	4.7		μ s

All values referred to V_{IH Min} = 3.0V and V_{IL Min} = 1.5V levels at 5V supply voltage

*Note that a transmitter must internally provide at least a hold time to bridge the undefined region (max. 300 ns) of the falling edge of SCL.

FIGURE 4. I²C-Bus Timing Requirements

- Two Wire Serial Bus with
 - *Data line
 - *Clock line
- Features:
 - *Multimaster Bus (Master/Slave)
 - *Busarbitration
 - *Transfer rate up to 100 kbits/s
 - *Bytetransfers
 - *Protocols with
 - Start Condition
 - Address
 - Ackn.
 - Data
 - Ackn. (Each Byte)
 - Stop Condition
- *Read, Write, Multiple R/W

FIGURE 5. I²C-Bus Features

The I²C test hardware uses the following components:

- 2 x PC F 8570: 256 x 8-Bit RAM
 - RAM 1: Address: 1010000X
 - RAM 2: Address: 1010010X
- 2 x PC F 8582: 256 x 8 Bit EEPROM
 - EEPROM 1: Address: 1010001X
 - EEPROM 2: Address: 1010011X
- 2 x PC F 8574: Remote 8-Bit I/O Expander
 - I/O 1: Address: 0100000X Used as 8-Bit LED Output Port
 - I/O 2: Address: 0100001X Used as 8-Bit Input Port
- 1 x HCT04: Inverter
- 1 x LS05: Inverter, Open Collector
- 8 x LEDs: Connected Via Pull Up Resistors to Output Pins of PCF 8574
- 2 x Rp: Pull Up Resistors for Clock Line and Data Line
- 8 Switches: Connected Via Pull Up Resistors to Input Pins of PCF 8574
- 1 x Pin Grid Socket: Socket for MOLE™ Connection
- 1 x Power Connector: 5V Power Supply

Four I/O lines of the HPC are used to connect a HPC-MOLE or a HPC-Designer Kit to the I²C-board: SO, SI, P0, and SK. SO drives the data bus line; SDA and P0 drive the clock bus line SCL.

The data on the SDA line is monitored by the input SI and the I²C-bus clock is available at input SK.

SI, SO and SK are μ Wire interface lines.

P0 is used as a continuous timer output during the transfer. All rise and fall times meet the I²C-bus specification. The highest I²C-clock frequency you can get with a 17 MHz HPC oscillator/4 Waitstates is ca. 20 kHz.

I²C-Bus Software HPC

- *Master only Mode
- *Tested I²C-Clock Frequency 16 kHz–20 kHz
- *3 Possible Formats Supported
 - Read
 - Write
 - Multiple Read or Write
- *Two Program Versions
 - Programmed I/O
 - Interrupt Driven I/O
- *Demo Loop:
 - Write Output
 - RAM test
 - Read Input
 - Increment Output or Set Output = Input
- *IRQ Driven Program Uses μ Wire Shift register
- *Message Field:

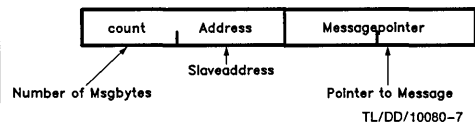
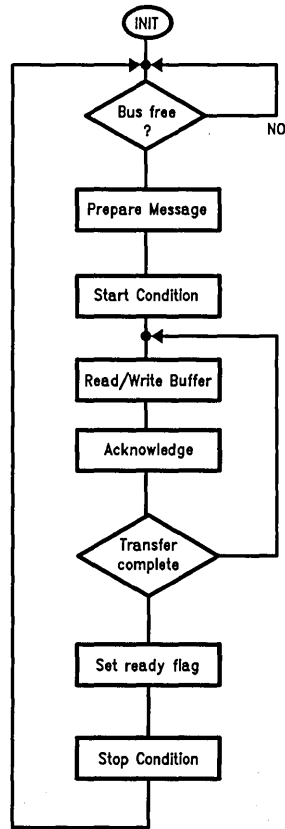


Figure 7. Software Features



TL/DD/10080-8

FIGURE 8. Programmed I/O Flowchart


```

;*****
;* INTER-IC-BUS (I2C-BUS) WITH HPC : APPL.NOTE
;*****

;* 12.10.87                               HU
;* 20.10.87                               HU

.INCLD HPC16083.MAP ; MEMORY MAP FOR HPC16083

;DEFINITIONS
CLK = 32 ;CLOCK LOW/HIGH TIME = 33us

.MACRO SAVE_AB

PUSH A ;SAVE A-REG
PUSH B ;SAVE B-REG

.ENDM

.MACRO SAVE_ABX

SAVE_AB ;SAVE REGS A,B
PUSH X ;SAVE X-REG

.ENDM

.MACRO RESTORE_AB

POP B ;RESTORE B-REG
POP A ;RESTORE A-REG

.ENDM

.MACRO RESTORE_ABX

POP X ;RESTORE X-REG
RESTORE_AB ;RESTORE REGS B,A

.ENDM

.SECT B1, BASE ;DEFINE BASEPAGE SECTION

WBUF1: .DSW 1 ;WORDBUFFER FOR I2C-TABLE
WBUF2: .DSW 1 ;WORDBUFFER FOR I2C-TABLE
INDEX: .DSW 1 ;POINTER TO THE NEXT TABLEENTRY
STATUS: .DSB 1 ;STATUSBYTE
DUMMY: .DSB 1 ;DUMMY BYTE
WPORT: .DSB 1 ;8-BIT VALUE WRITE TO PORT0
RPORT: .DSB 1 ;8-BIT VALUE READ FROM PORT1
WRBUFF: .DSB 10 ;RAM WRITE BUFFER
RDBUFF: .DSB 10 ;RAM READ BUFFER

.ENDSECT

.SECT I2C,ROM16

```

TL/DD/10080-9

```

;*****
;* INIT : THIS SUBROUTINE INITIALIZES TIMER T4, TIMER T5
;*        AND THE uWIRE-INTERFACE TO OPERATE AS I2C-BUS
;*
;* INPUT : NONE
;* OUTPUT : NONE
;* USED REGS : A, B, X ( ALL REGS ARE SAVED )
;*****
INIT:  SAVE_ABX          ;SAVE REGS A,B,X
      LD X,#PWMODE      ;ADDR. PWMODE-REG -> X
      LD B,#PORTP      ;ADDR. PORTP-REG -> B
      LD A,#0C          ;VALUE TO STOP TIMER
      ST A,[X].B        ;STOP T4, NO IRQ, ACK TIP-FLAG
      NOP
      ST A,[X].B        ;MAKE SHURE TIP-FLAGS ARE CLEARED
      LD A,#01          ;DISABLE TOGGLE AND SET OUTPUT HIGH
      ST A,[B].B        ;ON PINS P0 AND P1
      SBIT 3,[B].B      ;TOGGLE ON AT P0
      LD T4.W,#CLK-1    ;LOAD T4 (33us)
      LD R4.W,#CLK-1    ;LOAD R4 (33us)
      SBIT 5,PORTB.B    ;DATALINE OUTPUT = HIGH
      SBIT 5,DIRB.B     ;B5 = OUTPUT
      RBIT 5,BFUN.B     ;NO ALTERNATE FUNCTION SELECTED
      RBIT 6,DIRB.B     ;B6 = INPUT
      RBIT 6,BFUN.B     ;
      RESTORE_ABX      ;RESTORE REGS X,B,A
      RET

RWI2C: PUSH K          ;SAVE REG K
      SAVE_ABX        ;SAVE-REGISTER
RWRST: LD B,#PORTB     ;ADDRESS OF PORTB -> REG B
      LD STATUS.B,#0   ;RESET STATUSBYTE
      JSR TSTBUS       ;BUS FREE ?
      IFC
      JP RWERR         ;IF ERROR -> EXIT
      LD A,[X+].W      ;GET 2 BYTES OF TABLE
      ST A,WBUF1.W     ;SAVE TABLE CONTENTS
      IFBIT 0,A        ;TEST RECEIVE/TRANSMIT BIT
      JP RWRECV        ;BIT = 1 -> RECEIVE
      RBIT 0,STATUS.B  ;STATUS = TRANSMIT
      JP RW01          ;CONTINUE AT RW01
RWRECV: SBIT 0,STATUS.B ;STATUS = RECEIVE
RW01:  SBIT 1,STATUS.B  ;STATUS = FIRST BYTE
      LD A,[X+].W      ;GET NEXT 2 BYTES OF TABLE
      ST A,WBUF2.W     ;SAVE TABLE CONTENTS
      LD A,X
      ST A,INDEX.W     ;SAVE INDEX
      LD A,[X].W       ;GET NEXT WORD OF TABLE
      IFEQ A,#0        ;ANY MORE TO TRANSFER
      JP RW02          ;NO, EXIT
      SBIT 3,STATUS.B  ;STATUS = MULTISTART
RW02:  SBIT 7,STATUS.B  ;STATUS = BUSY

```

TL/DD/10080-10

```

RC                                ;CLR CARRY = NO ERROR

JSR    STRTCD                    ;STARTCONDITION
IFC
JP     STPERR
LD     X,WBUF2.W                ;X = BUFFERINDEX
LD     A,WBUF1.W                ;A.B = ADDRESS
JSR    TRANSF                    ;TRANSMITT 1.BYTE = ADDRESS
IFC
JP     STPERR
DECSZ WBUF1+1.B                ;DECREMENT BYTECOUNT
NOP                                         ;DUMMY FOR DECSZ
IFBIT 0,STATUS.B                ;STATUS = RECEIVE ?
JP     RCVE                      ;YES -> RCVE
TRMIT: LD     A,[X+].B            ;GET NEXT BYTE
DECSZ WBUF1+1.B                ;DECREMENT BYTECOUNT
JP     TRM1                      ;BYTECOUNT <> 0
TRM1:  SBIT 2,STATUS.B            ;FLAG LAST BYTE
JSR    TRANSF                    ;SEND BYTE
IFC                                         ;TEST ERROR
JP     STPERR                    ;ERROR DETECTED
IFBIT 2,STATUS.B                ;LAST BYTE ?
JP     TRM2                      ;YES
JP     TRMIT                    ;NO -> TRANSFER AGAIN
TRM2:  IFBIT 3,STATUS.B            ;MULTISTART ?
JP     TRM3                      ;YES
JP     TRM6                      ;NO -> STOPCONDITION
TRM3:  SBIT 5,[B].B                ;DATA LINE = HIGH
TRM4:  IFBIT 6,[B].B                ;WAIT UNTIL CLOCKLINE = HIGH
JP     TRM5                      ;CLOCK = HIGH
JP     TRM4                      ;CLOCK = LOW
TRM5:  SBIT 2,PWMODE.B            ;STOP TIMER 4
LD     T4.W,#2*CLK-1            ;SET START TIME
LD     X,INDEX.W                ;GET NEXT TABLENTRY
JP     RWRST                    ;PERFORM NEXT STARTCONDITION

TRM6:  JP     RWEND
RCVE:  DECSZ WBUF1+1.B            ;DECREMENT BYTECOUNT
JP     RCV1                      ;BYTECOUNT <> 0
SBIT 2,STATUS.B                ;FLAG LAST BYTE
RCV1:  JSR    RECEIV                ;GET 1 BYTE
ST     A,[X].B                  ;PUT BYTE TO BUFFER
INC    X                        ;X += 1
IFC                                         ;ERROR ?
JP     STPERR                    ;YES -> STOPCONDITION
IFBIT 2,STATUS.B                ;LAST BYTE FLAGGED ?
JP     TRM2                      ;YES, CHECK MULTISTART
JP     RCVE                      ;GET NEXT BYTE
RWEND: RC                        ;NO ERROR
STPERR: JSR    STOPCD            ;STOPCONDITION

RWERR: RESTORE ABX                ;RESTORE REGISTER
POP    K                        ;RESTORE REG K
RET

```

TL/DD/10080-11

```

STRICD: IFBIT 5,PORTI.B ;TEST DATALINE
        JP STRT01 ;IF HIGH -> CONTINUE
STRTER: SC ;ELSE ERROR
        RET
STRT01: IFBIT 6,[B].B ;TEST CLOCKLINE
        JP STRT02 ;IF HIGH -> CONTINUE
        JP STRTER ;ELSE ERROR
STRT02: RBIT 5,[B].B ;DATALINE = LOW
        AND PWMODE.B,#0FB ;START TIMER 4
STRT03: IFBIT 6,[B].B ;WAIT UNTIL CLOCK = LOW
        JP STRT03
        RC ;SIGNAL NO ERROR
        RET
TRANSF: IFBIT 7,A ;TEST FOR THE NEXT DATA
        JP TRNF1 ;PUT DATALINE HIGH
        RBIT 5,[B].B ;PUT DATALINE LOW
        JP TRNF2
TRNF1: SBIT 5,[B].B ;PUT DATALINE HIGH
TRNF2: SWAP A
        SWAP A ;EXCHANGE LOWER/HIGHER BYTE
        LD K,#8 ;SET LOOP COUNT
        SHL A ;DUMMY SHIFT
        JP TRF2 ;JUMP INTO THE LOOP
TRF1: SHL A ;SHIFT MSB -> CARRY
        IFC
        SBIT 5,[B].B ;DATALINE = HIGH
        IFNC
        RBIT 5,[B].B ;DATALINE = LOW
TRF2: IFBIT 6,[B].B ;WAIT UNTIL CLOCK HIGH
        JP TRF3 ;CLOCK = HIGH
        JP TRF2 ;CLOCK = LOW
TRF3: IFBIT 6,[B].B ;WAIT UNTIL CLOCK = LOW
        JP TRF3 ;CLOCK = HIGH
        DECSZ K ;DECREMENT LOOP COUNT
        JP TRF1 ;NEXT BIT
        JSR GETACK ;LOOK FOR ACKNOWLEDGE
        RET
SETACK: RBIT 5,[B].B ;DATALINE = LOW
        RC
GETACK: SBIT 5,[B].B ;DATALINE = HIGH
        RC
ACK01: IFBIT 6,[B].B ;WAIT UNTIL CLOCK = HIGH
        JP ACK02 ;CLOCK = HIGH
        JP ACK01 ;CLOCK = LOW , WAIT
ACK02: IFBIT 5,PORTI.B ;TEST DATALINE
        SC ;FLAG EROR IF HIGH
ACK03: IFBIT 6,[B].B ;WAIT UNTIL CLOCK = LOW
        JP ACK03
        SBIT 5,[B].B ;DATALINE = HIGH
        RET

```

TL/DD/10080-12

```

RECEIV: PUSH    K           ;SAVE REG K
        LD      K, #8      ;SET LOOP COUNT

REC1:   RC
REC2:   IFBIT   6, [B].B   ;WAIT UNTIL CLOCK HIGH
        JP      REC3      ;CLOCK = HIGH
        JP      REC2      ;CLOCK = LOW
REC3:   IFBIT   5, PORTI.B ;TEST DATALINE
        SC
        RLC      A        ;IF HIGH SET CARRY
        RLC      A        ;ROTATE LEFT WITH CARRY
REC4:   IFBIT   6, [B].B   ;WAIT UNTIL CLOCK = LOW
        JP      REC4      ;CLOCK = HIGH
        DECSZ   K        ;DECREMENT LOOP COUNT
        JP      REC1      ;NEXT BIT
        IFBIT   2, STATUS.B ;LAST BYTE FLAGGED ?
        JP      REC5      ;YES, NO ACKNOWLEDGE
        JSR     SETACK    ;SET ACKNOWLEDGE
REC5:   JSR     GETACK    ;LOOK FOR ACKNOWLEDGE
REC6:   POP     K        ;RESTORE REG K
        RET

STOPCD: RBIT    5, [B].B   ;DATALINE = LOW
STOP01: IFBIT   6, [B].B   ;WAIT UNTIL CLOCK = HIGH
        JP      STOPO2    ;CLOCK = HIGH -> STOPO2
        JP      STOPO1    ;WAIT
STOP02: SBIT    2, PWMODE.B ;STOP TIMER 4
        LD      T4.W, #CLK-1 ;INITIALIZE T4 TO STARTCONDITION
        RBIT    7, STATUS.B ;STATUS = I2CBUS NOT BUSY
        SBIT    5, [B].B   ;PERFORM STOPCONDITION
        RET

TSTBUS: RC
        IFBIT   5, PORTI.B ;TEST DATALINE
        JP      TST1
        SC
TST1:   IFBIT   6, [B].B   ;TEST CLOCKLINE
        JP      TST2
        SC
TST2:   RET

RESET:  LD      ENIR.B, #0  ;DISABLE ALL INTERRUPTS
        LD      PWMODE.W, #0CCCC ;STOP AND CLEAR ALL TIMERS
        LD      TMMODE.W, #0CCCC

START:  JSR     INIT
        LD      B, #WRBUFF
        LD      K, #WRBUFF+8 ;CLEAR 9 BYTES

START0: CLR     A
        XS     A, [B+].W
        JP     START0
        LD     RDBUFF.B, #0 ;SET READADDRESS TO 0

        LD     WPORT.B, #0FF ;INITIALISE PORT1 AS INPUT
        LD     X, #INIP01
        JSR   RWI2C

```

TL/DD/10080-13

```

START1: LD      WPORT.B,#0FF      ;PUT ALL LED'S OFF
        LD      X,#WRP00
        JSR     RWI2C

        LD      X,#WRRAM0      ;WRITE TO RAM
        JSR     RWI2C          ;START TRANSMISSION
        JSR     WAIT

        LD      X,#RDRAM0      ;READ RAM0
        JSR     RWI2C
        JSR     WAIT

        ADD     WRBUFF.B,#8     ;WRITE/READ NEXT 8 BYTES RAM
        ADD     RDBUFF.B,#8
        IFEQ    WRBUFF.B,#0     ;IF WRAP
        DECSZ   WPORT.B,#0     ;DECREMENT LED VALUE
        NOP      ;ONLY DECREMENT

        LD      X,#RDPO1      ;READ INPUT
        JSR     RWI2C
        JSR     WAIT

        IFBIT   7,RPORT      ;IF BIT SET FREE-RUN-LED
        JP      START1
        LD      WPORT.B,RPORT.B ;ELSE COPY INPUT TO OUTPUT

        JP      START1

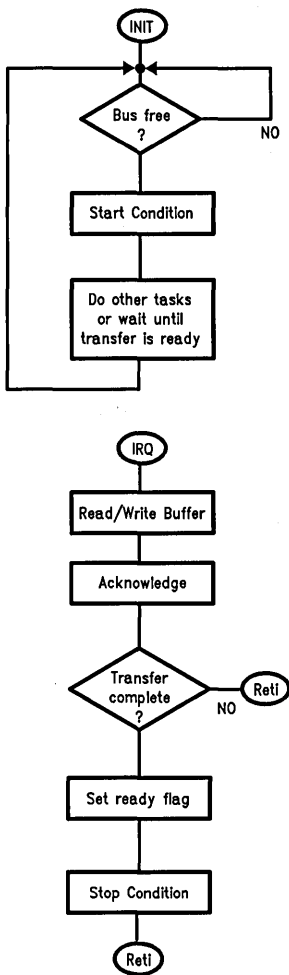
WAIT:   PUSH    X              ;SAVE X-REG
        LD      X,#010        ;INITIALIZE WAITLOOP
WAIT1:  DECSZ   X
        JP      WAIT1
        POP     X              ;RESTORE X-REG
        RET

INIP01: .DW     0242,WPORT,0    ;INITIALIZE PORT1 AS INPUT
RDPO1:  .DW     0243,RPORT,0    ;READ 1 BYTE FROM PORT1
WRP00:  .DW     0240,WPORT,0    ;WRITE 1 BYTE TO PORT0
WRRAM0: .DW     0AA0,WRBUFF,0   ;WRITE 8 BYTES TO RAM
RDRAM0: .DW     02A0,WRBUFF,0AA1,RDBUFF+1,0 ;READ 10 BYTES

```

.END RESET

TL/DD/10080-14



TL/DD/10080-15

FIGURE 9. Interrupt Driver Flowchart

```

;*****
;* INTER-IC-BUS (I2C-BUS) WITH HPC : APPL.NOTE
;* using uWire interface with interrupt
;*****
;* 12.10.87                               HU
;* 20.10.87                               HU
;* 04.11.87                               HU
;* 08.02.88                               HU

.INCLD HPC16083.MAP ; MEMORY MAP FOR HPC16083

;DEFINITIONS
CLK = 30 ;CLOCK LOW/HIGH TIME = 33us

.MACRO SAVE_AB
PUSH A ;SAVE A-REG
PUSH B ;SAVE B-REG
.ENDM

.MACRO SAVE_ABX
SAVE_AB ;SAVE REGS A,B
PUSH X ;SAVE X-REG
.ENDM

.MACRO RESTORE_AB
POP B ;RESTORE B-REG
POP A ;RESTORE A_REG
.ENDM

.MACRO RESTORE_ABX
POP X ;RESTORE X-REG
RESTORE_AB ;RESTORE REGS B,A
.ENDM

.SECT B1,BASE ;DEFINE BASEPAGE SECTION
WBUF1: .DSW 1 ;WORDBUFFER FOR I2C-TABLE
WBUF2: .DSW 1 ;WORDBUFFER FOR I2C-TABLE
INDEX: .DSW 1 ;POINTER TO THE NEXT TABLENTRY
STATUS: .DSB 1 ;STATUSBYTE
DUMMY: .DSB 1 ;DUMMY BYTE
WPORT: .DSB 1 ;8-BIT VALUE WRITE TO PORT0
RPORT: .DSB 1 ;8-BIT VALUE READ FROM PORT1
WRBUFF: .DSB 10 ;RAM WRITE BUFFER
RDBUFF: .DSB 10 ;RAM READ BUFFER

```

TL/DD/10080-16


```

.ENDSECT

.SECT I2C,ROM16

;*****
;* INIT : THIS SUBROUTINE INITIALIZES TIMER T4, TIMER T5
;*       AND THE UWIRE-INTERFACE TO OPERATE AS I2C-BUS
;*
;* INPUT : NONE
;* OUTPUT : NONE
;* USED REGS : A, B, X ( ALL REGS ARE SAVED )
;*****

INIT:  SAVE_ABX          ;SAVE REGS A,B,X
LD     X,#PWMODE       ;ADDR. PWMODE-REG -> X
LD     B,#PORTP       ;ADDR. PORTP-REG -> B
LD     A,#OCC         ;VALUE TO STOP TIMERS
ST     A,[X].B        ;STOP T4, T5, NO IRQ, ACK TIP-FLAG
NOP
ST     A,[X].B        ;MAKE SHURE TIP-FLAGS ARE CLEARED
LD     A,#011         ;DISABLE TOGGLE AND SET OUTPUT HIGH
ST     A,[B].B        ;ON PINS P0 AND P1
SBIT   3,[B].B        ;TOGGLE ON AT P0
SBIT   7,[B].B        ;TOGGLE ON AT P1
LD     T4.W,#CLK-1    ;LOAD T4 (33us)
LD     R4.W,#CLK-1    ;LOAD R4 (33us)
LD     T5.W,#17*CLK-1 ;9-BIT SHIFT TIME (STARTCONDITION)
LD     R5.W,#18*CLK-1 ;9-BIT SHIFT TIME (NORMAL MODE)
SBIT   5,PORTB.B      ;DATA LINE OUTPUT = HIGH
SBIT   5,DIRB.B       ;B5 = OUTPUT
RBIT   5,BFUN.B       ;NO ALTERNATE FUNCTION SELECTED
RBIT   6,DIRB.B       ;B6 = INPUT
SBIT   6,BFUN.B       ;SELECT SK-INPUT
LD     A,DIVBY.B      ;SET UWIRE-DEVIDE
AND    A,#0F0
OR     A,#02          ;SET CLKI /16
ST     A,DIVBY.B     ;STORE NEW VALUE
SBIT   1,IRCD.B      ;ACTIVATE UWIRE
INIT1: IFBIT 0,IRPD.B ;TEST IF READY
JP     INIT2         ;YES CONTINUE
JP     INIT1         ;NO WAIT
INIT2: RBIT 1,IRCD.B  ;SELECT SLAVE MODE
RBIT 6,BFUN.B
SBIT 4,[X].B        ;ENABLE T5-IRQ
OR     ENIR.B,#021   ;ENABLE GLOBAL TIMER IRQ
RESTORE_ABX         ;RESTORE REGS X,B,A
RET

RWI2C: PUSH  A        ;SAVE A-REGISTER
LD     STATUS.B,#0   ;RESET STATUSBYTE
LD     A,[X+].W      ;GET 2 BYTES OF TABLE
JSR    STRTCD        ;PERFORM STARTCONDITION
IFC

```

TL/DD/10080-17

```

JP      RWERR          ;IF ERROR -> EXIT
SBIT   5,BFUN.B       ;ENABLE SO-OUTPUT
ST     A,WBUF1.W      ;SAVE TABLE CONTENTS
IFBIT  0,A             ;TEST RECEIVE/TRANSMIT BIT
JP     RWRCV          ;BIT = 1 -> RECEIVE
RBIT   0,STATUS.B     ;STATUS = TRANSMIT
JP     RW01          ;CONTINUE AT RW01
RWRCV: SBIT   0,STATUS.B ;STATUS = RECEIVE
RW01:  SBIT   1,STATUS.B ;STATUS = FIRST BYTE
DECSZ  WBUF1+1.B      ;DEC BYTECOUNT
JP     RW02          ;MORE THAN 1 BYTE TO PROCESS
SBIT   2,STATUS.B     ;STATUS = LAST BYTE
RW02:  LD     A,[X+].W ;GET NEXT 2 BYTES OF TABLE
ST     A,WBUF2.W      ;SAVE TABLE CONTENTS
LD     A,X
ST     A,INDEX.W      ;SAVE INDEX
LD     A,[X].W        ;GET NEXT WORD OF TABLE
IFEQ   A,#0           ;ANY MORE TO TRANSFER
JP     RW03          ;NO, EXIT
RW03:  SBIT   3,STATUS.B ;STATUS = MULTISTART
SBIT   7,STATUS.B     ;STATUS = BUSY
RWERR: RC           ;CLR CARRY = NO ERROR
POP    A              ;RESTORE A-REGISTER
RET

STRICD:
IFBIT  5,PORT1.B      ;TEST DATALINE
JP     STRT01        ;IF HIGH -> CONTINUE
STRTER: SC           ;ELSE ERROR
RET
STRT01: IFBIT  6,PORTB.B ;TEST CLOCKLINE
JP     STRT02        ;IF HIGH -> CONTINUE
JP     STRTER        ;ELSE ERROR
STRT02: RBIT   5,PORTB.B ;DATALINE = LOW
AND    PWMODE.B,#0BB ;START TIMER 4 AND 5
STRT03: IFBIT  6,PORTB.B ;WAIT UNTIL CLOCK = LOW
JP     STRT03
ST     A,SIO.B        ;WRITE 1 BYTE TO SIO AND ENABLE SHIFT
RC
RET

TIMIRQ:
IFBIT  5,PWMODE.B     ;TIMER 5 IRQ ?
JP     T1IRQ        ;YES, CONTINUE
JP     IRQRET       ;NO TIMER IRQ
T1IRQ: SBIT   5,PORTB.B
RBIT   5,BFUN.B
SBIT   7,PWMODE.B     ;ACK IRQ
SAVE_ABX             ;SAVE REGS A,B,X
LD     B,#PORTB      ;PORTB-ADDR -> REG B
LD     X,#STATUS     ;STATUS-ADDR -> REG-X
PUSH  PSW.W
;
IFBIT  2,[X].B       ;LAST BYTE ?
JP     LAST          ;YES -> JUMP
IFBIT  1,[X].B       ;FIRST BYTE ?

```

TL/DD/10080-18

```

        JP      FIRST          ;YES -> JUMP
        IFBIT  0,[X].B        ;RECEIVEMODE ?
        JP      RECVE         ;YES -> JUMP
        JSR    GETACK         ;ACKNOWLEDGE
        IFC    ;ERROR ?
        JP      STOPCD        ;STOP TRANSMISSION
TRMIT:  LD     A,[WBUF2].B    ;GET NEXT BYTE
        ST     A,SIO.B        ;ENABLE SHIFT
        JP      TINC          ;-> INCREMENT POINTERS
RECVE:  LD     A,SIO.B        ;GET DATA
        ST     A,[WBUF2].B    ;PUT INTO BUFFER
        JSR    SETACK         ;ACKNOWLEDGE
        IFC    ;ERROR ?
        JP      STOPCD        ;STOP TRANSMISSION
        LD     SIO.B,#OFF     ;ENABLE SHIFT
TINC:   SBIT  5,BFUN.B        ;SELECT ALTERNATE MODE
TINC1:  INC   WBUF2.W         ;INC BUFFERPOINTER
TDEC:   DECSZ WBUF1+1.B       ;DEC BYTECOUNT
        JP      T5IRO2        ;MORE THAN 1 BYTE TO PROCESS
        SBIT  2,[X].B         ;STATUS = LAST BYTE
T5IRO2: JP      IRQEND        ;EXIT AND WAIT FOR NEXT IRQ

LAST:   IFBIT  0,[X].B        ;RECEIVE ?
        JP      LASTRD        ;YES -> JUMP
LAST1:  JSR    GETACK         ;ACKNOWLEDGE
        IFBIT  3,[X].B        ;RESTART ?
        JP      RESTRT        ;YES -> JUMP
        JP      STOPCD        ;STOP TRANSMISSION

LASTRD: LD     A,SIO.B        ;GET LAST CHARACTER
        ST     A,[WBUF2].B    ;PUT INTO BUFFER
        JP      LAST1

FIRST:  LD     A,[WBUF2].B    ;GET NEXT BYTE
        JSR    GETACK         ;ACKNOWLEDGE
        IFC    ;ERROR ?
        JP      STOPCD        ;STOP TRANSMISSION
        RBIT  1,[X].B        ;RESET FIRST BYTE FLAG
        IFBIT  0,[X].B        ;RECEIVE ?
        JP      IRRCV         ;YES -> JUMP
        ST     A,SIO.B        ;ENABLE SHIFT
        SBIT  5,BFUN.B        ;SELECT ALTERNATE FUNCTION
        JP      TINC1        ;-> INCREMENT POINTERS

IRRCV:  LD     SIO.B,#OFF     ;ACTIVATE SHIFT
        SBIT  5,BFUN.B        ;SELECT ALTERNATE FUNCTION
        JP      TDEC

IRQEND: POP    PSW.W          ;RESTORE REGS X,B,A
        RESTORE_ABX
IRQRET: RETI

RESTRT: LD     X,INDEX.W      ;GET NEXT POINTER TO ENTRYTABLE
        SBIT  5,[B].B        ;DATALINE = HIGH

```

TL/DD/10080-19

```

REST01: RBIT    5,BFUN.B      ;DISABLE SO-OUTPUT
         IFBIT   6,[B].B      ;WAIT UNTIL CLOCK = HIGH
         JP      REST02      ;CLOCK = HIGH -> REST02
         JP      REST01      ;WAIT
REST02: SBIT    2,PWMODE.B    ;STOP TIMER 4
         SBIT    6,PWMODE.B    ;STOP TIMER 5
         LD      T4.W,#2*CLK-1 ;LOAD TIMER 4
         LD      T5.W,#18*CLK-1;LOAD TIMER 5
         JSR     RWI2C         ;INITIALIZE READ/WRITE TO I2C-BUS
         JP      IRQEND

STOPCD: RBIT    5,[B].B      ;DATALINE = LOW
         RBIT    5,BFUN.B      ;DISABLE SO-OUTPUT
STOP01: IFBIT   6,[B].B      ;WAIT UNTIL CLOCK = HIGH
         JP      STOP02      ;CLOCK = HIGH -> STOP02
         JP      STOP01      ;WAIT
STOP02: SBIT    2,PWMODE.B    ;STOP TIMER 4
         SBIT    6,PWMODE.B    ;STOP TIMER 5
         LD      T4.W,#CLK-1   ;INITIALIZE T4 TO STARTCONDITION
         LD      T5.W,#17*CLK-1;INITIALIZE T5 TO STARTCONDITION
         RBIT    7,[X].B      ;STATUS = I2CBUS NOT BUSY
         SBIT    5,[B].B      ;PERFORM STOPCONDITION
         JP      IRQEND

SETACK: RBIT    5,[B].B      ;DATALINE = LOW
         RC
         JP      ACK01
GETACK: SBIT    5,[B].B      ;DATALINE = HIGH
         RC
ACK01:  IFBIT   6,[B].B      ;WAIT UNTIL CLOCK = HIGH
         JP      ACK02      ;CLOCK = HIGH
         JP      ACK01      ;CLOCK = LOW , WAIT
ACK02:  IFBIT   5,PORTI.B    ;TEST DATALINE
         SC              ;FLAG EROR IF HIGH
ACK03:  IFBIT   6,[B].B      ;WAIT UNTIL CLOCK = LOW
         JP      ACK03
         SBIT    5,[B].B      ;DATALINE = HIGH
         RET

TSTBUS: RC
         IFBIT   5,PORTI.B    ;TEST DATALINE
         JP      TST1
         SC
TST1:  IFBIT   6,[B].B      ;TEST CLOCKLINE
         JP      TST2
         SC
TST2:  RET

RESET:  LD      ENIR.B,#0     ;DISABLE ALL INTERRUPTS
         LD      PWMODE.W,#0CCCC ;STOP AND CLEAR ALL TIMERS
         LD      TMMODE.W,#0CCCC

START:  JSR     INIT
         LD      B,#WRBUFF

```

TL/DD/10080-20

```

START0: LD      K,#WRBUFF+8      ;CLEAR 9 BYTES
        CLR     A
        XS     A,[B+].W
        JP     START0
        LD     RDBUFF.B,#0      ;SET READADDRESS TO 0

        LD     WPORT.B,#0FF     ;INITIALISE PORT1 AS INPUT
        LD     X,#INIP01
        JSR   RWI2C
        JSR   WAIT

START1: LD     WPORT.B,#0FF     ;PUT ALL LED'S OFF
        LD     X,#WRP00
        JSR   RWI2C
        JSR   WAIT

        LD     X,#WRRAM0        ;WRITE TO RAM
        JSR   RWI2C             ;START TRANSMISSION
        JSR   WAIT

        LD     X,#RDRAM0        ;READ RAM0
        JSR   RWI2C
        JSR   WAIT

        ADD    WRBUFF.B,#8      ;WRITE/READ NEXT 8 BYTES RAM
        ADD    RDBUFF.B,#8
        IFEQ   WRBUFF.B,#0      ;IF WRAP
        DECSZ  WPORT.B          ;DECREMENT LED VALUE
        NOP    ;ONLY DECREMENT

        LD     X,#RDPO1         ;READ INPUT
        JSR   RWI2C
        JSR   WAIT

        IFBIT  7,RPORT          ;IF BIT SET FREE-RUN-LED
        JP     START1
        LD     WPORT.B,RPORT.B  ;ELSE COPY INPUT TO OUTPUT

        JP     START1

WAIT:   PUSH   X                ;SAVE X-REG
        LD     X,#010           ;INITIALIZE WAITLOOP
        IFC
        INC   DUMMY.B
WAIT1:  IFBIT  7,STATUS.B       ;WAIT UNTIL READY
        JP     WAIT1
WAIT2:  DECSZ  X
        JP     WAIT2
        POP   X                ;RESTORE X-REG
        RET

INIP01: .DW    0242,WPORT,0      ;INITIALIZE PORT1 AS INPUT
RDPO1:  .DW    0243,RPORT,0     ;READ 1 BYTE FROM PORT1
WRP00:  .DW    0240,WPORT,0     ;WRITE 1 BYTE TO PORT0
WRRAM0: .DW    0AA0,WRBUFF,0    ;WRITE 8 BYTES TO RAM

RDRAM0: .DW    02A0,WRBUFF,0AA1,RDBUFF+1,0 ;READ 10 BYTES

.IPT    5,TIMIRQ               ;SET TIMER IRQ ENTRY
.END RESET

```

TL/DD/10080-21

TL/DD/10080-22

```

;
;***** INCLUDE FILE HPC16083.MAP *****
;
;***** HPC-REGISTER DEFINITIONS *****

PSW      = 0C0      ;PROCESSOR STATUS REGISTER
;
SP        = 0C4      ;STACK POINTER
PC        = 0C6      ;PROGRAM COUNTER
;
A         = 0C8      ;ACCUMULATOR
;
K         = 0CA      ;K REGISTER
;
B         = 0CC      ;B REGISTER
;
X         = 0CE      ;X REGISTER
;
PORTA    = 0E0      ;PORTA DATA / OUTPUT BUFFER
DIRA     = 0F0      ;PORTA DIRECTION / INPUT BUFFER
PORTB    = 0E2      ;PORTB DATA REGISTER
DIRB     = 0F2      ;PORTB DIRECTION REGISTER
BFUN     = 0F4      ;PORTB ALTERNATE FUNCTION REG
PORTI    = 0D8      ;PORTI DATA REGISTER
PORTD    = 0104     ;PORTD DATA REGISTER
PORTP    = 0152     ;PORTP REGISTER
ENIR     = 0D0      ;INTERRUPT ENABLE REGISTER
IRCD     = 0D4      ;INTERRUPT AND CAPTURE CONDITION REG
IRPD     = 0D2      ;INTERRUPT PENDING REGISTER
HLTEN    = 0DC      ;HALT ENABLE CONTROL CIRCUIT
DIVBY    = 018E     ;DIVIDE BY REGISTER
PWMODE   = 0150     ;PULSE WIDTH MODE REGISTER
TMMODE   = 0190     ;TIMER MODE REGISTER
I2CR     = 0184     ;I2 CAPTURE REGISTER / R1
I3CR     = 0182     ;I3 CAPTURE REGISTER / T1
I4CR     = 0180     ;I4 CAPTURE REGISTER
EICR     = 015E     ;EI CAPTURE REGISTER
EICON    = 015C     ;EI CONFIGURATION REGISTER
T0CON    = 0192     ;T0 CAPTURE CONFIGURATION REG
T2        = 0188     ;TIMER2
R2        = 0186     ;TIMER2 MODULUS REGISTER
T3        = 018C     ;TIMER3
R3        = 018A     ;TIMER3 MODULUS REGISTER
T4        = 0140     ;TIMER4
R4        = 0142     ;TIMER4 MODULUS REGISTER
T5        = 0144     ;TIMER5
R5        = 0146     ;TIMER5 MODULUS REGISTER
T6        = 0148     ;TIMER6
R6        = 014A     ;TIMER6 MODULUS REGISTER
T7        = 014C     ;TIMER7
R7        = 014E     ;TIMER7 MODULUS REGISTER
WD        = 0194     ;WATCHDOG REGISTER
SIO      = 0D6      ;SERIAL INPUT OUTPUT SHIFT REG
ENU      = 0120     ;UART CONTROL AND STATUS REGISTER
ENUI     = 0122     ;UART INTERRUPT AND CLOCK SOURCE REG
RBUF     = 0124     ;UART RECEIVE BUFFER
TBUF     = 0126     ;UART TRANSMIT BUFFER
ENUR     = 0128     ;UART RECEIVE CONTROL AND STATUS REG
UPIC     = 0E6      ;UPI CONTROL REGISTER

```

TL/DD/10080-23

Extended Memory Support for HPC

National Semiconductor
Application Note 577
Raja Gopalan



INTRODUCTION

HPC™ family of microcontrollers have maximum addressing capability of 64 kbytes directly by the CPU. If an application requires more than 64k of address space, then the HPC address space can be expanded in terms of banks of memory, using an I/O port to select the memory banks. For example one can use PORTB pins 8, 9, 13 and 14 to select up to 16 banks of memory (which the MOLE development system also supports currently for debugging purposes). Please refer to the application note AN-497 "Expanding the HPC Address Space" by Joe Cocovich for hardware details.

The current version of HPC software package (Compiler, Assembler and Linker) however, does not directly support more than 64k of address space. This is mainly due to the Linker, which currently can handle only 64k of address space.

This report describes a method to handle more than 64k of address space from a software point of view. In order to do this, the user has to do multiple linking of modules in different banks and resolve the inter-bank symbol references.

The rest of the report describes the following:

1. Compiler generated selections (of code and data).
2. Programming conventions for bank switching.
3. Switch function to support bank switching.
4. Linking for bank switching.

SECTIONS GENERATED BY THE COMPILER

The compiler generates sections of relocatable assembly code which can be positioned in absolute address using the Linker in two ways:

1. Using the /SECT directive.
2. Using the /RANGE directive.

The following are the sections generated by the compiler for a source file named "MODULE":

1. MODULE_code,rom8	Code.
2. MODULE_code,rom16	
3. MODULE_ram8_bss,ram8	Data area for uninitialized static variables.
4. MODULE_ram16_bss,ram16	
5. MODULE_ram8_data,ram8	Data area for initialized static variables.
6. MODULE_ram16_data,ram16	
7. MODULE_ram8_strdata,ram 8	Data area for string literals.
8. MODULE_ram16_init,rom8	Initial value for static variables.
9. MODULE_ram8_init,rom8	
10. MODULE_ram8_strinit,rom8	Initial values for string literals.
11. MODULE_base16_bss,base	Base page area for uninitialized static variables.
12. MODULE_base8_bss,base	
13. MODULE_base16_data,base	Base page area for initialized static variables.
14. MODULE_base8_data,base	
15. MODULE_base16_init,rom16	Initial values for Base page initialized variables.
16. MODULE_base8_init,rom8	
17. MODULE_rom16_data,rom16	Area for constant storage type.
18. MODULE_rom8_data,rom8	
19. c_stack,ram16	Stack area in module containing main().
20. _init_info_	for each module which has any static variables defined.

PROGRAMMING CONVENTIONS TO BE USED FOR BANK SWITCHING

As far as the bank switching hardware is concerned, the HPC addressing space is divided into banks of memory. The Fixed Address space is referred to as shared bank and the switchable address space is called as switchable bank. Any mechanism for bank selection can be used, as long as the conventions mentioned below are strictly followed:

1. All static variables must be placed in the shared memory. Basepage must go in basepage (which is shared).
2. If string literals are not in ROM, they must be placed in the shared memory.
3. Initialization values for static variables or string literals in RAM must be in the shared memory. This includes basepage initializers and `__init_info__` sections.
4. If string literals for a bank are in ROM, and are never used as an argument to an inter-bank function call, the literals for that bank can be in the switchable bank.
5. If constants for a bank are never used by passing their address as an argument to an inter-bank function call, the constants for that bank can be in the switchable bank.
6. If the addresses of constants or string literals for a bank are used as arguments to an inter-bank function call, the constants or literals must be in the shared memory.
7. The stack must be in the shared memory.
8. Interrupt vectors must point to routines in the shared memory.
9. Only code and qualified ROM data can be placed in switchable banks.
10. A call to a function placed in the shared memory is always direct.
11. A function call from one switchable bank to another switchable bank must use a switching routine in the shared memory. Such a call cannot pass arguments which are addresses of functions, constants, or string literals in the calling bank. All pointers passed must be to objects in the shared memory.
12. A function which returns a structure cannot be used in an inter-bank function call if the returned structure is in memory in the calling bank. If the returned structure is an argument to another function, has a member of it accessed, or is assigned to a static or local variable, it is legal. If it is placed into switchable memory, by assigning to what is pointed at by a pointer, the operation will fail for an inter-bank function call.
13. The `START` macro in `CRTFIRST` must initialize the bankswitching port as necessary, and select the bank containing `main()` if it is not in the shared memory.

FUNCTION IN ASSEMBLER TO HANDLE SWITCHING OF BANKS

When bank switching must occur, the stack is set up by the compiler generated code for a normal function call. Instead of calling the destination function directly, however, the inter-bank link for the destination is called, as a result of the special manipulations with the linker `LNHPC`. This routine must change banks and then transfer to the destination, and must receive the return from the destination function so as to switch back to the original caller. This must be done transparently—no registers may be modified, and the stack must appear the same.

Included is the code to support the actual switching of banks during inter-bank function calls. This code allows a routine in either the shared memory or one of the switchable banks to make an inter-bank call to a routine in another bank.

The inter-bank link for each destination is created by a macro, invoked for each required linkage. The inter-bank link is simply a subroutine call to a common switching routine, with in-line arguments giving the bank and address of the destination. The common switching routine does the necessary manipulation of the stack to execute the destination and receive the return. The excess information is saved off in a separate software stack; upon return this information is used to restore the situation as if a normal function call had occurred.

Since the inter-bank transfer is completely transparent, it is not limited to handling C function calls. Any subroutine call which does not pass pointers to objects in switchable banks, which does not have in-line arguments, which does not use the Carry bit as either input or return, and which does not use a Return And Skip instruction, can be used with an inter-bank function call. However, the macro generates names using the C convention; an additional form is available for assembly subroutine names.

Also available is a version which allows the bank switching stack to be in 8-bit memory. It differs only in a few places from the 16-bit stack version.

The normal arrangement calls for the common switching routine and all the inter-bank links to be in shared memory. However, order of execution in the bank switching code is such that the inter-bank links for each destination that a bank needs can be in the switchable memory, and only the common routine needs to be in shared memory.

The software stack used by the bank switching is designed to grow downward, in contrast to the hardware stack, which grows upward. This allows the software stack to be placed in the same memory area as the hardware stack, but above it, and the two stacks will share their memory.

LINKAGE PROCEDURE FOR BANK SWITCHING

The actual linking of a multibank program is a series of individual linkages. The result will be a load module representing each bank's code, plus that bank's contribution to the shared memory area. It is essential that command files be used as inputs to `LNHPC` because each module must be linked several times, and changes would be ruinous:

First, each bank's set of modules must be linked independently. The Map files from each bank's linkage will give the necessary information on:

1. Undefined references, both functions and data.
2. A list of library routines invoked to support the code.
3. The size of the `__init_info__` section for the bank.
4. The size of the total code.
5. The entry for the functions defined in that bank.
6. The address of the variables defined in the bank (which is applicable for shared bank only).

This information should be checked and validated. The undefined data references must be only to data which will be in the shared memory. The undefined function references should be for the function calls defined in other banks. The library routines invoked may be reduced by library routines which will be in the shared memory to support code there, or can be placed in shared memory to use the shared ver-

sion for several banks. The size of each bank's `__init_info__` section will be used to make dummy sections for the initial shared memory linkage (see next step below). Finally, the total size of the code, allowing for library routines which will be in the shared memory, must fit in the bank.

Second, an initial linkage of the shared memory is done to determine the addresses of routines and data which will be in there. This requires certain routines to be assembled:

1. The inter-bank switching routine and all the links needed for inter-bank function calls (their bank and address values are left out initially).
2. External references for any additional library routines to be forced into the shared memory.
3. Dummy `__init_info__` sections which are each as large as the corresponding bank's real `__init_info__` section (or one dummy section as large as all the bank's sections combined).

The shared memory is then linked with all of these items included, and the Map file will give valid addresses of data, functions, and sections.

Third, the banks can be linked to produce actual modules. All entry points in the shared memory are now defined, and need to be provided to the linkages of each bank. Assembly files providing the definitions is the simplest way to go. One file can provide the addresses of all user functions, library routines, and data variables in the shared memory, from the Map of the shared memory. Individual files need to be made to provide the addresses of the inter-bank links, because the links for a bank cannot be given to that bank. Additionally, the next available addresses need to be figured for each memory area. This provides linkage and layout by creating the new names and values to resolve the undefined references in the linkage; the linker will do the work of substituting the link address for the undefined function address. Then each bank can be linked, with the addresses for memory areas given to the linker, and the additional files defining shared memory and the other banks inter-bank links being linked in. After each bank, the next available addresses must be updated. Note that the `__init_info__` sections must be contiguous and in the exact space created by the dummy routines.

Finally, the shared memory can be linked to produce the actual module. The banks and addresses must be provided for each inter-bank link and that module reassembled. The external references for additional library routines remains the same, and the dummy section for `__init_info__` are unchanged. The Map of this linkage must be checked against the Map of initial linkage and/or against all addresses fed to the bank switched modules.

EXAMPLE CODE DISTRIBUTION

The example is a skeleton for a realtime program which accumulates time data into tables, then processes those tables by regression fit into a table of coefficients. The system then monitors further events and uses the coefficient to predict behavior as it occurs. The following files are to be in a system with two banks, from 0x4000 to 0x7fff.

TABLES.H	Data structure
MAIN.C	Main program, for shared bank
TABLES.C	Table accumulation and processing
MONITOR.C	Monitor external events and predict
ERRORS.C	Error routines
TIMERS.C	Timer initialization and interrupt service
UART.C	UART processing and interrupt service

CRTFIRST.ASM	Modified to set up Port B for Bankswitching
CRTFIRST.INC	<Standard module, unchanged>
BANKSWIT.ASM	<Standard module, unchanged>
BANKLINK.INC	Modified for inter-bank linkages
BANKDEFS.INC	Macro definitions to simplify linkages

The distribution shown in Table I is intended as an initial starting point. The monitor and prediction code is very large, and fills the bank. The table processing code has room left so the error routines (which are seldom called) are fit in there. This bank has RAM in it, which is not known to the compiler but is managed by the program. Main is in shared memory because it is the major loop of the program. Timers and UART are in shared because they contain the Interrupt Service Routines.

Shared	Bank 0	Bank 1
Main	Tables	Monitor
Timers	Errors	Strings
UART	Strings	Constants
Crtfirst	Constants	
Statics	Table RAM	
Initialization		
Printf		
Stack		
Bank Stack		
Crtinit		

All statics will be in shared memory. Initialization data is in shared memory. The string literals are all in ROM, and will be in banks; since these are passed as arguments to `printf()`, `printf()` must either be in both banks or in shared memory in this case, to avoid duplication of memory usage and to save room in Bank 1. Constants are in banks, since inter-bank calls can be avoided when using constants and string literals. The stack and the bank stack are in the shared memory. The `crtfirst` routine is modified, and `crtinit` is with it in shared memory (although it may be possible to have `crtinit` in the bank selected by the `START` macro, this would require more manual linkage for the call in `crtfirst`).

LINKAGE PROCEDURE

Each bank load module is created by linking the banks separately. The linking is done in two steps. The first step is trial linkage and the parameters are specified in `BANK0_1.CMD`, `BANK1_1.CMD` and `SHARED_1.CMD` for linker. The information from this trial linkage is used in the second attempt where the load module is actually created. The command files used are `BANK0_2.CMD`, `BANK1_2.CMD` and `SHARED_2.CMD`.

Initial linker command files are:

```
SHARED_1.CMD
BANK0_1.CMD
BANK1_1.CMD
```

describing memory as

```
0000-01ff  shared: onchip RAM & I/O
0200-0fff  shared: offchip RAM
1000-3fff  shared: ROM
4000-7fff  banks
           bank 0: 4000-5fff  ROM
           6000-7fff  RAM, private
           bank 1: 4000-7fff  ROM
8000-ffff  shared: ROM
```

where the private RAM is not mentioned to the linker. The private RAM is defined to the compiler using constants; another alternative would have been to define an assembler module of the proper size allocating the space, and place it with the linker. This would require another piece of assembly code, but would limit the address information to the linker command files.

During the trial linkage Bank 0 links but contains `printf()`, which was desired to be in shared memory so it can be passed string literals; `putchar()` will also be there. This leaves only the variable `live`, which is just fine, will be placed in shared bank. The size of `__init_info__` is 0x4136 to 0x4147 or 18 bytes (this information is best taken from the Section Table of the map). The code is not present; it is assumed to fit. For Bank 1, `printf()` will again be defined in shared; `putchar()` will not be referenced. The undefined for `live`, `capture_table()`, and `error()` are correct. The size of `__init_info__` is 0x409A to 0x409F or 6 bytes. The code is assumed to fit.

The initial linkage of the shared memory requires the creation of the linkage files. The linkages have to be put into `BANKLINK.INC` for all inter-bank entry points, including from shared to a bank. The sizes for the `__init_info__` sections and the library access forcing requests are put in a file, using `BANKDEFS.INC` to make things easier. These are linked together, with the C stack and the switch stack in the off-chip RAM, with the switch stack on top so that they can share the same memory. There are few inter-bank calls, so the `SWITCH_STACK_DEPTH` used is 10. Linking this finishes the initial sequence, and the values are now available for the real second attempt of linking whereby the actual modules will be created.

Now the definitions to complete each bank are created. The module `BANKDEFS.INC` makes this easier. Each bank defines the linkages to entry points within that bank. The shared defines publics within the shared memory. (These values are best taken from the Symbol Table portion of the map.) Then the linker command files need to be modified (in the example new file names are used, but the user will probably not use new files, rather simply modify the existing files). The definition files needed for each bank will be added; these are the file for shared memory and for every other bank but this one. The No Output option is changed to giving a name for the object file, if desired, and the Ignore Errors is added because there is still no reset vector for a bank.

Finally, the memory addresses have to be determined from the shared load map and put into the command file (these values are best taken from the Memory Order Map, Memory Type Map, or the Section Table). The positioning of `__init_info__` is critical, the others can have gaps. A trial linkage shows where the linker places modules, and final adjustments are required to ensure such placements meet the requirements. Bank 0 requires only that the initialization data be moved to shared memory. The updated addresses from Bank 0 are used in Bank 1. Bank 1 is placed acceptably by the linker.

The final linkage of the shared memory can now be done. Address and bank information is added to the linkage list. The remaining parts don't change. This linkage must be checked against the first linkage of shared to be certain no addresses have changed. Finally, the addresses used in each bank or shared should be checked against other banks to check for overlaps, and the types of sections in each memory should be checked to make sure all conventions have been met.

If everything is correct, you have load modules for the system.

The code listed in this Application Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this Application Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

Contents of Linker command file BANK0_1.CMD:

```

/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank0_1.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x4000:0x5FFF,0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
tables,
errors
/NoOutput

```

TL/DD/10131-1

Contents of the Linker map file BANK0_1.MAP:

NSC LNHP, Version E2 (Nov 02 15:46 1987)

09-May-88 08:37

Reset Vector: 0000

-- Range Definitions --

```

BASE    0002:00BF
ROM16   4000:5FFF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

```

-- Memory Order Map --

```

0200 0211  RAM16
4000 4508  ROM16
450A 4687  ROM16
4688 47BF  ROM8

```

-- Memory Type Map --

```

BASE
  [size = 0000]

```

```

RAM16
  0200 0211
  [size = 0012]

```

```

RAM8
  [size = 0000]

```

```

ROM16
  4000 4508
  450A 4687
  [size = 0687]

```

```

ROM8
  4688 47BF
  [size = 0138]

```

```

VECTOR
  [size = 0000]

```

TL/DD/10131-2

-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8

0200 0211
[size = 0012]

TOTAL ROM = ROM16 + ROM8 + VECTOR

4000 4508
450A 4687
4688 47BF
[size = 07BF]

-- Section Table --

start	end	attributes	Section Module
0200	020D	RAM16 WORD	TABLES_RAM16_BSS
0200	020D		tables
4000	4135	ROM16 WORD	TABLES_CODE
4000	4135		tables
4136	4147	ROM16 WORD	_INIT_INFO_
4136	413B		tables
413C	4147		errors
020E	020F	RAM16 WORD	ERRORS_RAM16_DATA
020E	020F		errors
0210	0211	RAM16 WORD	ERRORS_RAM16_BSS
0210	0211		errors
4148	41C3	ROM16 WORD	ERRORS_CODE
4148	41C3		errors
4688	4689	ROM8 WORD	ERRORS_RAM16_INIT
4688	4689		errors
468A	4703	ROM8 BYTE	ERRORS_ROM8_STRDATA
468A	4703		errors
41C4	4508	ROM16 WORD	LIBI_CODE
41C4	4508		libi
450A	4687	ROM16 WORD	LIBP_CODE
450A	4687		libp
4704	47BF	ROM8 BYTE	LIBRARY
4704	47BF		LIBIDVL

Error: Undefined External: _live
Address: 0096
Module: tables

Error: Undefined External: _putchar
Address: 0044
Module: errors

Error: Undefined External: _putchar
Address: 004A

TL/DD/10131-3

```

Module: libi
Error: Undefined External: _putchar
Address: 0086
Module: libi
Error: Undefined External: _putchar
Address: 0190
Module: libi
Error: Undefined External: _putchar
Address: 028A
Module: libi
Error: Undefined External: _putchar
Address: 02D9
Module: libi
Error: Undefined External: _putchar
Address: 0337
Module: libi
Error: Undefined External: _putchar
Address: 0027
Module: libp
Error: Undefined External: _putchar
Address: 0057
Module: libp
Error: Undefined External: _putchar
Address: 0146
Module: libp
Error: Undefined External: _putchar
Address: 0175
Module: libp
Error: No End Address has been specified

```

```

signed_divide_32 . . . . 4704 Null ROM8
-LIBIDVL
signed_remainder_32 . . 4708 Null ROM8
-LIBIDVL
unsigned_divide_32 . . . 4739 Null ROM8
-LIBIDVL libp
unsigned_remainder_32 . 473D Null ROM8
-LIBIDVL libp
_build_tables . . . . . 404B Null ROM16
-tables
_capture_table . . . . . 404E Null ROM16
-tables
_compute_coefficients . 40AB Null ROM16
-tables
_d_printf . . . . . 453C Null ROM16
-libp libi
_error . . . . . 4148 Null ROM16
-errors
_fatal_error . . . . . 416B Null ROM16
-errors
_initialize_table_memory 4000 Null ROM16
-tables
_live . . . . . **** Null

```

TL/DD/10131-4

```

    tables
_printf . . . . . 41C4 Null ROM16
  -libi      errors
_putchar . . . . . **** Null
  errors     libi      libp
_quit . . . . . 41B0 Null ROM16
  -errors
_s_printf . . . . . 450A Null ROM16
  -libp     libi
_u_printf . . . . . 459A Null ROM16
  -libp     libi

```

TL/DD/10131-5

Information obtained from BANK0_1.MAP are:

1) The `_INIT_INFO` section size for Bank0 linkage is 18 bytes, ie from 0x4136 to 0x4147.

2) The entry address for functions obtained here as follows:

Function	Address
<code>initialize_table_memory</code>	0x4000
<code>build_tables</code>	0x404b
<code>capture_table</code>	0x404e
<code>compute_coefficients</code>	0x40ab
<code>error</code>	0x4136
<code>fatal_error</code>	0x4159

These addresses will be used by the `SWITCH_TO_FUNCTION` assembly macro calls in the file `BANKLINK.INC`.

3) The undefined external reference for the variable `live` is expected, which will be defined in the `SHARED` bank. The undefined function `putchar` will also be defined in the shared bank.

TL/DD/10131-6

Contents of Linker Command file BANK1_1.CMD:

```
/Echo  
/libfile=\hpc\library  
/Format=lm  
/Map=bank1_1.map  
/Table  
/Cr  
/Range=BASE=(0x0002:0x00BF)  
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)  
/Range=RAM8=RAM16  
/Range=ROM16=(0x4000:0x7FFF,0x8000:0xFFCF,0x1000:0x3FFF)  
/Range=ROM8=ROM16  
monitor  
/NoOutput
```

TL/DD/10131-7

Contents of the Linker map file BANK1_1.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)

09-May-88 08:37

Reset Vector: 0000

-- Range Definitions --

```
BASE    0002:00BF
ROM16   4000:7FFF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16
```

-- Memory Order Map --

```
0200 0201  RAM16
4000 43E4  ROM16
43E6 4563  ROM16
4564 465F  ROM8
```

-- Memory Type Map --

```
BASE
  [size = 0000]
```

```
RAM16
  0200 0201
  [size = 0002]
```

```
RAM8
  [size = 0000]
```

```
ROM16
  4000 43E4
  43E6 4563
  [size = 0563]
```

```
ROM8
  4564 465F
  [size = 00FC]
```

```
VECTOR
  [size = 0000]
```

TL/DD/10131-8

-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8

0200 0201
[size = 0002]

TOTAL ROM = ROM16 + ROM8 + VECTOR

4000 43E4
43E6 4563
4564 465F
[size = 065F]

-- Section Table --

start	end	attributes	Section Module
0200	0201	RAM16 WORD	MONITOR_RAM16_BSS
0200	0201		monitor
4000	4099	ROM16 WORD	MONITOR_CODE
4000	4099		monitor
4564	45A3	ROM8 BYTE	MONITOR_ROM8_STRDATA
4564	45A3		monitor
409A	409F	ROM16 WORD	_INIT_INFO_
409A	409F		monitor
40A0	43E4	ROM16 WORD	LIBI_CODE
40A0	43E4		libi
43E6	4563	ROM16 WORD	LIBP_CODE
43E6	4563		libp
45A4	465F	ROM8 BYTE	LIBRARY
45A4	465F		LIBIDVL

```

Error: Undefined External: _live
      Address: 0002
      Module: monitor
Error: Undefined External: _live
      Address: 0012
      Module: monitor
Error: Undefined External: _live
      Address: 0026
      Module: monitor
Error: Undefined External: _capture_table
      Address: 0030
      Module: monitor
Error: Undefined External: _error
      Address: 0091
      Module: monitor
Error: Undefined External: _putchar
      Address: 004A
      Module: libi

```

TL/DD/10131-9

```

Error: Undefined External: _putchar
      Address: 0086
      Module: libi
Error: Undefined External: _putchar
      Address: 0190
      Module: libi
Error: Undefined External: _putchar
      Address: 028A
      Module: libi
Error: Undefined External: _putchar
      Address: 02D9
      Module: libi
Error: Undefined External: _putchar
      Address: 0337
      Module: libi
Error: Undefined External: _putchar
      Address: 0027
      Module: libp
Error: Undefined External: _putchar
      Address: 0057
      Module: libp
Error: Undefined External: _putchar
      Address: 0146
      Module: libp
Error: Undefined External: _putchar
      Address: 0175
      Module: libp
Error: No End Address has been specified

```

```

signed_divide_32 . . 45A4 Null ROM8
-LIBIDVL
signed_remainder_32 45A8 Null ROM8
-LIBIDVL
unsigned_divide_32 . 45D9 Null ROM8
-LIBIDVL libp
unsigned_remainder_32 45DD Null ROM8
-LIBIDVL libp
_capture_table . . . **** Null
monitor
_compute_prediction 4032 Null ROM16
-monitor
_d_printf . . . . . 4418 Null ROM16
-libp libi
_error . . . . . **** Null
monitor
_live . . . . . **** Null
monitor
_monitor . . . . . 4000 Null ROM16
-monitor
_printf . . . . . 40A0 Null ROM16
-libi monitor
_putchar . . . . . **** Null
libi libp

```

TL/DD/10131-10

```
_s_printf . . . . . 43E6 Null ROM16
  -libp   libi
_u_printf . . . . . 4476 Null ROM16
  -libp   libi
_validate_calculation 4053 Null ROM16
  -monitor
```

TL/DD/10131-11

The informations derieved from this file are:

1) The `_INIT_INFO` section size for Bank0 linkage is 6 bytes.
ie, from `0x409a` to `0x409f`.

2) The entry address for functions obtained here as follows:

Function	Address
----------	---------

monitor	0x4000
---------	--------

These addresses will be used by the `SWITCH_TO_FUNCTION` assembly macro calls in the file `BANKLINK.INC`.

3) The undefined external reference for the variable `live` is expected, which will be defined in the `SHARED` bank. The undefined function `putchar` will also be defined in the `shared` bank. The undefined external references for functions defined in `Bank0` and `Shared` will be taken care by proper link addresses during second pass of linkage.

TL/DD/10131-12

Contents of the Linker command file SHARED_1.CMD:

```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=shared_1.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
/Sect=c_stack=0x0200:0x0FFF
/Sect=switch_stack=c_stack
main,
timers,
uart,
crtfirst,
bankswit, shared_1
/NoOutput
```

Note that we include the files BANKSWIT.ASM and SHARED_1.ASM. BANKSWIT.ASM includes BANKLINK.INC file in which we have made the switch_to_function macro calls for the inter bank function refernces. SHARED_1.ASM contains the init_dummy macro call to create continuous space for _INIT_INFO_ section in shared memory. Also it contains the force_library macro call to force PUTCHAR and PRINTF in shared address space.

TL/DD/10131-13

Contents of the Linker output file SHARED_1.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)

09-May-88 08:37

Reset Vector: FFAF

-- Range Definitions --

```
BASE    0002:00BF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16
```

-- Memory Order Map --

```
0002 0003  BASE
0200 0ABF  RAM16
8000 80A8  ROM16
80AA 8118  ROM16
811A 845E  ROM16
8460 85DD  ROM16
85DE 873C  ROM8
FFAF FFBF  ROM8
FFF4 FFF5  VECTOR
FFFA FFFB  VECTOR
FFFE FFFF  VECTOR
```

-- Memory Type Map --

```
BASE
0002 0003
[size = 0002]
```

```
RAM16
0200 0ABF
[size = 08C0]
```

```
RAM8
[size = 0000]
```

```
ROM16
8000 80A8
80AA 8118
811A 845E
8460 85DD
[size = 05DB]
```

TL/DD/10131-14

```

ROM8
85DE 873C
FFAF FFBF
[size = 0170]

```

```

VECTOR
FFF4 FFF5
FFFA FFFB
FFFE FFFF
[size = 0006]

```

-- Total Memory Map --

```

TOTAL RAM = BASE + RAM16 + RAM8
0002 0003
0200 0ABF
[size = 08C2]

```

```

TOTAL ROM = ROM16 + ROM8 + VECTOR
8000 80A8
80AA 8118
811A 845E
8460 85DD
85DE 873C
FFAF FFBF
FFF4 FFF5
FFFA FFFB
FFFE FFFF
[size = 0751]

```

-- Section Table --

start	end	attributes	Section Module
0200	09FF	RAM16 WORD	C_STACK
0200	09FF		main
0A00	0A27	RAM16 WORD	SWITCH_STACK
0A00	0A27		Bank_Switch
0A28	0A2D	RAM16 WORD	MAIN_RAM16_DATA
0A28	0A2D		main
0A2E	0A41	RAM16 WORD	MAIN_RAM16_BSS
0A2E	0A41		main
8000	8031	ROM16 WORD	MAIN_CODE
8000	8031		main
85DE	85E3	ROM8 WORD	MAIN_RAM16_INIT
85DE	85E3		main
8032	8061	ROM16 WORD	_INIT_INFO_

TL/DD/10131-15

8032	803D			main
803E	8043			timers
8044	8049			Bank_Switch
804A	8061			SHARED_1
0A42	0ABF	RAM16	WORD	TIMERS_RAM16_BSS
0A42	0ABF			timers
8062	80A8	ROM16	WORD	TIMERS_CODE
8062	80A8			timers
80AA	8118	ROM16	WORD	UART_CODE
80AA	8118			uart
FFAF	FFBF	ROM8	ABS	CRTFIRST
FFAF	FFBF			crtfirst
0002	0003	BASE	WORD	SWITCH_POINTER
0002	0003			Bank_Switch
85E4	85E5	ROM8	BYTE	SWITCH_INIT
85E4	85E5			Bank_Switch
85E6	8659	ROM8	BYTE	SWITCH_CODE
85E6	8659			Bank_Switch
865A	873C	ROM8	BYTE	LIBRARY
865A	8680			crtinit
8681	873C			LIBIDVL
811A	845E	ROM16	WORD	LIBI_CODE
811A	845E			libi
8460	85DD	ROM16	WORD	LIBP_CODE
8460	85DD			libp

initialize_memories	..	865A	Null	ROM8
-crtinit				crtfirst
PROGRAM_exit	FFBF	Null	ROM8
-crtfirst				
PROGRAM_start	FFAF	Null	ROM8
-crtfirst				main
STACK_end	0A00	Null	RAM16
-main				
STACK_start	0200	Null	RAM16
-main				crtfirst
signed_divide_32	8681	Null	ROM8
-LIBIDVL				
signed_remainder_32	..	8685	Null	ROM8
-LIBIDVL				
unsigned_divide_32	8686	Null	ROM8
-LIBIDVL				libp
unsigned_remainder_32	..	868A	Null	ROM8
-LIBIDVL				libp
_build_tables	85EB	Null	ROM8
-Bank_Switch				main
_button_service	8086	Null	ROM16
-timers				
_calibrating	0A2A	Byte	RAM16
-main				
_capture_table	85F0	Null	ROM8
-Bank_Switch				

TL/DD/10131-16

```

_coefficients . . . . . 0A2E Byte RAM16
  -main
_compute_coefficients . 85F5 Null ROM8
  -Bank_Switch      main
_d_printf . . . . . 8492 Null ROM16
  -libp      libi
_error . . . . . 85FF Null ROM8
  -Bank_Switch
_fatal_error . . . . . 8604 Null ROM8
  -Bank_Switch
_initialize_inputs . . . 8062 Null ROM16
  -timers      main
_initialize_outputs . . 80AA Null ROM16
  -uart      main
_initialize_table_memory 85E6 Null ROM8
  -Bank_Switch      main
_live . . . . . 0A42 Byte RAM16
  -timers
_main . . . . . 8000 Null ROM16
  -main      crtfirst
_monitor . . . . . 85FA Null ROM8
  -Bank_Switch      main
_operational . . . . . 0A28 Byte RAM16
  -main
_predicting . . . . . 0A2C Byte RAM16
  -main
_printf . . . . . 811A Null ROM16
  -libi      SHARED_1
_put_uart . . . . . 810E Null ROM16
  -uart
_putchar . . . . . 80AB Null ROM16
  -uart      libi      libp
_s_printf . . . . . 8460 Null ROM16
  -libp      libi
_timer_service . . . . . 8063 Null ROM16
  -timers
_u_printf . . . . . 84F0 Null ROM16
  -libp      libi

```

Notice that there is entry for each function that is actually placed in switchable bank being called from other banks. These entries are used as link addresses for the respective functions when linking the banks individually. Refer the files shared.asm, bank0.asm and bank1.asm.

TL/DD/10131-17

Contents of the linker command file BANK0_2.CMD:

```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank0_2.map
/Table
/Cr
/Range=BASE=(0x0004:0x00BF)
/Range=RAM16=(0x0B00:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAMB=RAM16
/Range=ROM16=(0x4000:0x5FFF,0x8740:0xFFAE,0x1000:0x3FFF)
/Range=ROMB=ROM16
tables,
errors,
shared, bank1
/Sect=_init_info_=0x804A:0x8061
/Sect=errors_ram16_init=0x8740
/Output=bank0
/Ignore
```

Notice the ROM address is space defined as 0x4000:0x5fff for the BANK0 space. In shared memory address range 0x8740:0xffae is available, which is obtained from shared_1.map. Also `_init_info_` goes into the range 0x804a:0x8061 which was reserved by the `init_dummy` macro and the address is obtained from shared_1.map. Also the section `errors_ram16_init` goes to address 0x8740 onwards. The link addresses for the functions and variables are specified in shared.asm and bank1.asm.

TL/DD/10131-18

Contents of the Linker output file BANK0_2.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)

09-May-88 08:38

Reset Vector: 0000

-- Range Definitions --

```
BASE    0004:00BF
ROM16   4000:5FFF
ROM16   8740:FFAE
ROM16   1000:3FFF
RAM16   0B00:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16
```

-- Memory Order Map --

```
0B00 0B11  RAM16
4000 41B1  ROM16
41B2 422B  ROM8
804A 805B  ROM16
8740 8741  ROM8
```

-- Memory Type Map --

```
BASE
  [size = 0000]
```

```
RAM16
  0B00 0B11
  [size = 0012]
```

```
RAM8
  [size = 0000]
```

```
ROM16
  4000 41B1
  804A 805B
  [size = 01C4]
```

```
ROM8
  41B2 422B
  8740 8741
  [size = 007C]
```

```
VECTOR
  [size = 0000]
```

TL/DD/10131-19

-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
 0B00 0B11
 [size = 0012]

TOTAL ROM = ROM16 + ROM8 + VECTOR
 4000 41B1
 41B2 422B
 804A 805B
 8740 8741
 [size = 0240]

-- Section Table --

start	end	attributes	Section Module
804A	805B	ROM16 WORD	_INIT_INFO_
804A	804F		tables
8050	805B		errors
8740	8741	ROM8 WORD	ERRORS_RAM16_INIT
8740	8741		errors
0B00	0B0D	RAM16 WORD	TABLES_RAM16_BSS
0B00	0B0D		tables
4000	4135	ROM16 WORD	TABLES_CODE
4000	4135		tables
0B0E	0B0F	RAM16 WORD	ERRORS_RAM16_DATA
0B0E	0B0F		errors
0B10	0B11	RAM16 WORD	ERRORS_RAM16_BSS
0B10	0B11		errors
4136	41B1	ROM16 WORD	ERRORS_CODE
4136	41B1		errors
41B2	422B	ROM8 BYTE	ERRORS_ROM8_STRDATA
41B2	422B		errors

Error: No End Address has been specified

```

_build tables . . . . . 404B Null ROM16
-tables
_capture_table . . . . . 404E Null ROM16
-tables
_coefficients . . . . . 0A2E Null
-SHARED
_compute_coefficients . 40AB Null ROM16
-tables

```

TL/DD/10131-20

```

_error . . . . . 4136 Null ROM16
-errors
_fatal_error . . . . . 4159 Null ROM16
-errors
_initialize_table_memory 4000 Null ROM16
-tables
_live . . . . . 0A42 Null
-SHARED tables
_monitor . . . . . 85FC Null
-BANK1
_printf . . . . . 811A Null
-SHARED errors
_putchar . . . . . 80AB Null
-SHARED errors
_quit . . . . . 419E Null ROM16
-errors

```

Notice that there is no undefined external references errors.

TL/DD/10131-21

Since the function Main is not defined in this bank there is no reset vector address defined and hence the Linker gives the 'no end address specified' error message, which can be ignored.

Contents of the Linker command file BANK1_1.CMD:

```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank1_2.map
/Table
/Cr
/Range=BASE=(0x0004:0x00BF)
/Range=RAM16=(0x0C00:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x4000:0x7FFF,0x8742:0xFFAE,0x1000:0x3FFF)
/Range=ROM8=ROM16
monitor,
shared, bank0
/Sect=_init_info_=0x805C:0x8061
/Output=bank1
/Ignore
```

Notice the `_init_info_` section is placed in address space 0x805c to 0x8061. This is basically the rest of the space after `Bank0_init_info_usage`. Also the Link addresses for BANK0 and SHARED are appropriately mentioned in the assembly files `bank0.asm` and `shared.asm` and they are also linked. The shared address (ROM16 and RAM16) space is properly updated with the information from `bank0_1.map`.

TL/DD/10131-22

Contents of the Linker output file BANK1_2.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)

09-May-88 08:38

Reset Vector: 0000

-- Range Definitions --

```

BASE    0004:00BF
ROM16   4000:7FFF
ROM16   8742:FFAE
ROM16   1000:3FFF
RAM16   0C00:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

```

-- Memory Order Map --

```

0C00 0C01  RAM16
4000 4099  ROM16
409A 40D9  ROM8
805C 8061  ROM16

```

-- Memory Type Map --

```

BASE
[size = 0000]

```

```

RAM16
0C00 0C01
[size = 0002]

```

```

RAM8
[size = 0000]

```

```

ROM16
4000 4099
805C 8061
[size = 00A0]

```

```

ROM8
409A 40D9
[size = 0040]

```

```

VECTOR
[size = 0000]

```

TL/DD/10131-23

-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8

0C00 0C01
[size = 0002]

TOTAL ROM = ROM16 + ROM8 + VECTOR

4000 4099
409A 40D9
805C 8061
[size = 00E0]

-- Section Table --

start	end	attributes	Section Module
805C	8061	ROM16 WORD	_INIT_INFO_ monitor
0C00	0C01	RAM16 WORD	MONITOR_RAM16_BSS monitor
4000	4099	ROM16 WORD	MONITOR_CODE monitor
409A	40D9	ROM8 BYTE	MONITOR_ROM8_STRDATA monitor

Error: No End Address has been specified

_build_tables	85ED	Null	
-BANK0				
_capture_table	85F2	Null	
-BANK0	monitor			
_coefficients	0A2E	Null	
-SHARED				
_compute_coefficients	85F7	Null	
-BANK0				
_compute_prediction	4032	Null	ROM16
-monitor				
_error	8601	Null	
-BANK0	monitor			
_fatal_error	8606	Null	
-BANK0				
_initialize_table_memory	85E8	Null	
-BANK0				
_live	0A42	Null	
-SHARED	monitor			
_monitor	4000	Null	ROM16
-monitor				

TL/DD/10131-24

```

_printf . . . . . 811A Null
  -SHARED   monitor
_putchar . . . . . 80AB Null
  -SHARED
_validate_calculation . 4053 Null ROM16
  -monitor

```

Notice that there is no undefined external reference error messages.

TL/DD/10131-25

Since the function Main is not defined in this bank there is no reset vector address defined and hence the Linker gives the 'no end address specified' error message, which can be ignored.

Contents of the linker command file SHARED_2.CMD which is same as SHARED_1.CMD:

```

/Echo
/libfile=\hpc\library
/Format=lm
/Map=shared_2.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
/Sect=c_stack=0x0200:0x0FFF
/Sect=switch_stack=c_stack
main,
timers,
uart,
crtfirst,
bankswit, shared_1
/Output=shared

```

TL/DD/10131-26

Contents of the Linker output file SHARED_2.MAP which should be identical to SHARED_1.MAP:

NSC LNHC, Version E2 (Nov 02 15:46 1987)

09-May-88 08:38

Reset Vector: FFAF

-- Range Definitions --

```
BASE    0002:00BF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16
```

-- Memory Order Map --

```
0002 0003  BASE
0200 0ABF  RAM16
8000 80A8  ROM16
80AA 8118  ROM16
811A 845E  ROM16
8460 85DD  ROM16
85DE 873C  ROM8
FFAF FFBF  ROM8
FFF4 FFF5  VECTOR
FFFA FFFB  VECTOR
FFFE FFFF  VECTOR
```

-- Memory Type Map --

```
BASE
 0002 0003
 [size = 0002]
```

```
RAM16
 0200 0ABF
 [size = 08C0]
```

```
RAM8
 [size = 0000]
```

```
ROM16
 8000 80A8
 80AA 8118
 811A 845E
 8460 85DD
 [size = 05DB]
```

TL/DD/10131-27

ROM8
 85DE 873C
 FFAF FFBF
 [size = 0170]

VECTOR
 FFF4 FFF5
 FFFA FFFB
 FFFE FFFF
 [size = 0006]

-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
 0002 0003
 0200 0ABF
 [size = 08C2]

TOTAL ROM = ROM16 + ROM8 + VECTOR
 8000 80A8
 80AA 8118
 811A 845E
 8460 85DD
 85DE 873C
 FFAF FFBF
 FFF4 FFF5
 FFFA FFFB
 FFFE FFFF
 [size = 0751]

-- Section Table --

start	end	attributes	Section Module
0200	09FF	RAM16 WORD	C_STACK
0200	09FF		_main
0A00	0A27	RAM16 WORD	SWITCH_STACK
0A00	0A27		Bank_Switch
0A28	0A2D	RAM16 WORD	MAIN_RAM16_DATA
0A28	0A2D		_main
0A2E	0A41	RAM16 WORD	MAIN_RAM16_BSS
0A2E	0A41		_main
8000	8031	ROM16 WORD	MAIN_CODE
8000	8031		_main
85DE	85E3	ROM8 WORD	MAIN_RAM16_INIT
85DE	85E3		_main

TL/DD/10131-28

8032	8061	ROM16	WORD	_INIT_INFO_
8032	803D			main
803E	8043			timers
8044	8049			Bank_Switch
804A	8061			SHARED_1
0A42	0ABF	RAM16	WORD	TIMERS_RAM16_BSS
0A42	0ABF			timers
8062	80A8	ROM16	WORD	TIMERS_CODE
8062	80A8			timers
80AA	8118	ROM16	WORD	UART_CODE
80AA	8118			uart
FFAF	FFBF	ROM8	ABS	CRTFIRST
FFAF	FFBF			crtfirst
0002	0003	BASE	WORD	SWITCH_POINTER
0002	0003			Bank_Switch
85E4	85E5	ROM8	BYTE	SWITCH_INIT
85E4	85E5			Bank_Switch
85E6	8659	ROM8	BYTE	SWITCH_CODE
85E6	8659			Bank_Switch
865A	873C	ROM8	BYTE	LIBRARY
865A	8680			crtinit
8681	873C			LIBIDVL
811A	845E	ROM16	WORD	LIBI_CODE
811A	845E			libi
8460	85DD	ROM16	WORD	LIBP_CODE
8460	85DD			libp

initialize_memories	..	865A	Null	ROM8
-crtinit	crtfirst			
PROGRAM_exit	FFBF	Null	ROM8
-crtfirst				
PROGRAM_start	FFAF	Null	ROM8
-crtfirst	main			
STACK_end	0A00	Null	RAM16
-main				
STACK_start	0200	Null	RAM16
-main	crtfirst			
signed_divide_32	8681	Null	ROM8
-LIBIDVL				
signed_remainder_32	..	8685	Null	ROM8
-LIBIDVL				
unsigned_divide_32	..	86B6	Null	ROM8
-LIBIDVL	libp			
unsigned_remainder_32	..	86BA	Null	ROM8
-LIBIDVL	libp			
_build_tables	85EB	Null	ROM8
-Bank_Switch	main			
_button_service	8086	Null	ROM16
-timers				
_calibrating	0A2A	Byte	RAM16
-main				
_capture_table	85F0	Null	ROM8

TL/DD/10131-29

```

    -Bank_Switch
  _coefficients . . . . . 0A2E Byte RAM16
    -main
  _compute_coefficients . 85F5 Null ROM8
    -Bank_Switch          main
  _d_printf . . . . . 8492 Null ROM16
    -libp          libi
  _error . . . . . 85FF Null ROM8
    -Bank_Switch
  _fatal_error . . . . . 8604 Null ROM8
    -Bank_Switch
  _initialize_inputs . . . 8062 Null ROM16
    -timers          main
  _initialize_outputs . . 80AA Null ROM16
    -uart          main
  _initialize_table_memory 85E6 Null ROM8
    -Bank_Switch          main
  _live . . . . . 0A42 Byte RAM16
    -timers
  _main . . . . . 8000 Null ROM16
    -main          crtfirst
  _monitor . . . . . 85FA Null ROM8
    -Bank_Switch          main
  _operational . . . . . 0A28 Byte RAM16
    -main
  _predicting . . . . . 0A2C Byte RAM16
    -main
  _printf . . . . . 811A Null ROM16
    -libi          SHARED_1
  _put_uart . . . . . 810E Null ROM16
    -uart
  _putchar . . . . . 80AB Null ROM16
    -uart          libi          libp
  _s_printf . . . . . 8460 Null ROM16
    -libp          libi
  _timer_service . . . . . 8063 Null ROM16
    -timers
  _u_printf . . . . . 84F0 Null ROM16
    -libp          libi

```

After final linkages the shared bank address space in the map files BANK0_2.MAP, BANK1_2.MAP and SHARED_2.MAP should be verified for no memory overlap.

TL/DD/10131-30

```

;
; *****
; *
; *      National Semiconductor MicroController Group      *
; *
; *      HPC C Compiler Support and Library Routines      *
; *      C Run Time Initialization User Tunable Code      *
; *      CRTFIRST.ASM - C run time initialization          *
; *****
;

```

```

;Copyright (c) 1987, National Semiconductor, Santa Clara Ca 95051

```

```

;See CRTFIRST.INC source code for explanation of macros and usage

```

```

;Code origin

```

```

.sect crtfirst,rom8,abs=0xffaf

```

```

ld 0x00f3.b,#0xff ;output pins for upper Port B
ld 0x00e3.b,#0x00 ;select bank 0

```

```

jp .

```

```

.incl crtfirst.inc

```

```

.end PROGRAM_start

```

TL/DD/10131-31

```
; SHARED_1.ASM - Bank switch support function
; To force library functions onto shared bank and to
; allocate continuous space for _init_info_ section on the
; shared bank.
;
.incl bankdefs.inc

force_library  printf

init_dummy    18, 6

.end

; BANK0.ASM - Link address for functions actually defined
; in bank0

.incl bankdefs.inc

link_address  initialize_table_memory, 0x85e8
link_address  build_tables, 0x85ed
link_address  capture_table, 0x85f2
link_address  compute_coefficients, 0x85f7
link_address  error, 0x8601
link_address  fatal_error, 0x8606

.end

; BANK1.ASM - Link address for the function actually defined
; in bank1.

.incl bankdefs.inc

link_address  monitor, 0x85fc

.end

; SHARED.ASM - Link address for the functions and variables
; defined in shared address space.

.incl bankdefs.inc

link_address  printf, 0x811a
link_address  putchar, 0x80ab
link_address  live, 0x0a42
link_address  coefficients, 0x0a2e

.end
```

TL/DD/10131-41

```

;
; .title crtfirst, 'C Run Time Initialization'
; *****
; *
; * National Semiconductor MicroController Group *
; *
; * HPC C Compiler Support and Library Routines *
; * CRTFIRST.INC - C Run Time Initialization *
; *
; *****
;
; Copyright (c) 1987, National Semiconductor, Santa Clara Ca 95051
;
; Edit History
; 12/15/86 DKL Create from CCHPC startup output
; 2/6/87 DKL Convert to new Assembler Syntax
; 2/9/87 DKL Separate out Tunable Code
; 3/4/87 RPG Modify to suit new compiler
; 3/10/87 DKL Changes to DKL arrangement, initialize memory
; 3/20/87 DKL Stack out, efficient list order in
; 5/6/87 DKL Make this the included, not includer, file
; 7/27/87 DKL Move Initialization of RAM to separate subroutine
;
;
; .public PROGRAM_start, PROGRAM_exit
; .extrn _main
; .ifndef memories_8bit
; .extrn initialize_memories
; .else
; .extrn initialize_memories_8bit
; .endif
; .extrn STACK_start
;
; .form
; This routine provides the standard C RunTime Routine for starting a
; compiled and linked program. It initializes the stack pointer and
; RAM memories, and enters the compiler generated code in function
; "main()" with no arguments.
;
; Four macros are used to allow the end user to have control of the start
; process at key moments, before the C code begins execution. The macros
; used are ORIGIN, START, READY, and HALT, in the following fashion:
;
; ORIGIN
; PROGRAM_start:
;   |d sp,<stack>
;   START
;   jsrl initialize_memories
;   READY
;   jsrl _main
; PROGRAM_exit:
;   HALT
;
; Code size is tested to ensure that the code does not overwrite any
; dedicated addresses (e.g., subroutine jump table), and optionally to

```

TL/DD/10131-32

```

;ensure that no space is wasted between the end and the dedicated area.
;The dedicated address is defined as ADDRESS_limit, and the check for
;waste space is controlled by ORIGIN_check being non-zero. Either of
;these may be redefined by the user in the ORIGIN macro.

```

```

;ORIGIN macro
;Must declare the section and set the absolute origin for the startup
;code. Code must end before any dedicated addresses (ADDRESS_limit),
;and should not waste any space. If any of the other macros here are
;lenghened, this must be adjusted. Might optionally redefine values
;of ADDRESS_limit or ORIGIN_check.
;

```

```

;START macro
;Code to execute after the stack pointer is initialized, and before the
;memories are initialized. Must enable the appropriate configuration
;options for the chip, so that memories can be accessed. Since all
;memories can be accessed, the list of RAM memories can be accessed
;where ever it may be.
;

```

```

;READY macro
;Code to execute after memory is initialized, but before the C code is
;entered.
;

```

```

;HALT macro
;Code to execute when the C code terminates.
;

```

```

;Limit address of code for this routine (first dedicated address)

```

```

;Whether to check that the origin provided is exactly correct

```

```

        .form
;C RunTime Initialization Startup Code
        ORIGIN ;declares absolute section and defines address

```

```

PROGRAM_start:
        ld     sp,#STACK_start      ;Initialize stack
        START                               ;User code option
        .ifndef memories_8bit
        jsrl   initialize_memories
        .else
        jsrl   initialize_memories_8bit
        .endif
        READY
        jsrl   _main
PROGRAM_exit:
        HALT

```

```

origin= ADDRESS_limit - . + PROGRAM_start
        .if . > ADDRESS_limit
        .ERROR 'Startup Routine overlaps Subroutine Jump Table'
        .else

```

TL/DD/10131-33

```

        .if . < ADDRESS_limit & ORIGIN_check
        .ERROR 'Startup Routine not contiguous to Subroutine Jump Table'
        .endif
    .endif

```

TL/DD/10131-34


```

; .Title Bank Switch, 'Bank Switch Function for Function Calls'
; *****
; *
; * National Semiconductor MicroController Group *
; *
; * HPC Code to Support Inter-Bank Function Calls *
; * BANKSWIT.ASM - Bank switch support functions *
; *****

```

```

;Copyright (c) 1988, National Semiconductor, Santa Clara Ca 95051

```

```

;Edit History
; 3/10/88 DKL Create for Memo/Apps note
; 3/15/88 DKL Add direct support for
; C function names, assembler special
;

```

```

;This is the main switching function to allow inter-bank function calls
;transparent to the compiler and assembler.

```

```

;Requires compilation with the value SWITCH_STACK_DEPTH defined, for the
;number of levels of inter-bank function call nesting to be allowed. The
;value should take into account any interrupt nesting from any interrupt
;service routines which may switch banks.

```

```

;Is called with stack as
; SP ----> Next free location
; SP-2 ---> Intermediate Switch Function Return Address
; SP-4 ---> Destination's Return Address
; SP-6 ---> Destination's Argument 1
; ... Destination's Argument Space
; old sp -> Destination's Argument n
; ... Caller's Local Variable Space
; FP ----> Caller's First Local Variable
; FP-2 ---> Caller's Parent's Frame Pointer
; FP-4 ---> Caller's Return Address
; FP-6 ---> Caller's Argument 1
; ... Caller's Argument Space
;and must call Destination Function with stack in same form, but the
;Destination's Return Address must cause return to the switcher function.

```

```

;An additional stack is necessary to store the additional information so
;the main stack is not polluted. This also requires an additional stack
;pointer.

```

```

        .form
        .macro switch_to      function, bank, address
        .public _function
        _function:
        jsr    function_call_switcher
        .if @ > 1
        .byte  low(address)
        .byte  high(address)
        .byte  bank

```

TL/DD/10131-35

```

    .else
      .byte 0,0,0          ;temporary place holders
    .endif
  .endm ;switch_to

  .macro switch_assembly function, bank, address
    .public function
function:
    jsr function_call_switcher
    .if @ > 1
      .byte low(address)
      .byte high(address)
      .byte bank
    .else
      .byte 0,0,0          ;temporary place holders
    .endif
  .endm ;switch_assembly

    .form
;Bank Switching Control Port
bank_switch_port= 0x00e3:b ;must not touch low byte of Port B

;Values for Bank Switching Control Port
bank0 = 0x00
bank1 = 0x01
bank2 = 0x02
bank3 = 0x03
bank4 = 0x20
bank5 = 0x21
bank6 = 0x22
bank7 = 0x23
bank8 = 0x40
bank9 = 0x41
bank10 = 0x42
bank11 = 0x43
bank12 = 0x60
bank13 = 0x61
bank14 = 0x62
bank15 = 0x63

;Switch stack
.sect switch_stack, ram16, rel
.dsw SWITCH_STACK_DEPTH * 2
growth = 4
.endsect

;Switch stack pointer
.sect switch_pointer, base, rel
switch_stack_pointer: .dsw 1
.endsect

;Initialization value for switch stack pointer
.sect switch_init, rom8, rel
.byte low(e_sect(switch_stack))
.byte high(e_sect(switch_stack))

```

TL/DD/10131-36

```

        .endsect

;Initialization control for switch stack pointer
        .sect   _init_info, rom16, rel
        .word   b_sect(switch_pointer)
        .word   e_sect(switch_pointer) -1
        .word   b_sect(switch_init)
        .endsect

        .sect   switch_code, rom8, rel
;Linkages
        .incl   banklink.inc

;Switch from caller's bank to destination bank, transparently
;All registers must be preserved
;
function_call_switcher:
    push    a                ;free up registers
    push    x
    add     switch_stack_pointer,#-growth ;get switch stack room
    ld     x,switch_stack_pointer
    ld     a,bank_switch_port ;put caller bank on switch stack
    x      a,[x+].w
    ld     a,-8[sp].w        ;put caller return on switch stack
    x      a,[x+].w
    ld     a,-6[sp].w        ;access destination information
    st     a,x
    ld     a,[x+].b         ;get destination address onto stack
    st     a,-6[sp].b
    ld     a,[x+].b         ;(as bytes because no alignment)
    st     a,-5[sp].b
    ld     a,[x+].b         ;put destination bank in port
    st     a,bank_switch_port
    ld     a,#function_call_returner ;put switcher return on stack
    st     a,-8[sp].w
    pop     x
    pop     a
    ret                                ;transfer to destination in new bank
;
;Return to caller's bank from destination bank, transparently
;All registers must be preserved
;
function_call_returner:
    push    a                ;space for return address
    push    a                ;free up register
    ld     a,[switch_stack_pointer].w ;restore caller bank
    st     a,bank_switch_port
    ld     a,2[switch_stack_pointer].w ;restore caller return
    st     a,-4[sp].w
    add     switch_stack_pointer,#growth ;give up switch stack room
    pop     a
    ret                                ;return to caller in original bank
;
        .endsect
        .end

```

```

;
; *****
; *
; *   National Semiconductor MicroController Group   *
; *
; *   Definitions of Inter-Bank Function Call Links   *
; *   BANKLINK.INC - Bank switch support functions   *
; *****

```

;For every inter-bank link required, enter a defining line

```

;
;   switch_to      function, bank<n>, address
;

```

;where the function name is the name of the destination function,

;bank<n> is the name of the bank number (bank0, bank1, ...), and

;the address is a numeric constant for the address of the actual

;destination function code in its bank.

;Assembly language functions can be linked using

```

;
;   switch_assembly function, bank<n>, address
;

```

;instead.

```

switch_to      initialize_table_memory, bank0, 0x4000
switch_to      build_tables, bank0, 0x404b
switch_to      capture_table, bank0, 0x404e
switch_to      compute_coefficients, bank0, 0x40ab
switch_to      monitor, bank1, 0x4000
switch_to      error, bank0, 0x4136
switch_to      fatal_error, bank0, 0x4159

```

TL/DD/10131-38

```

;
; *****
; *
; *   National Semiconductor MicroController Group *
; *
; *   Macros to Assist Bank Switching Linkages *
; *   BANKDEFS.INC - Bankswitch support functions *
; *****

```

```

;For every inter-bank link into a module, substitute definitions
;are needed using the values of the inter-bank link in shared
;memory. These macros make it easier.

```

```

;
;   link_address  function, address
;   link_assembly function, address
;

```

```

;where function is the name of the linked function and address is the
;address of the link code in the shared bank.

```

```

.macro link_address  function, address
    .public  _function
    _function = address
.endm

```

```

.macro link_assembly function, address
    .public function
function = address
.endm

```

```

;For forcing a library routine to be linked, even though not accessed.

```

```

;
;   force_library routine, routine, routine, ...
;   force_assembly routine, routine, routine, ...
;

```

```

;Multiple lines may be used.

```

```

.macro force_library list
    .set $count, 0
    .do @
    .set $count, $count + 1
    .extrn _@$count
    .enddo
.endm ;force_library

```

```

.macro force_assembly list
    .set $count, 0
    .do @
    .set $count, $count + 1
    .extrn @$count
    .enddo
.endm ;force_assembly

```

```

;To create the dummy place holders for the initialization information
;sections.

```

TL/DD/10131-39

```

;
;   init_dummy      size, size, size, ...
;

```

```

;Multiple lines may be used.

```

```

.macro init_dummy      list
    .sect  _init_info_, rom16, rel
    .set $count, 0
    .do @
    .set $count, $count + 1
    .dsb  @$count
    .enddo
    .endsect
.endm ;init_dummy

```

TL/DD/10131-40

```

/*      tables.c      Placed in Bank0.
*/

#include "tables.h"

extern int coefficients[10];
extern struct table_entry live;

#define table_memory      (* ((struct table_entry *) 0x6000))
#define table_memory_end (* ((struct table_entry *) 0x8000))

static int table_entries, table_values;
static struct table_entry * first_table;

/* this initializes special RAM memory in the bank for tables */

NOLOCAL
initialize_table_memory()
{
    static struct table_entry * p;

    /* initialize memory as an array of structure */
    for( p = &table_memory, table_entries = 0;
        p < &table_memory_end;
        p++, table_entries++ )
    {
        p->spins = 0;
        p->rolls = 0;
        p->result = 0;
    }
    /* record initial state */
    first_table = &table_memory;
    table_values = 0;
}

/* builds a series of table entries in the RAM memory from inputs */

NOLOCAL
build_tables()
{
    /*
    ...
    decide when table is ready
    ...
    */
    capture_table();
}

NOLOCAL
capture_table()
{
    static struct table_entry * next;

    if( table_values < table_entries )

```

TL/DD/10131-42

```

    {
        /* table not full, locate next and add one */
        next = first_table + table_values;
    }
    else
    {
        /* table full, advance one as ring */
        next = first_table;
        if( ++first_table >= &table_memory_end )
        {
            first_table = &table_memory;
        }
    }
    *next = live;
}

/* data reduction on table */

NOLOCAL
compute_coefficients()
{
    static int i;
    static struct table_entry * p;

    for( i = 0, p = first_table; i < table_values; i++ )
    {
        /*
         * ...
         * code to do data reduction on available data
         * ...
         */
        recursive_spin_reduction(p, 0);
        if( ++p >= &table_memory_end )
        {
            p = &table_memory;
        }
    }
}

/* reduction on each entry */

static
recursive_spin_reduction(entry, item)
struct table_entry * entry;
int item;
{
    /* ... */
    if( item < entry->spins )
    {
        recursive_spin_reduction(entry, item + 1);
        /* ... */
    }
    /* ... */
}

```

TL/DD/10131-43

```
/*      errors.c      Placed in Bank0.
*/

static int error_count = 0;

NOLOCAL
error(code)
int code;
{
    printf("Error number %i - continuing\n", code);
    error_count++;
}

NOLOCAL
fatal_error(code)
int code;
{
    static int i;

    for( i = 0; i < 15; i++ )
    {
        putchar(0x07);
    }
    printf("\n\nFATAL ERROR number %i - ABORTING PROCESSING\n\n",
        code);
    quit();
}

NOLOCAL
quit()
{
    printf("Program terminated.  %i recoverable errors\n",
        error_count);
}
}
```

TL/DD/10131-44


```
/*      monitor.c      Placed in Bank1.
*/

#include "tables.h"

extern struct table_entry live;

NOLOCAL
monitor()
{
    static int predictable;

    /*
    ...
    system monitoring
    */
    while( live.spins < 3
           || live.rolls < 5 );
    while( !live.result )
    {
        compute_prediction();
    }
    validate_calculation();
    capture_table();
}

compute_prediction()
{
    int i;

    /*
    ...
    complex calculations to give a SWAG
    */
    printf("Prediction: %i\n", i);
}

validate_calculation()
{
    int i, j, k;

    /*
    ...
    match latest result to what we would predict
    */
    printf("Final prediction: %i, actual: %i, accuracy: %i\n", i, j, k);
    if( k < 10 )
    {
        error(1);
    }
}
}
```

TL/DD/10131-45

```
/*
   main.c          Placed in Shared.
   This is the main program for the example.
*/

/*operational mode flags */
int operational = 1,
  calibrating = 1,
  predicting = 0;

/* controlling coefficient array */
int coefficients[10];

main()
{
  initialize_inputs();
  initialize_outputs();
  initialize_table_memory();
  while( operational )
  {
    while( calibrating )
    {
      build_tables();
    }
    compute_coefficients();
    while( predicting )
    {
      monitor();
    }
  }
}
```

TL/DD/10131-46

High Performance Controller in Information Control Applications

National Semiconductor
Application Note 585
Steve McRobert



ABSTRACT

This paper describes National Semiconductor's HPC™ family of High Performance microControllers. Included are two examples showing how the devices are used in actual Information Control applications.

The architecture, technology, and instruction set of the HPC family are presented, with emphasis on how these features are appropriate for use in microcontroller based information control systems. Two example applications are given, the first being the use of a single chip mode HPC as an I/O processor and interrupt handler in a laser beam printer. In this case the HPC acts as a slave to the main 32-bit CPU in the printer, freeing it from the many tasks which require fast interrupt response and thus improves system throughput. The second example shows the HPC used in expanded mode as the sole microprocessor in an ESDI to SCSI bridge adapter card. The operations performed by the HPC in this application are used as an example of how the instruction set and addressing modes work together to achieve high throughput. The paper concludes with a brief discussion of the future of the HPC family of devices.

INTRODUCTION

The HPC (High Performance Controller) family of microcontrollers was designed by National Semiconductor as the first of a new generation of 16-bit CMOS microcontrollers.

The intention was to start afresh, using the experience gained from earlier device families and, without software

compatibility constraints, to create an architecture sufficiently advanced to be competitive for 10 years or more. Other design goals were to minimize device complexity, thus allowing for dependable, economical, high volume production, and to make HPC easy to understand so that system designers could readily convert designs to use the new family's advanced features.

These goals have been met, and, since the first device was sampled in early 1986, the HPC family has developed into a well proven solution to many design problems.

ARCHITECTURE

The HPC family is based on a core concept. All devices share a common core including the CPU and a base set of peripherals such as timer/counters etc. *Figure 1* shows a block diagram of the HPC16083 with the core emphasized at left. HPC uses a memory-mapped Von Neuman architecture, in which all registers, I/O ports, peripherals etc. are assigned memory locations in one uniform address space.

This includes the CPU registers (*Figure 1*), allowing all HPC instructions to operate on every register in the programmer's model. Such uniformity simplifies the work of the assembly language programmer and the writer of the C compiler, making the HPC a particularly efficient microcontroller for running programs written in "C".

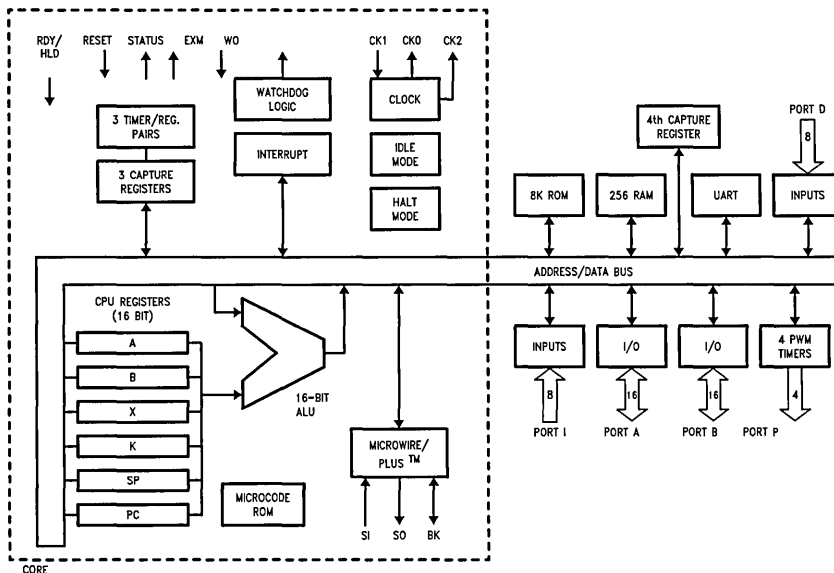


FIGURE 1. HPC16083 Block Diagram

TL/DD/10346-1

The core is connected to peripherals and on-chip memory by a 16-bit address/data bus, which is multiplexed to reduce die size. This bus is brought out on the A port when the device is used in expanded and/or ROMless modes, allowing off-chip devices to be accessed in exactly the same fashion as on-chip memory or peripherals.

When writing assembly language or C instructions the programmer perceives no difference between on-chip and off-chip memories, but both assembler and compiler take account of two key differences. When the HPC is run at high oscillator frequencies (up to 30 MHz on current production devices) a wait state must be applied for accesses to external memories or peripherals, but are never applied to on-chip RAM or registers. The other difference is that accesses to on-chip locations with addresses below 100 hexadecimal (called basepage accesses) require only a one byte address, so are thus shorter and faster than accesses to non-basepage locations (Figure 2).

FFF:FFF0	INTERRUPT VECTORS	HPC16083 ON-CHIP ROM SPACE
FFEF:FFD0	JSRP VECTORS	
FFCF:E000	GENERAL PURPOSE ROM	
DFFF:0200	EXPANDED MODE ADDRESS SPACE	EXTERNAL USER MEMORY
01FF:01C0	ON-CHIP RAM	ON-CHIP RAM AND REGISTERS
01BF:00C0	ON-CHIP REGISTERS	
00BF:0000	ON-CHIP RAM	

FIGURE 2

The programmer must choose which variables to put into on chip RAM or the basepage to achieve maximum performance and code efficiency.

Basepage RAM, because it is very fast and efficient to use, provides many of the benefits of the register file architecture used on some other microcontrollers. The HPC is different, however, in that it has a small set of registers: Accumulator, B pointer, X pointer and K (or limit) register. These registers all have addresses and can be used as general purpose memory locations, but are best used for their special func-

tions. Many HPC instructions have two operands, the source and the destination. If the Accumulator (A) register is used as the destination, this is implied in the opcode and the address of A need not be included in the instruction, thus making it shorter and faster than instructions using another memory location as the destination. If the address of the source is contained in the B register then this too can be implied from the opcode and the whole instruction becomes one byte long.

Most HPC instructions thus have a single-byte form, using the B or X register as a pointer to the memory location being accessed.

The use of the K register will be discussed in the next section.

The primary objective when designing the architecture and instruction set of HPC was to minimize code size, an approach which can reduce throughput if unlimited bus bandwidth is available. In typical microcontroller applications the use of external memory is undesirable for board space and cost reasons. If the code is too large for mask ROM, the best solution in terms of space and cost is a single, relatively slow, EPROM.

In this situation of low bus bandwidth, the high byte efficiency of the HPC goes hand-in-hand with good performance.

ADDRESSING MODES

In keeping up with the HPC philosophy of being simple and quick to understand, the HPC instruction set (Figure 3) has relatively few mnemonics. This is because for those instructions with one or two addressable operands the same mnemonic is used regardless of the addressing mode, operand size (byte or word) or address size (depending upon whether each operand is in the basepage or not). Each individual memory location may be addressed using one of the following addressing modes:

Direct: The 8- or 16-bit address is included in the series of bytes that make up the instruction.

Indirect: The 8-bit address of a word in the base page is included in the instruction. The contents of this word are used as a pointer to the variable to be accessed.

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	MA + Mem1 → MA carry → C
ADC	Add with carry	MA + Mem1 + C → MA carry → C
ADDS	Add short imm8	MA + imm8 → MA carry → C
DADC	Decimal add with carry	MA + Mem1 + C → MA (Decimal) carry → C
SUBC	Subtract with carry	MA - Mem1 + C → MA carry → C
DSUBC	Decimal subtract w/carry	MA - Mem1 + C → MA (Decimal) carry → C
MULT	Multiply (unsigned)	MA * Mem1 → MA & X, 0 → K, 0 → C
DIV	Divide (unsigned)	MA / Mem1 → MA, rem. → X, 0 → K, 0 → C
DIVD	Divide Double Word (unsigned)	(X & MA) / Mem1 → MA, rem → X, 0 → K, carry → C
IFEQ	If equal	Compare MA & Mem1, Do next if equal
IFGT	If greater than	Compare MA & Mem1, Do next if MA > Mem1
AND	Logical and	MA and Mem1 → MA
OR	Logical or	MA or Mem1 → MA
XOR	Logical exclusive-or	MA xor Mem1 → MA
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	Mem + 1 → Mem
DECSZ	Decrement, skip if 0	Mem - 1 → Mem, Skip next if Mem = 0

FIGURE 3. HPC Instruction Set Description

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	MemI → MA
ST	Load, incr/decr X	Mem(X) → A, X ± 1 (or 2) → X
X	Store to Memory	A → Mem
	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ± 1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP
POP	Pop Stack to Memory	SP - 2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ± 1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A, B ± 1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B, imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A - 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → [SP], SP + 2 → SP W(table #) → PC
JSR	Jump subroutine relative	PC → [SP], SP + 2 → SP, PC + # → PC (# is + 1025 to - 1023)
JSRL	Jump subroutine long	PC → [SP], SP + 2 → SP, PC + # → PC
JP	Jump relative short	PC + # → PC (# is + 32 to - 31)
JMP	Jump relative	PC + # → PC (# is + 257 to - 255)
JMPL	Jump relative long	PC + # → PC
JID	Jump indirect at PC + A	PC + A + 1 → PC then Mem(PC) + PC → PC
JIDW		
NOP	No Operation	PC + 1 → PC
RET	Return	SP - 2 → SP, [SP] → PC
RETSK	Return then skip next	SP - 2 → SP, [SP] → PC, & skip
RETI	Return from interrupt	SP - 2 → SP, [SP] → PC, interrupt re-enabled
<p>Note: W is 16-bit word of memory MA is Accumulator A or direct memory (8 or 16-bit) Mem is 8-bit byte or 16-bit word of memory MemI is 8- or 16-bit memory or 8 or 16-bit immediate data imm is 8-bit or 16-bit immediate data imm8 is 8-bit immediate data only</p>		

FIGURE 3. HPC Instruction Set Description

```

LD X, #0100      ; Point to beginning of source code
LD BK, # 0400, #0600;P ; Point to beginning & end of target
LOOP: LD A, [X+].W ; Get word from source block
XS A, [B+].W     ; Store it at target
JP LOOP

```

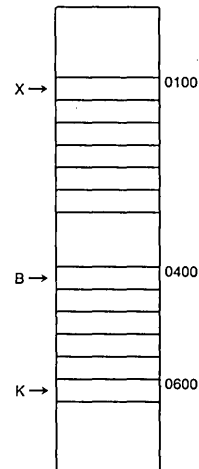


FIGURE 4. Word Block Move

- Indexed:** As indirect, but with an 8- or 16-bit immediate offset added to the pointer.
- Register Indirect:** As indirect, but the B or X registers are used as pointers, with their addresses implied in the opcode.
- Immediate:** Only for the source in two-operand instructions. An 8- or 16-bit immediate value is included in the instruction.

The first four addressing modes are used both for single operand instructions e.g. bit set, bit clear, bit test, increment, decrement, and two operand instructions such as ADD and LD.

Direct and immediate modes can be used in combination, allowing operations to be performed directly on memory or registers without using the accumulator.

Two variables, each byte or word, each located anywhere in memory, can be compared, added, divided or have any of the other two-address instructions performed on them. This improves the byte-efficiency of the HPC, and enhances the power of the instruction set in that it takes less lines of assembly code to perform a given function than it would for earlier, completely accumulator-based CPUs.

An important benefit provided by the indirect and indexed modes is that any of the 96 words of RAM or the basepage registers, such as port A or the accumulator, may be used as pointers.

There are two special addressing modes which are used only with the LD and X (exchange) instructions. These modes are called auto increment/decrement and auto increment/decrement with conditional skip, and their use is illustrated by the example shown in *Figure 4*.

This example uses the B pointer, the X pointer and the K register to move a block of data one word at a time. Some points to note are that the LD BK instruction initializes both registers with one instruction, and that both the LD and XS instructions increment the pointer by two because two bytes (one word) are moved. The S in XS signifies the conditional

skip. After A has been exchanged with the word pointed to by B, B is incremented, then compared with K. If B is greater than K (or, for an XS A, [B-] instruction, if B is less than K) the next statement is skipped over, thus terminating the loop. This example epitomizes the approach taken in designing the HPC family.

String operations are built up from simple data movement instructions, allowing them to be interrupted at any time with no need for complex re-start or recovery schemes.

INSTRUCTION SET

The HPC instruction set is noticeably different from other 16-bit controllers, in that many of its instructions are single byte. How this is achieved can be seen by looking at the opcode map (*Figure 5*).

Instructions such as bit manipulation operations and single byte jumps (JP) use many opcodes for the same mnemonic. This is because information, such as the jump length for JP, is coded into the opcode.

This makes these instructions very efficient, and enhances the performance of the HPC in information control applications, where decision making and bit manipulation operations tend to be important.

All of the arithmetic, comparison, logical and data movement instructions have a single byte form using register indirect addressing mode. The opcode space "used up" by having many opcodes for a few instructions is restored by using addressing mode prefixes for the less commonly used addressing modes. These make instructions using these modes one byte longer, but the use of these prefixes allows all of the two address instructions to use all of the addressing modes. Without the prefixes the HPC would run out of opcode space and restrictions would have to be placed on some instructions, making the assembly language much harder to use and the C compiler harder to write. Examples are given in *Figure 6* of several combinations of instructions and addressing modes, with execution times for systems using low cost external memories.

C.13 HPC OPCODE MAP
LSB/MSB →

	0	1	2	3	4	5	6	7
0	CLR A	IFBIT 0	JSRP 0	JSR +	JP +1*	JP +17	JP 0	JP -16
1	COMP A	IFBIT 1	JSRP 1	JSR +	JP +2	JP +18	JP -1	JP -17
2	SC	IFBIT 2	JSRP 2	JSR +	JP +3	JP +19	JP -2	JP -18
3	RC	IFBIT 3	JSRP 3	JSR +	JP +4	JP +20	JP -3	JP -19
4	INC A	IFBIT 4	JSRP 4	JSR -	JP +5	JP +21	JP -4	JP -20
5	DEC A	IFBIT 5	JSRP 5	JSR -	JP +6	JP +22	JP -5	JP -21
6	IFNC	IFBIT 6	JSRP 6	JSR -	JP +7	JP +23	JP -6	JP -22
7	IFC	IFBIT 7	JSRP 7	JSR -	JP +8	JP +24	JP -7	JP -23
8	SBIT 0	RBIT 0	JSRP 8	RBIT X	JP +9	JP +25	JP -8	JP -24
9	SBIT 1	RBIT 1	JSRP 9	SBIT X	JP +10	JP +26	JP -9	JP -25
A	SBIT 2	RBIT 2	JSRP 10	IFBIT X	JP +11	JP +27	JP -10	JP -26
B	SBIT 3	RBIT 3	JSRP 11	SWAP A	JP +12	JP +28	JP -11	JP -27
C	SBIT 4	RBIT 4	JSRP 12	RET	JP +13	JP +29	JP -12	JP -28
D	SBIT 5	RBIT 5	JSRP 13	RETSK	JP +14	JP +30	JP -13	JP -29
E	SBIT 6	RBIT 6	JSRP 14	RETI	JP +15	JP +31	JP -14	JP -30
F	SBIT 7	RBIT 7	JSRP 15	POP	JP +16	JP +32	JP -15	JP -31
	8	9	A	B	C	D	E	F
0	Dir-Dir	LD A,i	Dir-Dir	LD A,ii	LDS [B+].b	LD [X+].b	LDS [B+].w	LD [X+].w
1	Dir-Dir	LD K,i	Dir-Dir	LD K,ii	XS [B+].b	X [X+].b	XS [B+].w	X [X+].w
2	Imm-Dir	LD B,i	Index	LD B,ii	LDS [B-].b	LD [X-].b	LDS [B-].w	LD [X-].w
3	Imm-Dir	LD X,i	—	LD X,ii	XS [B-].b	X M[X-].b	XS [B-].w	X [X-].w
4	Dir-Dir	JMP +	Dir-Dir	JMPL	LD [B].b	LD [X].b	LD [B].w	LD [X].w
5	Dir-Dir	JMP -	Dir-Dir	JSRL	X [B].b	X [X].b	X [B].w	X [X].w
6	Imm-Dir	Direct	Index	Direct	ST [B].b	ST [X].b	ST [B].w	ST [X].w
7	Imm-Dir	LD bd,i	LD BK,ii	LD wd,ii	SHR A	RRC A	SHL A	RLC A
8	LD A,bd	ADD A,i	LD A,wd	ADD A,ii	ADC A,b	ADD A,b	ADC A,w	ADD A,w
9	INC bd	AND A,i	INC wd	AND A,ii	DADC A,b	AND A,b	DADC A,w	AND A,w
A	DECSZ bd	OR A,i	DECSZ wd	OR A,ii	DSUBC A,b	OR A,b	DSUB A,w	OR A,w
B	ST A,bd†	XOR A,i	ST A,wd†	XOR A,ii	SUBC A,b	XOR A,b	SUBC A,w	XOR A,w
C	LD bd,bd	IFEQ A,i	LD wd,wd	IFEQ A,ii	JID	IFEQ A,b	JIDW	IFEQ A,w
D	LD BK,i	IFGT A,i	Indirect	IFGT A,ii	—	IFGT A,b	—	IFGT A,w
E	X A,bd	MULT A,i	X A,wd	MULT A,ii	—	MULT A,b	—	MULT A,w
F	XIndirect	DIV A,i	PUSH	DIV A,ii	DIVD A,b	DIV A,b	DIVD A,w	DIV A,w

— = opcode is reserved for future use.

b = byte of memory

bd = direct byte of memory

i = 8-bit immediate value

w = word of memory

wd = direct word of memory

ii = 16-bit immediate value

Dir-Dir, Imm-Dir, Index, Direct, Indirect and XIndirect are all Addressing Mode directives.

Notes:

*NOP is the same as JP + 1 and has the same opcode.

†These opcodes are LD if prefixed by Dir-Dir or Imm-Dir directive.

FIGURE 5

		20 MHz	30 MHz
CLR	A	300 ns	200 ns
RRC	A	400 ns	267 ns
LD	B, H'3CF2	600 ns	400 ns
IFBIT	7,[B].B	800 ns	533 ns
ST	A,38.W	900 ns	600 ns
JSR		1.10 μ s	733 μ s
JSRL		1.40 μ s	933 μ s
ADC	[H'10].W, [H'20].W	1.70 μ s	1.13 μ s
DSUBC	[H'A0].W, [H'B0].W	2.00 μ s	1.33 μ s
MULT	A, [B].W	5.90 μ s	3.93 μ s
DIVD	A,[X].W	6.40 μ s	4.27 μ s

Times Calculated with 1 Wait State Inserted

FIGURE 6. Typical Execution Times

There are many more powerful features of the HPC instruction set, but space does not permit describing them here. For more information see the documents listed in the references section.

TECHNOLOGY

The HPC family and nearly all other new National Semiconductor analog and digital VLSI devices are fabricated in an advanced double metal process called M2CMOS. This is a very high speed process, as shown by the current production two micron (drawn) HPC46083, which is available as a 30 MHz version.

The HPC family has been migrated to a 1.5 micron (drawn) process for the first part with an analog to digital converter on chip, the HPC46164.

National Semiconductor already manufactures the NS32532 microprocessor in 1.25 micron M2CMOS, and will shrink this process still further in the future. The HPC devices will be migrated to these smaller geometries and will benefit from other process developments such as on chip EPROM.

INFORMATION CONTROL APPLICATIONS

Laser Beam Printer Front End Processor

This section describes a customer's application for an HPC46083 used in single chip mode. It makes use of the Universal Peripheral interface (UPI) port which is a feature of all HPC devices with on-chip mask ROM.

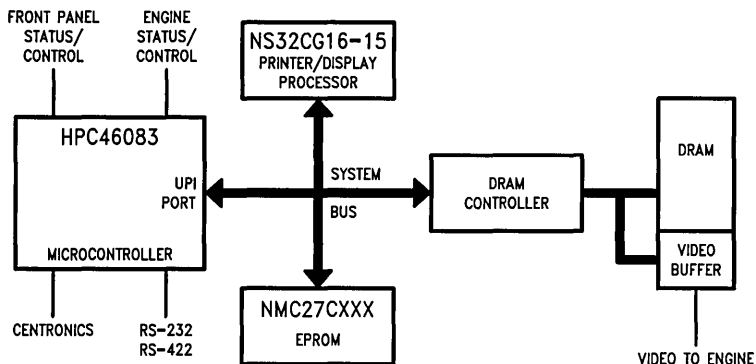


FIGURE 7

TL/DD/10346-2

The UPI port allows an HPC device to be used as a peripheral to a host processor, connected to the host via its data bus. The HPC in UPI mode appears to the host to be a peripheral device such as a UART, but provides additional processing power, relieving the host of interrupt-intensive tasks and thus improving the host's performance.

The UPI port of the HPC provides status signals to both the HPC CPU and that of the host which ensure that no data is lost when the CPUs communicate.

In the laser beam pointer (LBP) application (Figure 7), the HPC handles the serial and Centronics interfaces of the printer, buffering received characters and interrupting the host CPU when a block of up to 128 characters has been received. When the host CPU (a National Semiconductor NS32CG16 printer/display controller) is interrupted it then transfers the whole block of data into its own memory very rapidly.

This approach reduces the number of interrupts received by the 32CG16 by a factor of over 100 compared to a solution using a conventional UART while being simpler, cheaper and offering higher system performance than using a DMA approach. These overhead reductions are very important in LBP systems, because the main CPU must keep up with the paper movement, otherwise image data will be lost.

In addition to improving printer performance, the HPC reduces the system cost by providing functions that would otherwise need extra devices. The HPC acts as the interrupt controller for the 32CG16, generating an interrupt signal to it and then placing the interrupt vector on the UPI port when the 32CG16 acknowledges the interrupt. Another function provided by the HPC is an intelligent interface to the printer front panel displays and push buttons controlling such functions as LCD contrast. Finally, the HPC implements a serial interface to the electronic subsystem of the printer engine itself, providing diagnostic capability to the 32CG16. For all of these functions, the HPC performs first-level error checking, further relieving the main CPU of minor tasks.

The LBP is at one extreme of the range of HPC applications, where the HPC uses virtually nothing but its on-chip peripherals and memories.

The next section deals with an application towards the other end of the range.

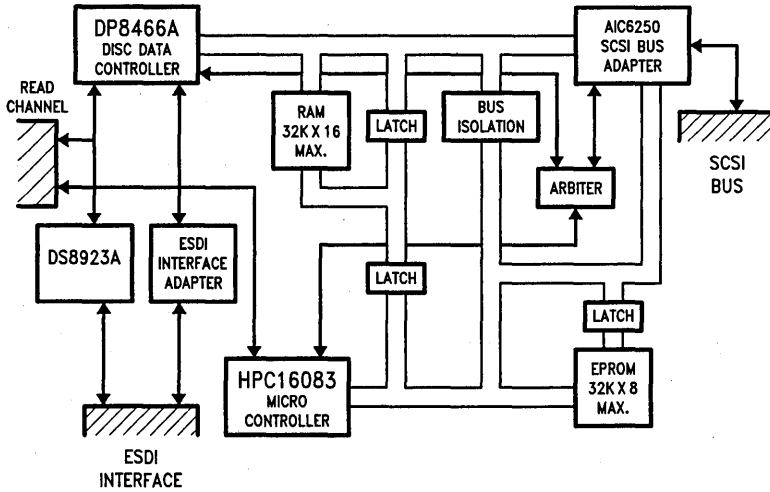


FIGURE 8

TL/DD/10346-3

SCSI Bridge Adapter

The fast growing usage of Winchester disk drives in the Small Computer System Interface (SCSI) environment has provided another important market for the HPC family.

The HPC architecture is well suited for use in embedded SCSI systems, as the peripherals such as the SCSI interface device may be memory mapped into the HPC address space, allowing bit and byte manipulation operations to be performed directly on the registers of the peripheral using single assembly language instructions. Many SCSI interface devices are relatively unintelligent, requiring the CPU to perform many bit test, set, and clear operations to set up a data transfer operation. Most other microcontrollers need up to three instructions to set a bit in one of these peripherals, thus reducing drive performance.

National Semiconductor has produced an ESDI-to-SCSI bridge adapter board, which demonstrates the use of the HPC46003 and the DP8466A disk data controller in a real synchronous SCSI system. A software package has been written in HPC assembly language which implements the SCSI common command set and is available in source code form to companies wishing to use the HPC in embedded SCSI or host adapter designs.

The code was written in HPC assembly language because for very high volume, cost sensitive designs, like a disk drive, the extra development cost of writing in assembler is outweighed by the advantages of reduced code size and improved performance.

The adapter board design (Figure 8) uses the HPC46003 running in 8-bit mode with a single EPROM providing program memory. Data memory is provided by the 256 bytes of on-chip RAM which provides fast scratch pad and stack space.

One important function in embedded SCSI disk drives is logical to physical address conversion, in which a logical address (typically 24 bits) is divided twice by constants, the result and the two remainders being the head, cylinder and sector numbers.

The HPC is capable of dividing a 32-bit number in under four microseconds, thus providing a dramatic improvement in logical to physical address conversion time compared to earlier 8-bit microcontroller solutions. As a final point in this necessarily brief discussion, the HPC uses very little power due to its advanced CMOS manufacturing technology. This is important in disk drive applications, where low power consumption is an important performance parameter for the end product.

CONCLUSION AND FUTURE DEVELOPMENTS

This paper has discussed the design of the HPC family and described two actual applications in important market areas. The development work performed for these and other projects has shown that the HPC architecture provides very high performance in embedded control applications.

The plans for future products are to take the high performance core and add various combinations of peripherals, thus allowing the family to reach a wide range of markets. *Figure 9* shows some of the current and future devices.

HPC16083	8K ROM, 256 RAM
HPC16003	ROMless, 256 RAM
HPC16400	256 RAM, 2 HDLC + 4 DMA Channels
HPC16164	16K ROM, 512 RAM, 8 Channel ADC
HPC16064	16K ROM, 512 RAM
HPC16104	ROMless, 8 Channel ADC
HPC16004	ROMless, 512 RAM
HPC167164	16K EPROM, 512 RAM, 8 Channel ADC

FIGURE 9. HPC Family Devices Principal Features

REFERENCES

- National Semiconductor Application Note AN-510: *Assembly Language programming for the HPC.*
- National Semiconductor Publication Number 424410897-001A July 1987: *HPC16083/HPC16043/HPC16003 User's Manual.*

Pulse Width Modulation Using HPC

As the use of MicroControllers in embedded control applications grows in popularity, we find more use of width modulated pulse trains. Typical applications that use Pulse Width Modulation are automotive engine control, motor speed control, display intensity control, and sound generation.

PWM DEFINITION

Pulse width modulation is simply a method of communicating information to a device. It can be viewed as an analog signal provided in digital form. *Figure 1* shows a typical timing diagram of a PWM signal. The duty cycle is expressed as the duration of T_{on} over the sum of T_{on} and T_{off} . A signal has a constant duty cycle if T_{on} and T_{off} are uniform. If T_{on} is equal to T_{off} , the signal has a 50% duty cycle.

$$\text{Duty Cycle} = \frac{T_{on}}{T_{on} + T_{off}}$$

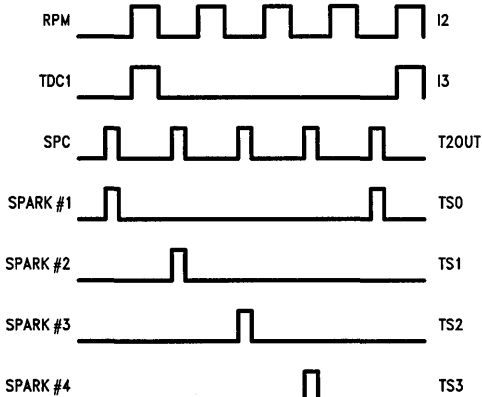


TL/DD/10347-1

FIGURE 1. A Typical PWM Signal

TYPICAL APPLICATIONS THAT REQUIRE PWM

One element of an automotive engine control system is the spark ignition. In a distributorless ignition system, spark control signals are required to appear in sequence, with a time delay between each of them. Typical signals for a four spark plug system are shown in *Figure 2*. The generation of these signals will be explained further in the timer synchronous output section.



TL/DD/10347-2

FIGURE 2. HPC Based Spark Ignition Control

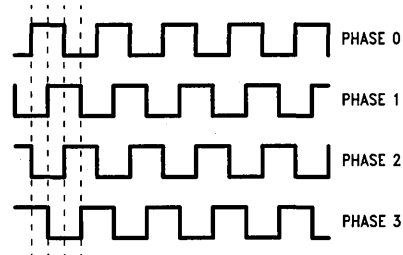
Another element of an automotive system is the carburetion and idle speed control. When no pressure is applied to the

National Semiconductor
Application Note 586
Alvin Chan, Bill Miller,
Bob Moeckel, Bob Hanrahan



accelerator pedal, the throttle is completely shut off. The idle speed control utilizes a stepping motor to operate an auxiliary fuel valve. *Figure 3* shows the control signals that have to be generated for a four phase stepper motor. Each of the PWM signals should have a phase lag of one quarter of a cycle from the previous one.

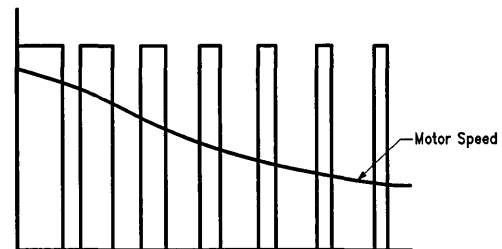
PWM is applied to motor speed control. The speed of a dc motor is directly proportional to the voltage applied.



TL/DD/10347-3

FIGURE 3. Stepper Motor Control Signals

PWM is used selectively to switch full supply power on and off to the motor at some frequency and duty cycle. The bigger the duty cycle, the more power is supplied to the motor. Hence the speed is higher. Motor speed can be controlled by adjusting the ON time of the signal. *Figure 4* depicts the relationship of motor speed and the applied signal.



TL/DD/10347-4

FIGURE 4. Using PWM to Control Motor Speed

The same manipulation also applies to controlling the intensity of light emitting diodes. The brightness of the LED can be varied by using different duty cycles.

Sound synthesis can be achieved by uniting the process of sinusoidal signal generation and envelope generation.

A sinusoidal signal can be generated by a variety of methods. A common technique is to use Walsh functions. Walsh functions are the digital equivalent of Fourier Series. They are essentially pulse signals with varying duty cycles. The individual Walsh components are generated by the microcontroller and combined with the proper weighting factors to form the sinusoids.

Envelope generation can be done by using PWM to build a D/A converter. The envelope will give the composite sinusoidal signal the characteristic sharp attack followed by slow decay. The amplitude of the envelope function is altered by changing the duty cycle of the PWM input to the D/A converter. This function is performed by another timer.

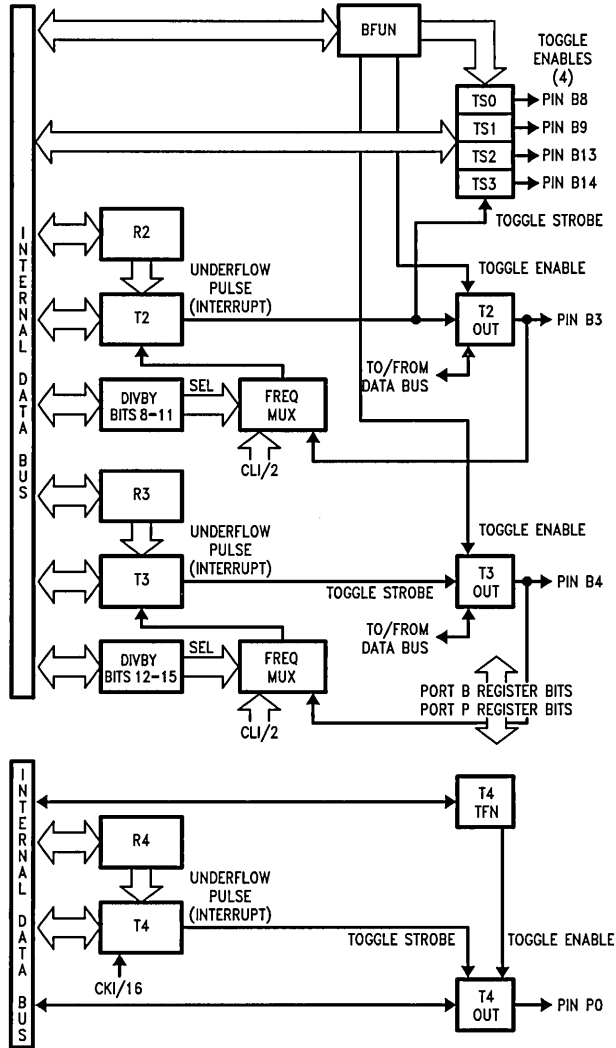
HPC Implementation

National Semiconductor's HPC, High Performance Micro-Controller, provides a simple method for generating width modulated pulse trains, with little or no software overhead, by use of the device's 9 on-chip timers, T0 through T8.

SETTING UP HPC TO DO PWM

PWM Outputs in the HPC

Timers T1 through T7 are down-counters with associated input registers R1 through R7. The value in the registers is loaded automatically into the timers when the timers underflow. Timers T2 through T7 have individual output signals which toggle when the timers underflow. Interrupts are generated at the time of underflow *Figure 5* shows the structure of these timers.



Note: Only Time 4 is Shown. T5, T6, and T7 are identical.

FIGURE 5. HPC Timers T2-T7

TL/DD/10347-5

Timers T2 through T7 can be separated into 2 groups. Different procedures and registers are used to set up the two groups of timers. In one group is timers T4 through T7, which are dedicated to PWM applications. They count down at a constant rate of $\frac{1}{16}$ of the input clock (CKI/16) while enabled to do so. In the other group are the more versatile timers, T2 and T3. The clock input to timers T2 and T3 may be independently selected as coming from one of 14 available prescaled versions of the CKI clock, or from an external pin, as specified in the DIVBY register. Timer T2 can also be specified to be clocked on underflows from timer T3 by appropriate selection in the DIVBY register; the pair then form, in effect, a single 32-bit counter.

With timers T4 through T7, the maximum PWM frequency that can be achieved is half of CKI/16. The associated register provides a 16-bit resolution for the duration of the pulse width.

To use T2 and T3 as PWM timers, the clock must come from an internal source. By configuring the DIVBY register and selecting a value for the counter, the maximum frequency that can be achieved is half CKI/16 and the minimum frequency is half (CKI/131072)/65536.

50% Duty Cycle PWM

On underflow of the timers T2 through T7, the value in the corresponding input register is automatically reloaded into the counters. Therefore a 50% duty cycle PWM can be generated without software intervention once the timer is set up.

Listings 1 and 2 illustrate the use of T4 and T2 in generating PWM outputs. The PWM frequency to be generated is 20 kHz. By using a 16 MHz crystal and CKI/16 as the input clock, the counter value to be loaded into the registers is 24 so that an underflow occurs and the output toggles every 25 μ s.

Generating Non 50% Duty Cycle PWM without Listing 1 Use of T4 to Generate 50% Duty Cycle

```
.TITLE 'T4 PWM 50% DC'
.SECT CODE,ROM16,REL
TMMODE = 0190:W
DIVBY = 018E:W
T4 = 0140:W
R4 = 0142:W
T5 = 0144:W
R5 = 0146:W
PWMODE = 0150:W
PORTP = 0152:W
PWMSTR: LD SP,#STKS ;initialize stack pointer
LD PWMODE,#0x4 ;stop timer T4
NOP ;delay to provide 8
;CK2 cycles
NOP ;to make sure timer
;is updated
LD PWMODE,#0xC ;clear T4
;interrupt pending bit
LD T4,#24 ;load T4 with counter
;value to obtain a 20 kHz PWM
;frequency the counter should
;underflow on a 40 kHz frequency,
;therefore by using a 16 MHz crystal
;and CKI/16 input to the timer, the
;counter value should be 24
;16 MHz/16/25 = 40 kHz
LD R4,#24 ;load auto-reload
;register
SBIT 0,PORTP ;set initial value of
;output pin for T4 to 0
SBIT 3,PORTP ;enable toggling of
;pin on underflow
RBIT 2,PWMODE ;start timer
STOP: JP STOP
.ENDSECT

.SECT STACK,BASE
STKS: DSW 10
.ENDSECT
.END PWMSTR
```

Non 50% Duty Cycle PWM (Software/Interrupts)

Timers T1 through T7 will generate an interrupt on underflow. For non-50% duty cycle PWM, software has to be involved in controlling the duty cycle. The same software for the 50% duty cycle is used to set up the timers for counting down. On interrupt from the timers, the interrupt service routine loads the other half of the cycle time into the timer register.

On each interrupt from the timer the user software alternately loads T_{on} and T_{off} into the register. The result is a constant duty cycle output. Examples of programming the interrupts are shown in listings 3 and 4.

TIMER SYNCHRONOUS OUTPUTS OF TIMER T2

Timer T2 has in addition to the normal output pin, four output pins which can be independently selected. These pins are referred to collectively as the "Timer Synchronous" outputs. *Figure 2* shows the synchronous output being applied

to engine control in spark ignition. The signals TS0 to TS3 are synchronous outputs derived from timer T2. By enabling each pin in sequence, the spark control signals SP1 to SP4 can be generated.

SOFTWARE INTERVENTION

Another problem facing the designer of a MicroController based system is that software overhead must be kept to a minimum. Interrupt latency and changing input registers can use a significant portion of the time which would otherwise be available for processing of sensor data.

The conventional way of generating non-50% duty cycle was discussed earlier. That involves software changing the value of the auto-reload register every time the timer counts down and interrupts. Two timers can be used to generate two synchronized and offset 50% duty cycle pulses. By EXCLUSIVE-ORing them, a non-50% duty cycle PWM is generated.

Listing 2 Use of T2 to Generate 50% Duty Cycle

```

.TITLE 'T2 PWM 50% DC'
.SECT CODE,ROM16,REL
TMMODE = 0190:W
DIVBY = 018E:W
BFUN = 00F4:W
DIRB = 00F2:W
PORTB = 00E2:W
T2 = 0188:W
R2 = 0186:W
PWMSTR: LD SP,#STKS ;initialize stack pointer
LD TMMODE,#0x400 ;stop timer T2
NOP ;delay to provide 8
NOP ;CK2 cycles to make
;sure timer is updated
LD TMMODE,#0xC00 ;clear T2
;interrupt pending bit
SBIT 3,BFUN ;set pin 3 of port B
;as timer 2 output
SBIT 3,DIRB ;set output direction
;on port B pin 3
LD DIVBY,#0x200;set clock as CKI/16
;for T2
LD T2,#24 ;load T2 with counter
;value to obtain a 20 kHz PWM
;frequency the counter should
;underflow on a 40 kHz frequency
;therefore by using a 16 MHz crystal
;and CKI/16 input to the timer, the
;counter value should be 24
;16 MHz/16/25 = 40 kHz
LD R2,#24 ;load auto-reload
;register
RBIT 3,PORTB ;initialize output pin
;value to 0
RBIT 3,TMMODE ;start timer

STOP: JP STOP
.ENDSECT

.SECT STACK,BASE
STKS: DSW 10
.ENDSECT
.END PWMSTR

```

Listing 3 Use of T4 to Generate Non-50% Duty Cycle with Interrupts

```

.TITLE 'T4 NON-50% DC'
.SECT CODE,ROM16,REL

TMMODE = 0190:W
DIVBY  = 018E:W
T4     = 0140:W
R4     = 0142:W
T5     = 0144:W
R5     = 0146:W
PWMODE = 0150:W
PORTP  = 0152:W
ENIR   = 00D0:B
IRPD   = 00D2:B
PWMSTR: LD SF,#STKS ;initialize stack pointer
LD PWMODE,#0x4;stop timer T4
NOP ;delay to provide 8
NOP ;CK2 cycles to make
;sure timer is updated
LD PWMODE,#0xC ;clear T4
;interrupt pending bit
LD ENIR,#00 ;disable interrupts
LD IRPD,#00 ;clear interrupt
;pending bits
LD T4,#9 ;load T4 with counter
;value to obtain a 20 kHz PWM
;frequency with 20% duty cycle
;using a 16 MHz crystal and CKI/16
;input to the timer, the counter
;value should be 9
LD R4,#39 ;load auto-reload
;register with count
;of 80%
LD TCYCLE,#48 ;set total cycle time
;count -2
SBIT 0,PORTP ;set initial value of
;output pin for T4 to 0
SBIT 3,PORTP ;enable toggling of
;pin on underflow
LD ENIR,#0x20 ;enable timer interrupt
RBIT 2,PWMODE ;start timer

STOP:
JP STOP
.ENDSECT

.STECT STACK,BASE
STKS: DSW 10
.ENDSECT

.IPT 5,INTRPT5

.STECT DATA,BASE,REL
TCYCLE: .DSW 1 ;total cycle time
.ENDSECT

.STECT SUBR,ROM 16,REL
INTRPT5: LD A,TCYCLE ;get total cycle time
SC
SUBC A,R4 ;subtract current
;counter
ST A,R4 ;to get alternate cycle
;time and store to
;auto-reload reg

RETI
.ENDSECT
.END PWMSTR

```

Listing 4 Use of T2 to Generate Non-50% Duty Cycle with Interrupts

```

.TITLE 'T2 NON-50% DC'
.SECT CODE,ROM16,REL
TMMODE = 0190:W
DIVBY = 018E:W
BFUN = 00F4:W
DIRB = 00F2:W
PORTB = 00E2:W
T2 = 0188:W
R2 = 0186:W
ENIR = 00D0:B
IRPD = 00D2:B
PWMSTR: LD SP,#STKS ;initialize stack pointer
LD TMMODE,#0x400 ;stop timer T2
NOP ;delay to provide 8 CK2 cycles
NOP ;to make sure timer is updated
LD TMMODE,#0xC00 ;clear T2
LD ENIR,#00 ;interrupt pending bit
LD IRPD,#00 ;disable interrupts
LD IRPD,#00 ;clear interrupt
;pending bits
SBIT 3,BFUN ;set pin 3 of port B
;as timer 2 output
SBIT 3,DIRB ;set output direction
;on port B pin 3
LD DIVBY,#0x200;set clock as CKI/16
;for T2
LD T2,#9 ;load T2 with counter
;value to obtain a 20 kHz PWM
;frequency using a 16 MHz crystal
;and CKI/16 input to the timer, the
;counter value should be 9
LD R2,#39 ;load auto-reload
;register with count
;of 80%
LD TCYCLE,#48 ;set total cycle time
;count -2
RBIT 3,PORTB ;initialize output pin
;value to 0
LD ENIR,#0x20 ;enable timer interrupt
RBIT 3,TMMODE ;start timer
STOP: JP STOP
.ENDSECT

.STKS: .SECT STACK,BASE
.DSW 10
.ENDSECT

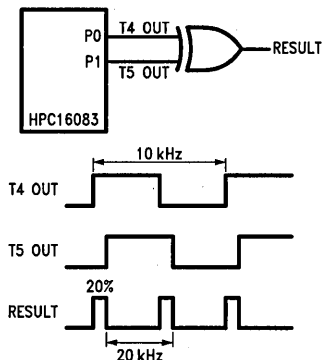
.IPT 5,INTRPT5

.TCYCLE: .SECT DATA,BASE,REL
.DSW 1 ;total cycle time
.ENDSECT

.INTRPT5: .SECT SUBR,ROM16,REL
LD A,TCYCLE ;get total cycle time
SC
SUBC A,R2 ;subtract current
;counter
ST A,R2 ;to get alternate cycle
;time and store to
;auto-reload reg
RETI
.ENDSECT
.END PWMSTR

```


Figure 6 shows the result of EXCLUSIVE-ORing the two timers. The duty cycle depends only on the phase shift between the timer outputs. In can be seen that the resulting frequency is actually twice the frequency of the original timers. Therefore, in order to generate a 20 kHz result, two 10 kHz timers must be used. The code is shown in listing 5. By varying the initial delay in the second timer, different duty cycles can be chosen. In the example given, a one digit difference in the counter value results in a 2% difference in the duty cycle.



TL/DD/10347-6

FIGURE 6. Using 2 Timers to Generate Non 50% Duty Cycle

Listing 5 Use of T4, T5 to Generate Non-50% Duty Cycle without Interrupts

```
.TITLE 'NON 50% PWM (T4,T5) '
.SECT CODE,ROM16,REL
TMMODE = 0190:W
DIVBY = 018E:W
T4 = 0140:W
R4 = 0142:W
T5 = 0144:W
R5 = 0146:W
PWMODE = 0150:W
PORTP = 0152:W
FREQ: .DW 49 ;counter value for the timers
;this generates 10 kHz PWM
DC: .DW 20 ;duty cycle = 20%
PWMSTR: LD SP,#STKS
LD PVMODE,#0X44 ;stop T4 and
;T5
NOP
NOP
LD PVMODE,#0XCC ;clear T4, T5
;int pending bits
LD T4,FREQ ;load T4, R4, R5
LD R4,FREQ ;with counter value
LD R5,FREQ
LD A,DC ;calculate delay for
MULT A,FREQ ;T5
DIV A,#100
ST A,T5 ;store in T5
LD PORTP,#0X10 ;set output pins, T4
;low T5 high
SBIT 3,PORTP ;enable toggling of
SBIT 7,PORTP ;pins on underflow
AND PVMODE,#0XFFBB ;start T4 and
;T5
STOP: JP STOP
.ENDSECT
.SECT STACK,BASE
STKS: .DSW 10
.ENDSECT
.END PWMSTR
```

Listing 6 Use of T2, T3 to Generate Non-50% Duty Cycle without Interrupts

```

.TITLE 'NON 50% PWM (T2,T3) '
.SECT CODE,ROM16,REL
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
TMMODE    =      0190:W
DIVBY     =      018E:W
T4        =      0140:W
R4        =      0142:W
T5        =      0144:W
R5        =      0146:W
PWMMODE   =      0150:W
PORTP     =      0152:W
T2        =      0188:W
R2        =      0186:W
R3        =      018A:W
FREQ:     .DW    49      ;counter value for the timers
                        ;this generates 10 kHz PWM
DC:       .DW    20      ;duty cycle = 20%
PWMSTR:   LD      SP,#STKS
          LD      TMMODE,#0x4400      ;stop T2,T3
          NOP
          NOP
          LD      TMMODE,#0xCCC8      ;clear T2, T3
                        ;int pending bits
          LD      PORTB,#0x10         ;set output pins, T2
                        ;low T3 high
          LD      DIRB,#0xFFFF       ;output on PORT B
          OR      BFUN,#0x0018       ;set T2,3 as timers
          OR      DIVBY,#0x2200      ;select CKI/16 clock
          LD      T2,FREQ             ;load T2 R2, R3 with
                        ;counter value

          LD      R2,FREQ
          LD      R3,FREQ
          LD      A,DC                 ;calculate delay for
          MULT   A,FREQ                ;T3
          DIV    A,#100
          ST     A,R3                 ;store delay in T3
          AND   TMMODE,#0xBBFF       ;start T2,T3
STOP:     JP     STOP
          .ENDSECT

          .SECT  STACK,BASE
STKS:    .DSW   10
          .ENDSECT
          .END   PWMSTR

```

CONCLUSION

PWM is easily generated by the HPC 16083 with its abundant source of timers. With a 30 MHz crystal, the maximum PWM frequency that can be achieved is 937.5 kHz. The timers run by themselves once the proper setup is performed. A method of obtaining non-50% duty cycle PWM without software intervention was presented.

C in Embedded Systems and the Microcontroller World

National Semiconductor
Application Note 587
David LaVerne



ABSTRACT

C is becoming the higher-level language of choice for microcontroller programming. Traditional usage of C depends on assembly language for the intimate interface to the hardware. A few extensions to ANSI C allow embedded systems to connect directly and simply, using a single language and avoiding detailed knowledge of the compiler and hardware connections.

HIGHER-LEVEL LANGUAGE USAGE

The desires leading to the greater use of higher-level languages in microcontrollers include increased programmer productivity, more reliable programs, and portability across hardware. Few such languages have served well when required to manipulate hardware intimately because most have been for mathematical computation. The C language has always been close to machine level. Indeed Kernighan and Ritchie^[1] refer to it as not really a higher-level language; one view of C is as a higher-level syntax expressing PDP-11 assembly language.

C has gained a great deal of its reputation and popularity associated with its use for operating systems, specifically UNIX[®][2] and similar systems. Many languages will do well enough for the application and utility programs of such a system, but being appropriate for the kernel indicates C can probably do the job of hardware control in an effective manner.

The needs of an embedded system, however, are not identical to the environment from which C has come. This warrants looking at C as it is and comparing it to the needs of C for the microcontroller world.

Operating Systems vs Embedded Systems

In most non-embedded programs, it is the processing which is important, and the Input/Output is only to get the data and report the results. In embedded or realtime applications, it is the Input/Output which is vital, and the processing serves only to connect inputs with outputs.

Operating systems are actually not as closely tied to the hardware as they might appear initially, and those portions which are close are not very portable. Operating systems manipulate hardware registers primarily for memory management (to map tasks), task process switching (to activate tasks), interrupt response (to field requests), and device drivers (to service requests). Because memory management hardware is so different between systems; because task process changing is so contingent on processor operations and compiler implementations; because interrupt system behavior is so varied; and because device control is so dependent on architecture and buses, these particular aspects of the operating system are not concerned with portability. As a result, they are generally kept separate, use a less convenient form of C depending on constants, and frequently are implemented in assembly language. This is not a major problem, since they comprise only a small portion of the total system, and have to change anyway each time the system is ported.

Embedded systems, by their very nature, are closely tied to the hardware throughout the system. The system consists of manipulating the hardware registers, with varying amounts of calculation and data transformations interspersed with the manipulations. As the system gets larger, the calculations may get more complex and may become a larger share of the program, but it is still the hardware operations which are the purpose of the system. Because the system in which these hardware pieces reside consists mostly of these hardware pieces, it is reasonable to hope for portability across processors or controllers for an application or product. Attempting to isolate all of the hardware operations is often impractical; using inconvenient forms of C is troublesome throughout the system and throughout its life-cycle; and implementing them in assembly language defeats the advantages of higher-level language usage and eliminates portability for those (and related) portions. For embedded systems, conveniently accessing hardware registers while doing calculations is essential.

Computer Systems vs Embedded Systems

Computational systems generally can be down the cable, and thus down the hall, from where they are used and can be whatever size is necessary to get the performance; production quantities are measured in hundreds and thousands, so price is a price/performance issue. Embedded systems end up tucked away in some of the strangest and tiniest places, so size can be a success or failure issue; quantities are often tens of thousands to millions of units, so additional chips or costs are multiplied ferociously and become a bottom-line issue.

The computer systems for which C was originally developed were relatively small and not especially sophisticated. However, as systems have grown, C and its implementation has grown right along with them. Most computer systems for which C is used now involve high-speed processors with large memory caches to huge memory spaces, backed by virtual memory. Many have large register sets. Such linear memory with heuristic accelerators allow for very large programs and fast execution. A major effort in optimization is in the allocation and usage of the registers, which tend to be general purpose and orthogonally accessible. Such systems, processor chips, and compilers compete almost exclusively in the field of speed.

Embedded systems, and most especially microcontrollers, have a different nature. While some applications may add external devices and memories to the controller, many are meant to be fully self-contained on one chip or have at most a few I/O chips. Microcontroller systems are small, are often required to fit in a physically small space, and are usually fed small amounts of power. Even when the system is externally expanded, the memories provided on-chip are significantly faster than the external memories because of buss driving. The total addressing space is usually very limited (32k, 64k) with expansion not linear. The registers in microcontrollers are usually a limited number of special pur-

pose registers, thus eliminating orthogonal usage. Speed is only one of many considerations in the microcontroller competition. Cost, package size, power consumption, memory size, number of timers, and I/O count are very important considerations.

Embedded Systems

Higher-level languages will achieve the goals of programmer productivity, program reliability, and application portability only if they fit the target environment well. If not, productivity will disappear into work-arounds and maintenance, reliability will be lost to kludges, and portability will not exist.

DESIRED TRAITS IN C FOR MICROCONTROLLERS

The environment in which C has developed is not the same as the embedded microcontroller world. What changes or extensions or implementations of C will provide the means to adapt the language? National Semiconductor Microcontroller Division has a compiler^[3] developed for the 16-bit High Performance Controller (HPCTM[4]) which has led to some exploration of these issues. The needs can be summarized as:

- Compatibility
- Direct Access to Hardware Addresses
- Direct Connection to Interrupts
- Optimization Considerations
- Development Environment
- Re-Entrancy

Compatibility

The first consideration for any such adaptation MUST be compatibility. Any attempt to create a different language, or another dialect of C, will create more problems than using C will solve. Dialects create problems in portability, maintenance, productivity, and possibly reliability. A programmer used to working in C will be tripped up by every little gotcha in a dialect; everyone will be tripped up by a different language.

Providing extensions to the language, while maintaining compatibility and not creating a new dialect, is accomplished by using the C Pre-Processor. By carefully choosing the extensions and their syntax, the use of the preprocessor's macro capability allows them to be discarded for normal C operation with non-extended compilers. By carefully choosing their semantics, the elimination of the extensions does not render the program invalid, just less effective.

Within these considerations there should be no unnecessary additions. An extension should not be made to avoid the optimizer's having to work hard. An extension should be made only to give the user an ability he would not have without it, or to tell the compiler something it cannot figure out by itself.

Direct Access to Hardware Addresses

Access to hardware addresses is improper in computation programs, is unusual in utility programs, is infrequent in operating systems, and is the *raison d'etre* of microcontrollers. The normal means of accessing hardware addresses in C is via constant pointers. This is adequate, if not great, when the accesses are minimal. For example

```

struct HDLC_registers
{
    ....
};
#define HDLC_1(*(struct HDLC_registers*)
    0x01a0)

```

allows reference to a structure of HDLC device registers at address 0x01a0, but never actually creates the entity of such a structure. If a debugger were asked about HDLC_1, it would not recognize the reference. If many registers and devices are involved, it becomes a problem to be handled by the programmer, not his tools. If the debugger tries to read the source for preprocessor statements, it adds significant complexity.

Another way of doing it is

```

struct HDLC_registers
{
    ....
};
extern struct HDLC_registers HDLC_1;

```

and providing an external file defining the address of HDLC_1, written in assembly language. This is clean, and does create the actual entity of a structure at the address, but has required an escape to assembly language for the system (although only at the system definition level). This was the first choice at National, and retains merit because the use of macros in the definition file allows the simple creation of a table exactly like the table in the hardware manual.

What is desirable, so that the user can do his own definitions without resorting to two languages, is a means to create the entities and define the addresses of those entities, a simple means of saying that this variable (or constant) is at a specific absolute address. The syntax

```

struct HDLC_registers HDLC_1 @ 0x01a0;

```

would be excellent as an official enhancement to the language, since the @ parses like the = for an initialization (and the program shouldn't initialize a hardware register this way like a variable). However, this violates the compatibility rule for an extension, since the preprocessor cannot throw away the address following the @ character. Therefore,

```

struct HDLC_registers HDLC_1 At (0x01a0);

```

is a much more practical form as an extension—and can be made to expand to the previous (or any other) form if it is ever added as an enhancement to the language. The resulting forms

```

volatile struct HDLC_registers HDLC_1 At
    (0x01a0);
volatile struct HDLC_registers HDLC_2 At
    (0x01b0);
volatile const int Input_Capture_3 At
    (0x0182);

```

are straightforward, simple, readable, and intuitively understandable, and provide the data item definitions as desired.

Direct Connection to Interrupts

Operating systems attach to interrupts in one centralized, controlled location and manage them all in that module. Embedded systems attach to varied interrupts for a variety of purposes, and frequently the different interrupt routines are in different modules with associated routines for each purpose. It is possible to do this with another escape to assembly language, but this requires that the system be maintained and enhanced in two languages.

The solution chosen for the National compiler is to provide an identifier for functions which are to service interrupts.

These functions obviously take no arguments and return no values, so they are worth considering as special. The syntax chosen was simply

```
INTERRUPT2 timer_interrupt ( )
```

although a more desirable form as an official enhancement would be

```
INTERRUPT(type)
interrupt_service_routine( )
```

because the chosen syntax can be preprocessed into whatever might be the final form. The semantics of the interrupt function were more difficult to guarantee for the future—should an interrupt function be callable by the other functions? Prohibiting it allows eventually permitting it if necessary; for improved efficiency, the National compiler does not allow an interrupt function to serve as anything other than an interrupt service routine, although one function can be attached to several interrupts.

Because the functions are special purpose, the function entry and exit code can be dedicated to interrupt entry and exit, rather than having to hide it in a separate library module. The National compiler actually generates the interrupt vector to point directly to the interrupt function; the function saves and restores the registers which it may destroy. Latency is minimized.

Interrupt response speed (latency) and interrupt system performance are important characteristics of a microcontroller. It is one thing (inconvenient or embarrassing) for a multi-MIPS machine to choke on long 9600 baud transmissions and drop a character or two because of inefficient interrupt response. It is another thing entirely—lethal, a total failure—for an embedded system's interrupt response to be so poor as to miss even one critical interrupt.

Optimization Considerations

Computer systems compete on speed (or at least MIPS ratings); compilers for them must be speed demons. Microcontrollers compete on size and costs; compilers for them must be frugal. Embedded systems are limited in their memory and different memories frequently have significantly different behavior.

The major concern of optimization comes down to code size. In most controller systems, as generated code size decreases speed usually increases. The effort in the code generation and optimization should be directed towards reducing code size. Claims for exactly how close the generated code gets to hand-written assembly code depend on specific benchmarks and coding techniques. An acceptance criterion for the National HPC compiler was code size comparison on a set of test programs. A level slightly below 1.4 times larger than assembly was reached.

In addition to the implementation of the optimization, other concerns of microcontrollers affect the way code can be generated. An example is the different forms of memory. Many controllers have memories which can be accessed by faster or shorter code. Certain variables should be placed in these memories without all the variables of a module going there (which is a linker process). There is no possible way for the optimizer to guess which variables should go there,

especially in a multiple module program, so it must be told. The syntax used is

```
static BASEPAGE int important_variable;
```

because the special memory in National's HPC is the first page of RAM memory. Several other possibilities offer themselves, including using for an official enhancement

```
static register int important_variable;
```

because currently static register variables are specifically prohibited. This cannot be an extension, because the register word could not be redefined to the preprocessor. If some variables need to be accessed by fast code, and some need to be accessed by short code, and if the two were mutually exclusive, it would be desirable to have two separate extension words. Since such hardware is unlikely, the single word BASEPAGE is probably sufficient.

Additional savings can be achieved by reconsidering string literals. The ANSI C requires that each string literal is a separate variable, but in actual usage they are usually constants and therefore need not be separate nor variables. The National compiler provides an invocation line switch to indicate that all string literals (but not string variables) can be kept in ROM rather than being copied to RAM on system start-up. Such strings can be merged in the ROM space to eliminate duplication of strings.

An extension to the language to identify functions which will not be used recursively is

```
NOLOCAL straight_forward_function( );
```

which causes all local variables to be converted to static variables, which are easier and faster to access and use. If the function has no arguments, the compiler can even eliminate the use and creation of the Frame Pointer for the function, saving additional code and time.

The particular processor, the HPC, has a special form of subroutine call. Since the optimizer cannot guess across modules which functions should be called with the special form, the extension

```
ACTIVE specially_called_function(arg);
```

was added. This may or may not be appropriate for other processors, but is a good example of why the language needs careful extensions to take advantage of different processors.

One command extension was added to the language because it allows the programmer to guarantee something the optimizer cannot usually determine. The form

```
switchf(value) {...}
```

provides for a switch/case statement without a default case. When speed and size become critical, the extra code required to validate the control value and process the default is highly undesirable when the user's code has already guaranteed a good value.

The National compiler has one extension which violates the issues stated under compatibility. It remains for historical reasons. It is a command

```
loop(number) {...};
```

which produces a shortened form of the for loop, without an accessible index. This does not provide the user with any new ability, it merely allows the compiler optimizer to know, without figuring out, that the index is not used inside nor outside the loop, and can therefore be a special counted form. The preprocessor cannot produce an exact semantic equivalent for the statement. This is a perfect example of a poor extension and will eventually be eliminated.

Development Environment

Languages developed for large or expensive systems can usually depend on large systems for development support, either self-hosted or with a large system host providing cross-development tools. Microcontrollers are often price sensitive, are frequently in the laboratory or the field, and are not always supported by a large system as a development host. Personal computers provide an excellent platform for the entire suite of development tools.

National Semiconductor currently provides its compiler and associated cross-development programs on the IBM PC and clone type of computer. The software is all very portable, and can be run under VAX/VMS, VAX/Ultix, or VAX/BSD4.2, and on the NSC 32000-based Opus add-in board for the PC running UNIX V.3, and some other versions of UNIX. The demand has been for the PC version; the PC is a very good workstation environment for microcontrollers. Other environments may be desirable, but the PC is first.

Re-Entrancy

Even with all these other considerations handled, there is a time bomb lurking in C on microcontrollers. C is a single thread, synchronous language as it is usually implemented. Since most utilities are strictly single-thread and the UNIX kernel forces itself into a single-thread, this is not a big problem for them. Embedded systems involving controllers are inherently asynchronous; the language in which they are implemented must be multi-thread without special rules and exception cases.

The passing of arguments on the stack and the returning of values in registers allow for complete re-entrancy and thus asynchronous multi-threading, but this breaks down when structures are returned. Most implementations of C use a static structure to contain the returned value and actually return a pointer to it; the compiler generates the code to access the returned structure value as required. This cannot

be used in a microcontroller environment, because if an interrupt occurs during the time the static structure is being used, it cannot re-enter the function. On an operating system level such conflicts can be managed with gates, semaphores, flags, or the like, but that solution is completely inappropriate on the language level. Turning the interrupts off is similarly not a language level concept, and is impossible on a system with a NonMaskable Interrupt. Telling users not to get themselves into that situation is crippling at best, impossible to enforce, and extremely difficult to track down and correct.

The solution should be at the language level, and should allow the return of a structure without hindering re-entrancy. The author's solution, developed with National, has been to have the code calling the function provide the address of a structure in which to build the return value. Since this is frequently on the caller's stack, and is never invisibly static, the program has no hidden re-entrancy flaws.

The HPC C Compiler

The HPC C Compiler (CCHPC) is a full and complete implementation of ANSI Draft Standard C (Feb 1986) for free-standing environment. Certain additions take advantage of special features of the HPC (for the specific needs of microcontrollers). The extensions include the support of two non-standard statement types (loop and switch), non-standard storage class modifiers and the ability to include assembly code in-line. The compiler supports enumerated types, passing of structures by value, functions returning structures, function prototyping and argument checking.

Symbol Names, both internal and external, are 32 characters. Numerics are 16-bit for **short** or **int**, 32-bit for **long**, and 8-bit for **char**, all as either **signed** or **unsigned**; floating point are offered as **float** or **double**, both using 32-bit IEEE format.

All data types, storage classes and modifiers are supported. All operators are supported, and anachronisms have been eliminated (as per the standard). Structure assignment, structure arguments, and structure functions are supported. Forward reference functions and argument type checking are supported.

Assembly code may be embedded within C programs between special delimiters.

See Table I.

CCHPC SPECIFICATIONS

TABLE I

Note: Extensions are boldface

Name length	32 letters, 2 cases
Numbers	
Integer, Signed and Unsigned	16-32 Bits
Short and Long	16 bits and 32 bits
Floating, Single and Double	32 bits and 32 bits
Data Types	
Arrays	
Strings	
Pointers	
Structures	
Preprocessor	
#include	
#define #define() #undef	
#if #ifdef #ifndef #if defined #else #elif #endif	
Declarations	
auto register const volatile BASEPAGE	
static static global static function NOLOCAL INTERRUPTn ACTIVE	
extern extern global extern function	
char short int long signed unsigned float double void	
struct union bit field enum	
pointer to array of function returning	
type cast typedef initialization	
Statements	
;{...} expression; assignment; structure assignments;	
while (...)...; do...while (); for(;;)...; loop()...;	
if (...) else...; switch ()...; case:...; default:...; switch ()...;	
return; break; continue; goto...; ...:	
Operators	
primary:	function() array[] struct_union. struct_pointer ->
unary	* & + - ! ~ ++ -- sizeof (typecast)
arithmetic:	* / % + - << >>
relational:	< > <= >= == !=
boolean:	& ^ &&
assignment:	= += -= *= /= %= >>= <<= &= ^= =
misc.:	?: ,
Functions	
arguments:	Numbers, Pointers, Structures
return values:	Numbers, Pointers, Structures
forward reference (argument checking)	

Library Definition **Limited-Freestanding environment**

Embedded Assembly Code

CONCLUSIONS

With the right extensions, the right implementations, and the right development environment, National is providing its customers with a C compiler tool which allows effective higher-level language work within the restrictive requirements of embedded microcontrollers. Productivity increases do not have to come at the expense of larger programs and more memory chips. No strangeness has been added to the language to cause reliability problems. Portability has been retained. Assembly language code has been eliminated as the chewing gum and baling wire trying to hold it all together, further increasing reliability and portability.

FOOTNOTES

1. Kernighan, Brian W. and Ritchie, Dennis M., "*The C Programming Language*", Prentice-Hall 1978, Pages ix and 1.
2. UNIX® is a registered trademark of AT&T.
3. Produced by Bit Slice Software, Waterloo, Ontario, Canada.

ADDITIONAL INFORMATION**Datasheet**

HPC Software Support Package

User's Manual

HPC C Compiler Users Manual #424410883-001

HPC16400

A Communication Microcontroller with HDLC Support

National Semiconductor
Application Note 593
Nick Burd



INTRODUCTION

The HPC16400 is a communications microcontroller for HDLC based applications and is the latest in the range of High Performance microcontrollers (HPC™) from National Semiconductor Corporation. HPC is a family of 16-bit CMOS microcontrollers which feature a common core to which are added peripherals for a specific application area. In the case of the HPC16400, these include dual HDLC channels and a four channel DMA controller which make the HPC16400 ideally suited to embedded protocol processing, such as X.25/LAPB. In addition, the HPC16400 also contains an on-chip serial decoder which allows the HDLC channels to be time multiplexed onto common transmit and receive lines as used by the ISDN (Integrated Services Digital Network) Basic Rate interface. This means that together with Nationals' ISDN line interface and COMBO™ circuits, and a software

package which implements the generic ISDN protocols (Q.921 and Q.931) a complete system solution for ISDN Basic Rate applications is possible.

The HPC16400 is capable of running at a maximum clock frequency of 20 MHz, and each of its HDLC channels can operate up to a maximum 4.65 Mbps data rate. A photograph of the HPC16400 chip is shown in *Figure 1*.

This article describes the features of the HPC16400, and in particular the operation of the HDLC/DMA channels and the serial decoder. As an example of how the HPC16400 would be used in an ISDN application, an ISDN terminal is described together with the features of the ISDN software package which can be used to minimize the time and effort in developing such equipment.

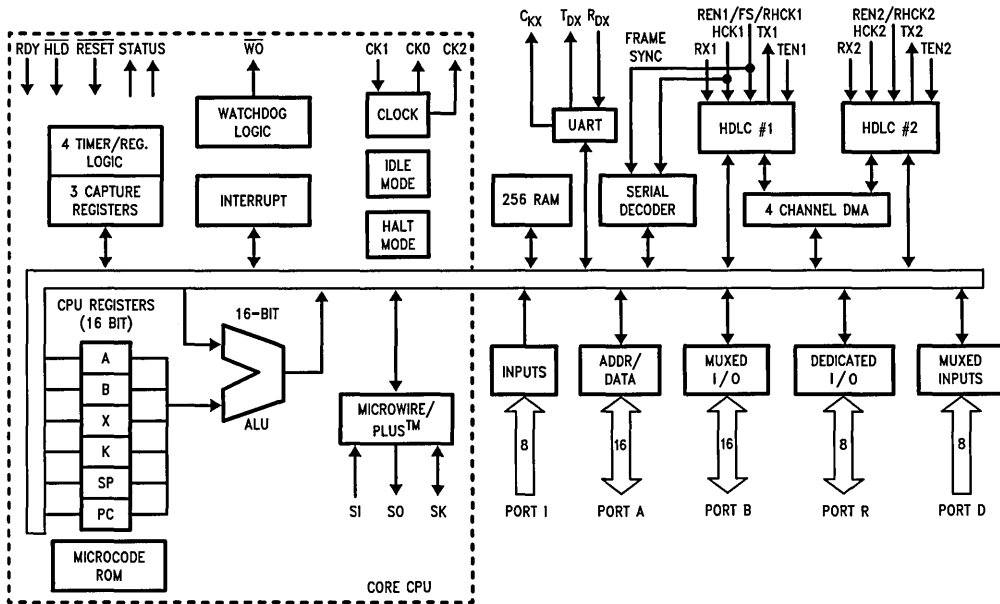


FIGURE 1. Block Diagram of the HPC16400

TL/DD/10361-2

THE HPC CORE

Figure 1 shows the block diagram of the HPC16400 in which the functions within the dotted line form the HPC core which is common to all HPC family members. It can be seen that the core contains the CPU as well as several peripherals. Those functions outside the dotted line are the peripherals specific to the HPC16400.

The CPU contains a 16-bit ALU and a 16-bit accumulator which acts as the source and destination for most operations. Two 16-bit address pointer registers, B and X, are intended to be used for indirect addressing of data with auto increment and decrement of the register. The K register is used to set a limit for the B register when it is either incremented or decremented with successive execution within program loops. A specific feature of the instruction set of the HPC CPU is that conditional execution of an instruction is based on a skip structure instead of the traditional conditional branch or jump. This is best illustrated through an example using the B, K and X registers described above. The example listed in Figure 2 swaps the contents of two areas of memory in the ranges 0x4000 to 0x4FFF and 0x5000 to 0x5FFF. A single instruction is used to load the B and K registers which define the boundaries of the lower memory area, and the X register is loaded to point to the

beginning of the upper memory area. The first instruction within the loop loads the accumulator with the memory word pointed to by the X register, and the X register is then incremented. The fact that a word value has been specified here means that the X register will automatically be incremented by two. If a byte value had been specified, it would be incremented by one. The second instruction in the loop is an exchange with a conditional skip which exchanges the contents of the 16-bit accumulator with the memory word pointed to by the B register, and the B register is then incremented by two. If the new value of the B register now exceeds the value in the K register, the following jump instruction will be skipped and program execution will exit the loop. If the value of the B register is less than the K register, then the next instruction is executed and the loop is continued. Judicious encoding of the opcodes for the HPC instruction set has resulted in a very efficient implementation of common constructs such as the loop just described. The register indirect instructions are encoded as single-byte instructions as well as the short jump instruction where a six bit offset is included within the opcode. The loop described above therefore generates only three bytes of program code. In total, the HPC has 54 instructions and nine addressing modes.

```

LD BK, #4000, #4FEE ;load B and K with start and end of 1st. memory block.
LD X, #5000 ;load X with start of second memory block.
LOOP: LD A, [X+].W ;get word from second block, increment X.
XS A, [B+].W ;exchange with word from first block, increment pointer,
;skip if B>K.
JP LOOP ;do loop again

EXIT: ↓ Continue program

```

FIGURE 2. An Example HPC Program to Swap Two Memory Areas

The HPC core contains several peripheral features. The MICROWIRE/PLUS™ is an inter-chip serial communication port which consists of an 8-bit shift register and a clock. Writing data to the microwire port when configured as a master causes the data to be loaded into the shift register and eight clock pulses generated to shift the data out. At the same time, these clock pulses can be used to clock data in from a microwire slave device such as the ADC0834 A/D converter or the NMC93C46 EEPROM.

The HPC core also contains a number of timers. A purpose of one of these timers, T0, is to provide a means for accurate time interval measurements, and when configured in this mode, it is associated with up to three capture registers which can be triggered by external interrupt inputs. Timer T1 provides a dual function as it can operate as a normal timer, or its registers can be used as two of the capture registers for T0. The timer T0 also drives the Watchdog™ logic which causes the Watchdog output to trigger whenever it is not serviced before a timeout of T0. The remaining two timers can be used to generate a variety of timing outputs.

Interrupt logic provides enabling circuitry for the numerous sources of interrupt on the HPC, and an interrupt pending register eases the processing of multiple interrupts. The HPC can be placed into one of two power saving modes by programming the Processor Status Word (PSW) register and the Halt Enable register. In the Halt mode, all processor activities, including the clock and timers, are stopped thereby reducing the power requirements of the HPC to a minimum. Recovery from the Halt Mode can either be from a Reset or from the NMI. In Idle mode, all processor activity apart from the on-board oscillator and timer T0 is stopped so that recovery from the Idle mode can be achieved with the timer T0 overflow as well as the reset or NMI functions as in the Halt mode (except that in the Halt mode recovery is not immediate as the oscillator will take time to stabilize).

HPC MEMORY

All functions on the HPC chip are memory mapped. The on-chip peripherals, core registers, and on-chip user RAM (16-bit) occupy an address area between 0 and 0x1FF as shown in the memory map of *Figure 3a*. The area of user on-chip RAM in the range 0–0xBF is in the BASEPAGE (0–0xFF) of the address space, and in addition to being used as general purpose storage locations for variables, the indirect addressing mode of the HPC allows memory words in this area to be used as pointers containing the effective address of the operand. This allows many additional pointers to be created in addition to the B and X register and significantly eases the programming of many tasks.

The memory requirements in telecom applications are generally large for both program and data areas, and so the HPC16400 does not have a single-chip configuration with

on-chip ROM. Instead, a 16-bit multiplexed address and data bus is brought external to the chip and is used to add program memory and additional data memory to the system in the address range 0x200 to 0xFFFF.

FFFF	External User Memory
0200	User RAM
01c0	HDLC 2 Registers
01b0	HDLC 1 Registers
01a0	TIMER and WATCHDOG Registers
017e	DMA Tx2 Registers
0170	DMA Rx2 Registers
0160	DMA Tx1 Registers
0150	DMA Rx1 Registers
0140	UART Registers
0120	PORTR and PORTD Registers
0100	PORTB Registers
00e0	MICROWIRE™, PORT Control and INTERRUPT Control Registers
00d0	HPC CORE Registers
00c0	On-Chip RAM
0000	

FIGURE 3a. The Basic 64k Memory Map of the HPC16400

The 64 kbyte address space can be expanded further by the use of bank switching. Four lines from Port B may be used to select one of sixteen banks of 32 kbytes in the address range 0–0x7FFF as shown in *Figure 3b*. In this way, the upper 32 kbytes of memory are common to all of the banks and allows a program in one bank to jump or call subroutines in other banks via this common area where the banks can be safely switched (see reference 1 for a more in-depth discussion of bank switching on the HPC). The common memory also provides storage area for global variables and stack locations when operating in a bank-switched environment. The total memory addressing capability of the HPC16400 amounts to just over 500 kbytes as the section of on-chip RAM in the range 0–0x1FF is common to all banks.

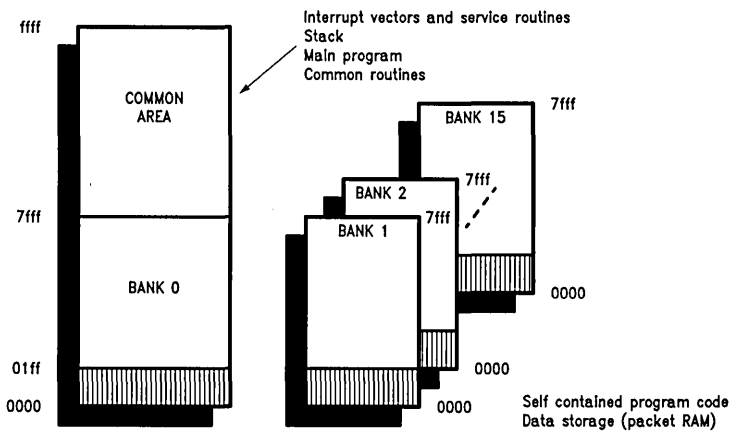


FIGURE 3b. HPC Extended Memory Addressing by Bank Switching

TL/DD/10361-3

When the HPC fetches program from memory, it does so one byte at a time because the opcode encoding is byte oriented. This allows the HPC to be configured with either 16-bit external memory, 8-bit external memory for more cost sensitive applications, or a mixture of both. When operating with 16-bit memory, the HPC can access both odd and even bytes, and words on an even boundary. In 8-bit mode the HPC makes only byte accesses to external memory, and although the registers in the BASEPAGE memory of the HPC are 16 bits, they may also be addressed individually as high and low bytes thereby enabling the 16-bit architecture of the CPU to be used.

Selection of 8- or 16-bit bus mode for the HPC is achieved on reset of the processor when the "high byte enable" control line is sampled by the CPU. If this line is detected in a high state, the HPC enters 8-bit mode. However, if the line is detected as high impedance, as a result of it being used as a control output to select low and high 8-bit memory banks, then the HPC enters 16-bit bus mode.

THE HDLC AND DMA CHANNELS

The HPC16400 contains two identical on-chip HDLC channels, each capable of transmitting and receiving HDLC frames transparently to the operation of the CPU. The format of an HDLC frame is shown in Figure 4. The frame is delimited by an identical opening and closing flag which is a

unique bit pattern consisting of a zero followed by six consecutive ones and then a final zero. This pattern must not occur anywhere else within the frame and is guaranteed by a zero insertion mechanism which, after the transmission of five consecutive ones in the data stream between flags, will insert a zero before continuing to transmit data. A reverse procedure is adopted at the receiver to delete the additional zeroes. Immediately following the opening flag is an address field which identifies which equipment on the network is to receive the frame. The control field contains information, such as handshake control, which is used to control the flow of frames between communicating devices. This is followed by the application specific data, a frame check sequence which validates the integrity of the frame with a cyclic redundancy check (CRC) code, and the closing flag. The HPC HDLC channels provide automatic framing functions such as opening and closing flag insertion and deletion, zero bit insertion and deletion (also known as bit-stuffing), CRC16 or CCITT implementations of CRC checking, and abort sequence transmission and recognition. The abort sequence in this case is a modified flag consisting of a zero followed by seven ones. In addition, the transmitters can be programmed to generate flags, abort sequences, or just idle (transmitting consecutive ones) between the transmission of consecutive frames.

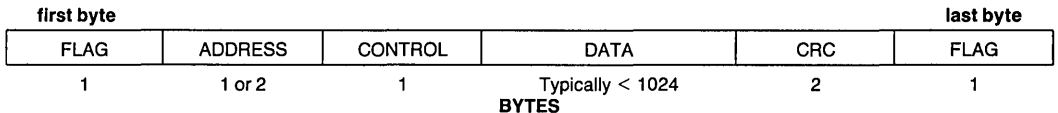
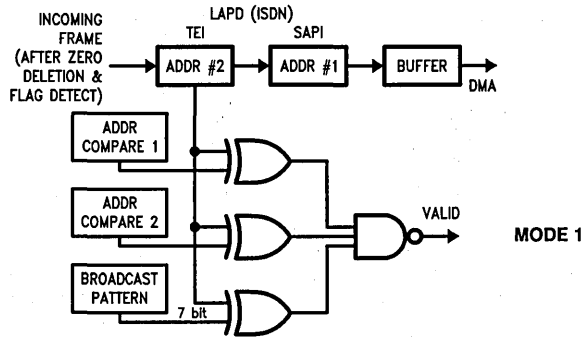


FIGURE 4. The HDLC Frame Format

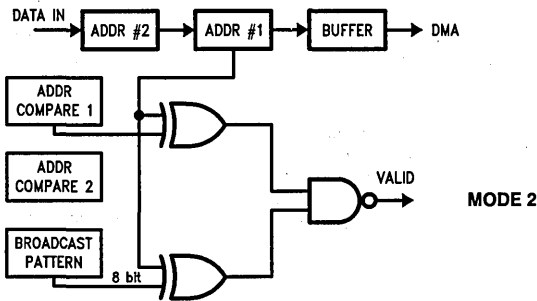
A feature which helps to reduce the CPU overhead in protocol processing is the address recognition logic. Each channel has two address recognition registers that can be programmed with a byte which can be compared in a number of different ways with the first two bytes received by an HDLC channel. The different comparison modes are intended to cope with a range of different communication network addressing modes. *Figure 5* shows the logical operation of three of the four possible modes. In mode one, the second byte received after the opening flag of the frame is com-

pared with both address registers and a seven bit broadcast address pattern (0x7F). If any of the registers match the incoming address, then the HDLC channel will continue to receive the complete frame. If no match is detected, the HDLC channel will stop receiving the frame, discard the address already received, and start to look for the opening flag of the next frame. This particular address recognition mode is useful in ISDN communications because the second byte received will be the address of the terminal equipment, such as a telephone or perhaps a PC, on the ISDN network.



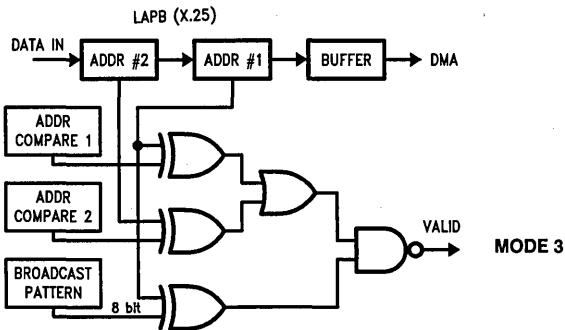
MODE 1

TL/DD/10361-4



MODE 2

TL/DD/10361-5



MODE 3

TL/DD/10361-6

FIGURE 5. The Address Recognition Logic for the HDLC Receivers

Mode two matches the first byte received with the first address register and an 8-bit broadcast address which could be used in X.25/LAPB applications. Mode three compares a 16-bit address field so that the contents of the first comparison register must match the first address byte received, and the contents of the second comparison register must match the second address byte received. Or, if the first byte corresponds to the 8-bit broadcast pattern, an address match will also be signalled to the CPU. The last mode, Mode zero, is the "transparent mode" in which all frames are received by the HDLC controller regardless of the address field contents. This mode would be used, for example, in a device which had to gather all information from the communications network and compute statistics about its communications loading.

Both HDLC channels are capable of implementing bit oriented protocols, such as IBMs SDLC, by programming the number of bits to be transmitted in the last byte of the information field. Further flexibility is achieved with a bypass mode which disables all of the HDLC framing functions allowing designers to implement their own byte-oriented synchronous protocols.

As mentioned earlier, all programmable features of the HPC are memory mapped, so the HDLC registers are mapped to an area of on-chip RAM above the BASEPAGE section in the range 0x1A0 and 0x1B8. Each HDLC channel has an identical set of registers, and each set contains receiver status, control, address comparison, and error status registers. In addition, there are two global registers which handle the enabling and servicing of interrupts from the HDLC channels. An interrupt can be generated whenever an HDLC channel signals an "End of Message" (EOM) which indicates that an HDLC frame has just been received or an HDLC frame has just finished being transmitted. Should a transmitter or receiver generate an EOM before the previous EOM has been serviced, then an overrun interrupt may be generated. All of these interrupt sources have a single interrupt service vector, and so the global registers contain bits which allow the source of the interrupt to be uniquely identified. Additional error conditions, such as reception of a bad CRC, reception of an abort sequence, or a framing error, cause bits to be set in the error status register which

may also generate an interrupt, although this may lead to the generation of multiple interrupts. A more straightforward approach would be to test the condition of the error status register once an EOM interrupt has been received.

The HPC16400 contains an on-chip four channel DMA controller. The operation of the DMA controller is closely linked to the HDLC channels because they are responsible for interfacing them to the memory. Hence, as each byte is received by an HDLC channel, it signals the DMA controller which requests and gains control of the processor bus and writes the received byte to a predetermined area of memory. Similarly, when an HDLC channel is transmitting a frame, it requests data from the DMA controller which transfers a byte from an area in memory to the HDLC channel. During DMA accesses the CPU loses control of the memory bus. However, for the HPC16400 running at 20 MHz, the CPU bus occupancy is only expected to decrease by 10% for an aggregate HDLC data rate of 2 Mbps. For typical Basic Rate ISDN applications the decrease is expected to be less than 2%.

The DMA channels contain several addressing features which allow convenient transmit and receive buffers to be created in memory. Each DMA channel supports a split-frame mode which allows the transmitted or received frame to be split into two sections with each section being stored in a different area of memory. In HDLC, it may be convenient to have all the address and control fields in one area of memory, and all the information fields in another. (The CRC and flag fields are stripped off or appended by the HDLC channels, and so are not present in the memory area.) In the DMA receiver, there are two pairs of address pointers, each pair pointing to the two sections of the same frame as shown in *Figure 6*. As the HDLC controller starts to receive data, the DMA channel places the first received byte in the memory pointed to by the first address pointer, and the pointer is then incremented. This continues until the number of bytes for the first segment, which can be programmed up to a maximum of 7 bytes in the DMA receiver control-status register, has been reached, at which point the contents of the second address pointer becomes the destination for the remainder of the received frame.

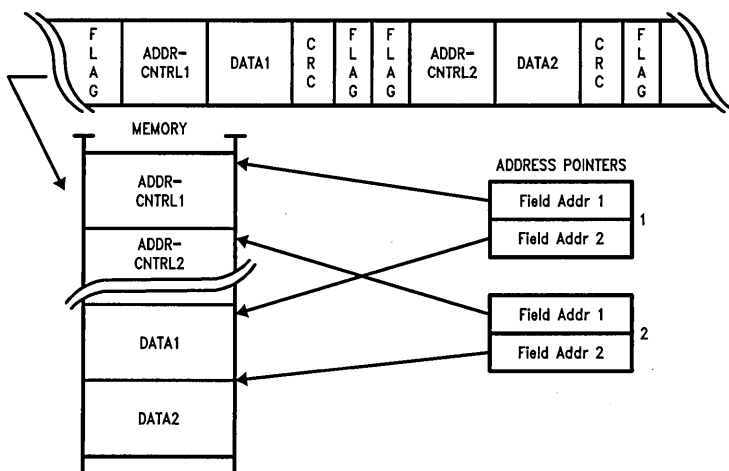


FIGURE 6. Split-Frame Operation for HDLC/DMA Receiver

TL/DD/10361-7

For the DMA channel which supports the HDLC transmitter, each pair of registers contains a single pointer and a byte counter which holds the number of bytes to be transmitted, as illustrated in Figure 7. When the split-frame mode is not used, each pair of registers in the transmit DMA, and each address pointer in the receive DMA, refers to a separate complete frame. This means that the HDLC receiver can receive four frames before the DMA address pointer registers need to be updated, provided the EOM is serviced after each frame to prevent an overrun interrupt.

In the previous section, the extended memory configuration of the HPC16400 using bankswitching was described. The DMA channels are capable of taking full advantage of this extended memory by a programmable field in the control-status registers whose value is written to the external bank-switch control lines during a DMA cycle. This allows the extended memory banks to be used for storing frame information.

The DMA controller is only capable of taking control of the processor bus when the CPU has finished executing the current instruction. When the HPC16400 executes long instructions, such as the Multiply or Divide instructions, and the HDLC channels are being used at very high data rates (in excess of 2.2 Mbps with a 20 MHz HPC), it may be possible that the DMA cannot gain control of the processor bus in time to service the HDLC channels. In this situation, the receiver is forced to overwrite the last byte received and a receiver overrun is flagged in the error status register. When this occurs during transmission, the transmitter no longer has any valid information to send and so it transmits an abort character and sets a transmitter underrun bit in the error status register. Programming the HDLC/DMA controllers is relatively straightforward, both for their initialization and interrupt servicing. Because the DMA controllers have two sets of registers, it means that the pointers to the next message to be received or transmitted can be set up while reception or transmission is in progress, thereby maximizing the throughput of the HDLC channels.

THE SERIAL DECODER—BASIC RATE ISDN AS AN EXAMPLE

As already described, the HDLC channels of the HPC16400 can be used in general purpose communications and networking applications. To enhance their capabilities, and provide on-chip support for ISDN, a serial decoder has been implemented to time division multiplex the two HDLC channels onto common transmit and receive lines.

Each HDLC channel can be enabled and disabled both internally by the serial decoder, and externally by individual receiver and transmitter enable pins. The internal enable signals are generated by the serial decoder according to six time division multiplexing (TDM) formats. The framing of these TDM formats, or modes, is synchronized by an externally generated frame sync. pulse which will normally be derived from an external clock signal used to clock the HDLC channels. With these inputs, the serial decoder generates the internal enable signals for the HDLC channels at the correct time within the frame according to mode that has been selected. The serial decoder can also be programmed to generate enable signals for the HDLC channels based on combinations of both the external enable signals and those generated internally by the serial decoder, thereby giving the designer a wide choice of possibilities.

As an example of the use of the serial decoder, we shall look at Basic Rate ISDN. Basic Rate ISDN specifies that a terminal equipment, such as a telephone or computer, should have two general purpose B channels (Bearer channels) for voice data or perhaps computer packet switched data, and a D channel which is used specifically for control of the ISDN network, such as setting up a call to another user. These 2B + D channels are time division multiplexed within a 125 μ s frame on a bus which interconnects functional blocks within a piece of equipment. The time slot for each B channel is the transmission time for 8 bits at a data rate of 64 kbps, and the D channel time slot is 2 bits at 16 kbps.

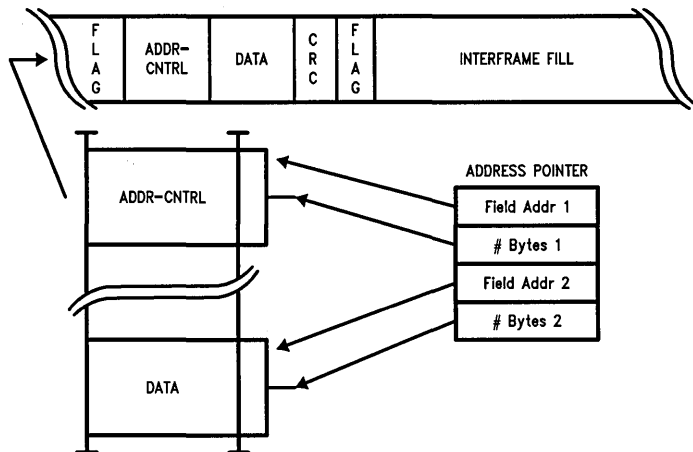


FIGURE 7. Split Frame Operation for HDLC/DMA Transmitter

TL/DD/10361-8

The overall scheme is shown in *Figure 8*. Now the HPC16400, having two HDLC channels, could be set up so that one HDLC channel is a B channel, and the other HDLC channel is the D channel. The serial decoder therefore has to be programmed so that its mode corresponds to the format shown in *Figure 7*, and that the enable signals are chosen internally such that the D time slot is assigned to one of the HDLC channels, and the correct B channel is assigned to the other HDLC channel. The remaining B channel could be occupied by any other device capable of generating a 64 kbps data stream within its time slot, such as a voice COMBO. The frame sync. signal and the HDLC clock will be generated externally to the HPC16400, typically by the ISDN line interface circuit as described in the next section.

AN ISDN TELEPHONE

Figure 9 shows the block diagram of an ISDN telephone. The three main components of the system are the HPC16400 microcontroller, the TP3420 "S" Interface Device (SID) which is the line interface to the ISDN subscriber

(S) link, and the TP3057 COMBO which provides the interface to the system for a handset. The inter-chip data bus, whose timing format was used as an example in the previous section, is called the Digital System Interface (DSI) bus, and combines the B and the D channels into common transmit and receive lines. Hence, the HDLC Tx outputs are tied together with the Dx output of the COMBO and are input to the SID DSI input pin Bx, and the HDLC Rx pins are combined with the Dr input from the COMBO and are driven by the SID DSI output pin Br. The SID, when configured in master mode, generates the frame sync. and clock signals which are derived from the received signal on the S bus. These signals are both connected to the HPC16400 and the COMBO so that the correct multiplexing format for the DSI bus as shown in *Figure 9* can be achieved. An additional output from the SID, DENx, indicates the presence of D channel bits on the DSI bus, and is used to enable the HDLC channel of the HPC16400 which has been assigned to handle the D channel communications, in this case HDLC channel 1.

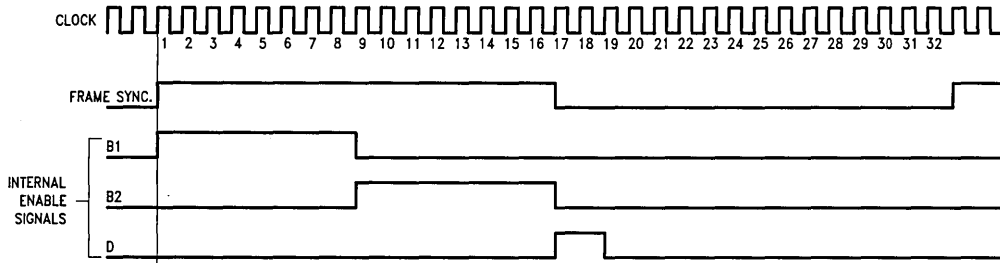


FIGURE 8. The Serial Decoder Format for ISDN

TL/DD/10361-9

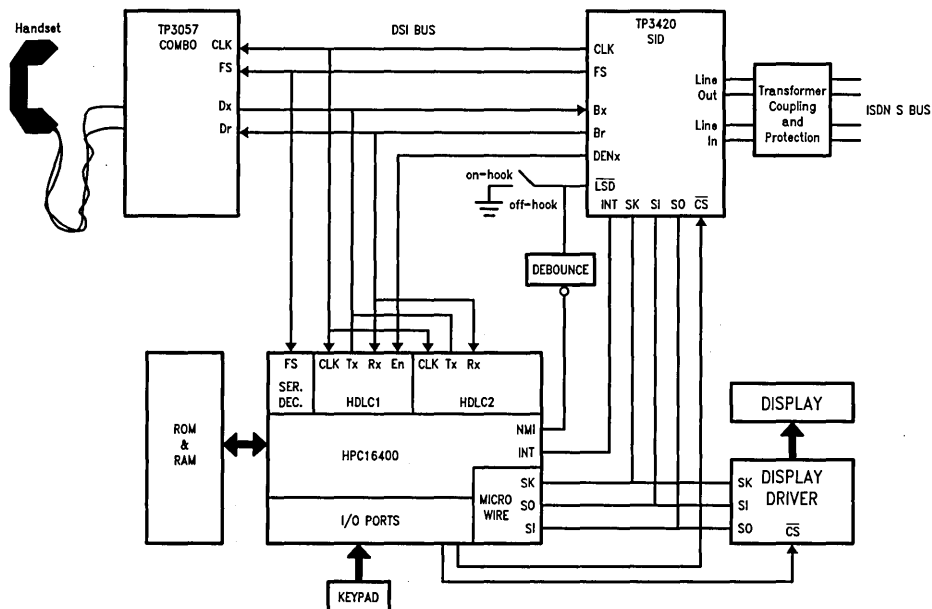


FIGURE 9. Block Diagram of an ISDN Telephone

TL/DD/10361-10

The SID is a programmable device with various modes and functions that conform to the CCITT I.430 specification for the physical layer of ISDN. Programming of the SID is achieved with the MICROWIRE/PLUS interface which is also used to drive the display for the telephone using a COP472-3 liquid crystal display controller. Selection of either the SID or the display driver is achieved with port lines from one of the general purpose I/O ports on the HPC16400 so that the chip selects for each device are software driven.

The Halt power saving mode can be used whenever the telephone is not active. This is indicated by the on/off hook signal from the handset which is interfaced to the NMI input of the HPC. When a telephone conversation is finished and the handset placed on-hook, the HPC can be put into Halt mode by software. When the handset is subsequently picked up for another call, and so goes off-hook, it will generate an NMI which will wakeup the HPC.

THE ISDN SOFTWARE

The control of end-to-end communications in a telephone system can be a complex procedure. Many things have to be taken into consideration, such as procedures for establishing a call, dial-plans, disconnecting calls, and so on. All of these procedures amount to the protocols which are part of ISDN. In particular, the control protocols for ISDN are those which are used on the D channel to establish and disconnect physical links between two (or more) users of the telephone network. *Figure 10* shows the three protocol layers of ISDN according to the ISO seven layer reference model for Open Systems Interconnection.

At the physical layer, the CCITT standard I.430 is used to specify the requirements of the ISDN S-Bus interface device. The TP3420 SID conforms to this specification, and in fact exceeds it in some aspects such as its ability to drive longer cable lengths. (The DSI bus is not part of this standard as it refers to the equipment side of the network.)

The data link layer protocol is responsible for the safe delivery of frames across the network. Here, ISDN uses the

CCITT standard Q.921 which is more commonly known as LAPD, or "Link Access Protocol on the D Channel". LAPD defines the "HDLC" frame format and a set of procedures to control the flow of information on the network, and recovery from errors. It is similar to the LAPB link access protocol used in X.25 and true HDLC networks, but defines an expanded set of procedures to cope with communications on a telephone network instead of a typical computer network.

Finally, in Layer 3, the CCITT Q.931 standard specifies a series of procedures for establishing, maintaining, and disconnecting calls between users on the network. Part of these services are application dependent, so in order to make the ISDN standard generic as possible, the Layer 3 is split into two parts. The generic part of Layer 3 executes the "protocol control procedures" and the application dependent part performs the "Call Control Procedures".

Figure 10 also shows how the ISDN protocols are mapped onto the hardware components. The SID is the Layer 1 device and the HPC16400 provides hardware support, by means of its HDLC channels, for the Layer 2 protocol. The clear boundary between the Layer 1 and Layer 2 devices results in a well structured system architecture, with the DSI bus creating the physical interface between these two layers. The remaining parts of Q.921 and Q.931 are implemented as a software package which includes drivers for the SID and HDLC/DMA channels, and tools which aid the debugging of application tasks that interface to the software at the Layer 3 call control level.

Within the software, the individual layers and drivers are implemented as tasks which run under a multi-tasking executive. The operation of the executive has been optimized to work with layered tasks, and includes features such as a mail manager, timer manager, and memory manager. The entire software package is written in "C" so that application tasks can be developed, run with the layer software (excluding the drivers), and debugged on a PC before being ported to the target hardware.

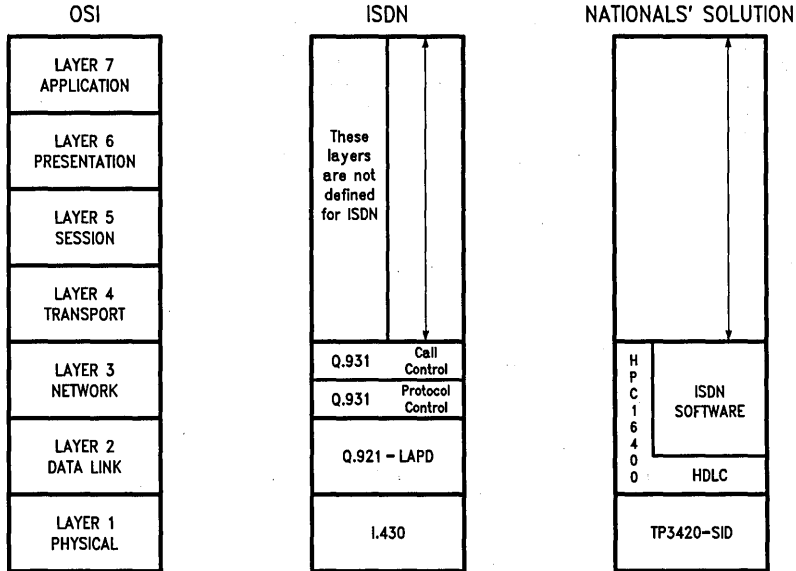


FIGURE 10. National's Solution to ISDN

CONCLUSIONS

The HPC16400 is a versatile high performance 16-bit CMOS microcontroller for embedded communications applications. Its fast CPU together with dual HDLC channels provides an ideal platform for implementing proprietary or standard communication protocols that use the HDLC framing structure.

REFERENCES

1. *"Expanding the HPC Address Space"*, National Semiconductor Application Note 497.
2. *"Intuitive ISDN—An ISDN Tutorial"*, National Semiconductor Application Note 492.

Signed Integer Arithmetic on the HPC™

National Semiconductor
Application Note 603
Raj Gopalan



This report describes the implementation of signed integer arithmetic operations on the HPC. HPC hardware support for unsigned arithmetic operation. In order to support signed integer arithmetic operations on the HPC, the user can represent negative numbers in two's complement form and perform the signed arithmetic operations explicitly through software.

The following signed integer arithmetic routines are implemented in the package:

Multiplication:

16 by 16 yielding 16-bit result
32 by 32 yielding 32-bit result

Division:

16 by 8 yielding 16-bit quotient and 16-bit remainder
32 by 16 yielding 16-bit quotient and 16-bit remainder
32 by 32 yielding 16-bit quotient and 16-bit remainder

Addition:

16 by 16 yielding 16-bit

Subtraction:

16 by 16 yielding 16-bit

Comparison:

16 by 16 for greater to, less than or equal to.

REPRESENTATION OF NEGATIVE NUMBERS:

For binary numbers, negative numbers are represented in two's complement form. In this system, a number is positive if the MSB is 0, negative if it is 1.

The decimal equivalent of two's complement number is computed the same as for an unsigned number, except that weight of the MSB is $-2^{**n} - 1$ instead of $+2^{**n} - 1$. The range of representable numbers is $-(2^{**n} - 1)$ through $+(2^{**n} - 1 - 1)$.

The two's complement of a binary number is obtained by complementing its individual bits and adding one to it.

The advantage of representing a negative number in two's complement form is that addition and subtraction can be done directly using unsigned hardware.

```

        .title      SIMUSL
        .sect       code,rom8,byte,rel
;Signed multiply (16 by 16)
;      B      Multiplicand
;      A      Multiplier
;      X;A    return
;
        .public signed_mult_16
        .local
signed_mult_16:
        st        a,0.w
        mult      a,b                ;do unsigned multiplication.
        sc
        ifbit    7,(1).b            ;if multiplier is negative
        subc     x,b
        sc
        ifbit    7,(B+1).b          ;if multiplicand is negative
        subc     x,0.w
$exit:
        ret
        .endsect

```

MULTIPLICATION**Method 1:**

Signed multiplication can be achieved by taking care of the signs and magnitudes of the multiplicand and multiplier separately.

Perform the multiplication on the magnitudes alone.

The sign of the result can be set based on the signs of the multiplier and the multiplicand.

Method 2:

This method does not require finding the magnitude of the operands. Multiplication can be done using unsigned hardware on the two's complement numbers. The result will be signed based on the signs of the operands.

```

        .title      SIMULL
        .sect       code,rom8,byte,rel
;Multiply (Signed or Unsigned are the same)
;32 bit
;
;   K:A           Multiplicand
;   -4:8[SP]      Multiplier
;   K:A           return
;
        .public multiply_32
        .local
multiply_32:
    push        x                ;(Argument now at -6:8[SP])
    st         a,0.w
    ld         a,k                ;Multiply hi reg* lo stack
    mult       a,-8[sp].w
    x         a,0.w                ;hold, retrieve lo reg
    push       a                ;(argument now at -8:10[SP])
    mult       a-8[sp].w        ;Multiply lo reg* hi stack
    add        0.w,a            ;add into hi partial
    pop        a                ;(Argument now at -6:8[SP])
    mult       a,-8[sp].w        ;Multiply lo reg* lo stack
    add        x,0.w            ;add in hi partial
    ld         k,x                ;Position
    pop        x                ;Restore
    ret

        .endsect

```

The algorithm is as follows:

Step 1. Result = op1 * op2

Step 2. If op1 < 0 then subtract op2 from upper half of the result.

Step 3. If op2 < 0 then subtract op1 from upper half of the result.

Now the Result will yield the correct value of the multiplication on two's complement numbers.

Method 3:

By sign extending the multiplier and multiplicand to the size of the result one can always obtain the correct result of signed multiplication using unsigned multiplication.

DIVISION

Similar to multiplication method 1, one can perform the division on the magnitudes of the dividend and divisor.

The sign of the quotient can be set based on the signs of the dividend and the divisor.

The sign of the remainder will be same as the dividend.

```

        .title      SIDVSS
        .sect       code,rom8,byte,rel

;Division & Remainder
;16,8 bit (signed only, unsigned uses inline code)
;
;   A      Dividend
;   -4[SP] Divisor
;   A      return
;
        .public signed_divide_8,signed_remainder_8
        .public signed_divide_16,signed_remainder_16
        .local
signed_divide_8:
        jsr      $shared_8          ;Uses shared routine
        ret
;
signed_remainder_8:
        jsr      $shared_8          ;Uses shared routine
        ld      a,k                ;Return remainder
        ret
;
$shared_8:
        ifgt    a,#0x7f
        or     a,#0xff00
        st     a,k                  ;Get arguments
        ld     a,-6[sp].w
        ifgt    a,#0x7f
        or     a,#0xff00
        jp     $shared
;
signed_divide_16:
        jsr      $shared_16         ;Uses shared routine
        ret
;
signed_remainder_16:
        jsr      $shared_16         ;Uses shared routine
        ld      a,k                ;Return remainder
        ret
;
$share_16:
        st     a,k                  ;Get arguments
        ld     a,-6[sp].w
$shared
        ifeq    a,#0
        ret     ;division by zero
        push   x
        ifgt    a,#0x7fff
        jp     $unknown_negative   ;unknown/negative
        x      a,k
        ifgt    a,#0x7fff
        jp     $negative_positive   ;negative/positive
        div    a,k                 ;Positive/positive is plus,plus
        jp     $positive_positive

```

```

$unknown_negative:                                ;Unknown/negative
    comp      a
    inc      a
    x        a,k
    ifgt     a,#0x7fff
    jp      $negative_negative                    ; negative/negative
    div      a,k                                ;Positive/negative is minus,plus
    comp      a
    inc      a
$positive_positive:
    ld      k,x
    jp      $exit
$negative_positive:                                ;Negative/positive is minus,minus
    comp      a
    inc      a
    div      a,k
    comp      a
    inc      a
    jp      $negate_remainder
$negative_negative:                                ;Negative/negative is plus,minus
    comp      a
    inc      a
    div      a,k
$negate_remainder:
    x        a,x
    comp      a
    inc      a
    st      a,k
    ld      a,x
$exit:
    pop      x
    ret
    .endsect

```

```

        .title      SIDVLS
        .sect       code,rom8,byte,rel

;Division & Remainder
;Signed 32 by 16 divide
;      X;A      Dividend
;      K        Divisor
;      X,A      return (remainder and quotient)
;
        .public signed_div_32
        .local
signed_div_32:
        sc
        ifeq     k,#0                ;Divide by zero, set carry and return
        ret
$shared_signed:
        ifbit   7,x+1.b
        jp     $negative_dividend
        jsr    $process_divisor      ;Skipping return
        ret     ;+/-+,-,+
$negate_quotient:
        comp   a
        inc   a
        ret     ;+/-=-,-,+
$negative_dividend:
        comp   a
        add   a,#01
        x     a,x
        comp   a
        adc   a,#0
        x     a,x
        jsr    $process_divisor      ;skipping return
        jsr    $negate_quotient     ;-/+=-,-,-
$negate_remainder:
        x     a,x
        comp   a
        inc   a
        x     a,x
        ret
$process_divisor:
        ifbit   7,k+1.b
        jp     $negative_divisor
        divd   a,k                  ;?/+
        ret
$negative_divisor:
        x     a,k
        comp   a
        inc   a
        x     a,k
        divd   a,k                  ;?/-
        retsk
        .endsect

```

```

        .title      SUDVLL
        .sect       code,rom8,byte,rel

;Division & Remainder
;Signed 32 by 32 Divide
;
;   K:A          Dividend
;   -4:6[SP]     Divisor
;   K:A          return
;
;Stack frame as built and used consists of
;top:
;   0, initial subtrahend hi /dividend shifts into subtrahend
;   0, initial subtrahend lo /becomes remainder
;   k, dividend hi /dividend shifts into subtrahend, and
;   a, dividend lo /quotient shifts into dividend
;   b preserved
;   x preserved
;   return address
;   sp-4-12, divisor hi
;   sp-6-12, divisor lo
;Sign flag (0 = negative, 1 = positive, for test sense at exit)
;bit 0, divisor sign (1 = negative)
;bit 1, dividend sign (1 = positive)
;Inc of flag causes bit 1 = (bit 1 xor bit 0) by carry/nocarry out of bit 0
;so that two positives (010) or two negatives (001) indicate a positive
;quotient (011 or 010) in bit 1. Bit 1 always indicates sign if remainder.
;Operation is indicated by bit 3 of the flag, 1 = remainder.
;
        .public signed_divide_32, signed_remainder_32
        .public unsigned_divide_32, unsigned_remainder_32
        .local
signed_divide_32:
        ld         l.b,#0x02
        jp         $shared_signed
;
signed_remainder_32:
        ld         l.b,#0x0a
$shared_signed:
        ifbit      7,k+1.b           ;Check dividend
        jsr        $negate           ;Negate dividend and note sign
        ifbit      7,-6+3[sp].b     ;Check divisor
        jp         $negate_divisor
        jmp        $shared
;
$negate_divisor:
        x          a,-6[sp].w       ;Negate divisor and note sign
        comp      a
        add       a,#1
        x          a,-6[sp].w
        x          a,-4[sp].w
        comp      a
        adc       a,#0
        x          a,-4[sp].w
        sbit     0,l.b
        jp         $shared
;
unsigned_divide_32:
        ld         l.b,#0x02
        jp         $shared
;
unsigned_remainder_32:
        ld         l.b,#0x0a

```



```

$shared:
    push    x                ;Preserve registers
    push    b
    ld      b,sp            ;Place dividend, becomes quotient
    push    a
    push    k
    ld      x,sp            ;Set subtrahend, becomes remainder
    clr     a
    push    a
    push    a
    ld      k,#-18          ;Access divisor argument
    add     k,sp
    ld      a,[k].w
    or      a,2[k].w
    ifeq   a,#0
    jmp     $zero           ;division by zero
    ld      0.b,#32        ;Set counter

$loop:
    ld      a,[b].w        ;Shift Dividend:Quotient
    shl    a
    xs     a,[b+].w
    nop
    ld      a,[b].w
    rlc    a
    xs     a,[b-].w
    nop
    ld      a,[x].w
    rlc    a
    x      a,[x+].w
    ld      a,[x].w
    rlc    a
    x      a,[x-].w
    ifc
    jp     $subtract      ;Carry out - dividend divisor
    sc     ;Check for dividend divisor
    ld      a,[x+].w
    subc   a,[k].w
    ld      a,[x-].w
    subc   a,2[k].w
    ifnc
    jp     $count         ;dividend divisor

$subtract:
    ld      a,[x].w        ;Subtract out divisor (c is set)
    subc   a,[k].w
    x      a,[x+].w
    ld      a,[x].w
    subc   a,2[k].w
    x      a,[x-].w
    sbit   0,[b].b        ;Set quotient bit

$count:
    decsz  0.b            ;Count 32 shifts
    jmp    $loop

$zero:
    pop    k                ;Get Remainder and/or Quotient
    pop    a                ;and clear working off stack
    pop    x
    pop    b
    ifbit 3,1.b
    jp     $exit           ;want remainder, have it
    ld      a,b            ;Want Quotient
    ld      k,x
    inc    1.b            ;Divisor's sign Xors Dividend's

```

```
$exit:      pop          b                ;Restore registers
            pop          x
            ifbit       1,1,b
            ret                    ;positive result

$negate:    comp         a                ;Negate K:A
            add         a,#1
            x           a,k
            comp        a
            adc         a,#0
            x           a,k
            rbit        1,1,b           ;Note sign (for entrance)
            ret

            .endsect
```

ADDITION

Two's complement numbers can be added by ordinary binary addition, ignoring any carries beyond the MSB. The result will always be the correct sum as long as the result doesn't exceed the range.

If the result is the same as for the subtrahend, then overflow has occurred.

```

        .title      SIADD
        .sect       code,rom8,byte,rel
;Signed add (16 by 16)
;
;   A      Operand1
;   B      Operand2
;   Carry  Return
        .public sign_add
        .local

sign_add:
        ld         0,b,#00
        ifbit 7,(A+1).b
        inc       0,b
        ifbit 7,(B+1).b
        inc       0,b

        ;if bit 0 of 0.b = 1 then op1 and op2 have different sign
        ;if bit 0 of 0.b = 0 then op1 and op2 sign are same
        ;then if bit 1 of 0.b = 0 both operands are positive
        ;else both operands are negative.

        add       a,b                ;Perform unsigned addition
        rc
        ifbit 0,0.b                ;both operands are different sign
        ret
        ifbit 1,0.b                ;both op1 and op2 are negative
        jp $negatives
$positives:                          ;both op1 and op2 are positive
        ifbit 7,(A+1).b            ;if result sign is negative then
                                    set overflow bit
        sc
        ret                        ;overflow
$negatives:                          ;if sign bit of result is
        ifbit 7,(A+1).b            negative, then no overflow

        ret
        sc                          ;overflow
$exit:
        ret

        .endsect

```

SUBTRACTION

Subtraction can be achieved by negating the subtrahend and perform the addition operation.

Overflow can be detected as mentioned before by checking the signs of minuend and the negation of the subtrahend and that of the sum.

```

        .title      SISUB
        .sect       code,rom8,byte,rel

;Signed subtract (16 by 16)

;      B      Operand1
;      A      Operand2
;      Carry,A Return
        .public sign_sub
        .local

sign_sub:
        ld        0.b,#00                ;initialize sign flags
        ifbit    7,(B+1).b
        inc 0.b

$negate_A:
        comp A
        inc A

$ngative_comp_A:
        ifbit 7,(A+1).b
        inc 0.b
        ;if bit 0 of 0.b = 1 then opl and op2 have different sign
        ;if bit 0 of 0.b = 0 then opl and op2 sign are same
        ;then if bit 1 of 0.b = 0 both operands are positive
        ;else both operands are negative.
        add A,B                          ;Perform unsigned addition
        rc
        ifbit 0,0.b                       ;both operands are different sign
        ret
        ifbit 1,0.b                       ;both opl and op2 are negative
        jp $negatives

$positives:                               ;both opl and op2 are positive
        if bit 7, (A+1).b                 ;if result sign is negative then
                                           set overflow bit
        sc                                ;bit 0 of byte 0.b is set to
                                           indicate overflow

        ret

$negatives:                               ;if sign bit of result is
        ifbit 7, (A+1).b                 negative, then no overflow

        ret
        sc                                ;sign bit of result is positive,
                                           hence overflow.

$exit:ret

        .endsect

```

```

        .title      NSISUB
        .sect       code,rom8,byte,rel

;Signed sub (16 by 16)

;      A      Operand1
;      B      Operand2
;      Carry   Return
        .public sign_sub
        .local

sign_sub:
        ld        0.b,#00
        ifbit 7,(A+1).b
        inc       0.b
        ifbit 7,(B+1).b
        inc       0.b
        ;if bit 0 of 0.b = 1 then opl and op2 have different sign
        ;if bit 0 of 0.b = 0 then opl and op2 sign are same
        ;then if bit 1 of 0.b = 0 both operands are positive
        ;else both operands are negative.
        sc
        subc      a,b                ;Perform unsigned addition
        rc
        ifbit 0,0.b                ;both operands are different sign
        jp        $chkovf
        ret                        ;both operands are same sign,
                                   can't produce overflow

$chkovf:
        ifbit     7,(B+1).b
        jp        $negminu

$posminu:
        ifbit     7,(A+1).b
        sc
        ret

$negminu:
        ifbit     7,(A+1).b
        sc
        ret

        .endsect

```

COMPARISON

To do signed comparison on n bit two's complement numbers first add $2^{**}(n - 1)$ to the numbers. This will basically shift the numbers from $-(2^{**}n - 1)$ to $+(2^{**}n - 1 - 1)$ range to 0 to $2^{**}n - 1$.

Now comparison operations on the numbers will produce the correct result.

```

        .title      SICMP
        .sect       code,rom8,byte,rel

;Signed compare (16 by 16)

;      A           Operand1
;      B           Operand2
;      0.b        Return=00                if a = b
;                                     02                if a > b
;                                     01                if a < b
signed_compare:
        push       a
        push       b
        add        a,#08000
        add        b,#08000
        ifgt      a,b                      $great
        jp        $great
        ifeq      a,b                      $less
        jp        $less
$less:
        ld        0.b,#01
        pop       b
        pop       a
        ret
$great:
        ld        0.b,#02
        pop       b
        pop       a
        ret
$equ:
        ld        0.b,#00
        pop       b
        pop       a
        ret

        .endsect

```

EMI/RFI Board Design

National Semiconductor
Application Note 643
Joe Cocovich



INTRODUCTION

The control and minimization of Electro-Magnetic Interference (EMI) is a technology that is, out of necessity, growing rapidly. EMI will be defined shortly but, for now, you might be more familiar with the terms Radio Noise, Electrical Noise, or Radio Frequency Interference (RFI). The technology's explorations include a wide frequency spectrum, from dc to 40 GHz. It also deals with susceptibility to EMI as well as the emissions of EMI by equipment or components. Emission corresponds to that potential EMI which comes out of a piece of equipment or component. Susceptibility, on the other hand, is that which couples from the outside to the inside.

In HPC designs to date, we have looked at noise situations ranging from 2 MHz to 102 MHz. EMI, in some cases, can affect radio reception, TV reception, accuracy of navigation equipment, etc. In severe cases, EMI might even affect medical equipment, radar equipment, and automotive systems.

This Application Note will define ElectroMagnetic Interference and describe how it relates to the performance of a system. We will look at examples of Inter-system noise and Intra-system noise and present techniques that can be used to ensure ElectroMagnetic Compatibility throughout a system and between systems.

We will investigate and study the sources of noise between systems through wire-harness and backplane cables and connectors. Active circuit components can be contributors of noise and be susceptible to it. The fast switching times of CMOS devices fabricated in today's technology can cause incredible noise in a system. This noise typically is made up of crosstalk, power supply spiking, transient noise, and ground bounce.

The minimization and suppression of EMI can be obtained by utilizing proper control techniques. Intra-system noise, noise within a single module, sometimes can be controlled with methods such as filtering, shielding, careful selection of components, and following good wiring and grounding procedures. Controlling noise between systems, Inter-system noise, uses subtler techniques such as frequency management and time management, etc.

Appropriate time and resources should be spent during the design of a system or systems to insure that no problems will be encountered due to effects of EMI. Design guidelines will be presented that can be used to increase ElectroMagnetic Compatibility between systems by reducing the effects of noise between them. Above all, don't forget that the development tools used are also systems and are important to consider in your planning.

A brief look will be taken at the environment and tools required for different levels of noise testing. Relative risk-costs between preparing for EMC or excluding EMI concerns from the project will be listed.

DESCRIPTION OF NOISE

ElectroMagnetic Interference

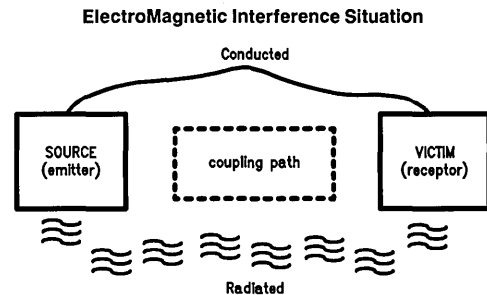
EMI is a form of electrical-noise pollution. Think of the time when an electric drill or some other power tool jammed a nearby radio with buzzing or crackling noises. Sometimes it

got so bad that it prevented you from listening to the radio while the tool was in use. Or the ignition of an automobile idling outside your house caused interference to your TV picture making lines across the screen or even losing sync altogether making the picture flip. These examples are quite annoying but not catastrophic.

More serious, how about a sudden loss in telephone communication caused by electrical interference or noise while you are negotiating an important business deal? Now EMI can be economically damaging.

The results of EMI incidences can be even farther reaching than these examples. Aircraft navigation errors resulting from EMI or interruption of air traffic controller service and maybe even computer memory loss due to noise could cause two aircraft to collide resulting in the loss of lives and property.

These were just a few examples to help you identify the results of EMI in a familiar context. To help understand an ElectroMagnetic Interference situation, the problem can be divided into three categories. They are the source, the victim, and the coupling path. Secondary categories involve the coupling path itself. If the source and victim are separated by space with no hard wire connection, then the coupling path is a radiated path and we are dealing with radiated noise. If the source and victim are connected together through wires, cables, or connectors, then the coupling path is a conducted path and we are dealing with conducted noise. Incidentally, both types of noise can exist at the same time.



TL/DD/10562-1

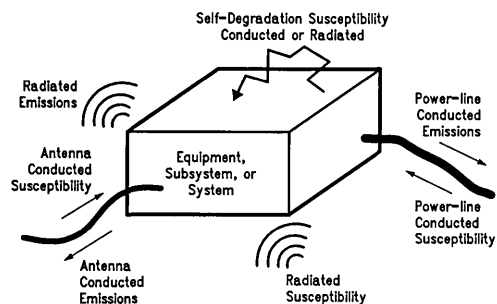
ElectroMagnetic Compatibility

If you think about the examples given, one can understand that EMI or electrical noise is of national concern. The Government and certain industry bodies have issued specifications with which all electrical, electromechanical, and electronic equipment must comply. These specifications and limitations are an attempt to ensure that proper EMC techniques are followed by manufactures during the design and fabrication of their products. When these techniques are properly applied, the product can then operate and perform with other equipment in a common environment such that no degradation of performance exists due to internally or externally conducted or radiated electromagnetic emissions. This is defined as ElectroMagnetic Compatibility or EMC.

Inter-System EMI

For the purpose of this Application Note, when the source of noise is a module, board, or system and the victim is a different and separate module, board, or system under the control of a different user, that is considered to be an inter-system interference situation. Examples of inter-system interference situations could be a Personal Computer interfering with the operation of a TV or an anti-lock brake module in a car causing interference in the radio. This type of interference is more difficult to contain because, as mentioned earlier, the systems are generally not under the control of a single user. However, design methods and control techniques used to contain the intra-system form of EMI, which are almost always under the control of a single user, will inherently help reduce the inter-system noise.

Intra-System EMI Manifestations



TL/DD/10562-2

This Application Note will address problems and solutions in the area of intra-system noise. Intra-system interference situations are when the sources, victims, and coupling paths are entirely within one system or module or PC board. Systems may provide emissions that are conducted out power lines or be susceptible to emissions conducted in through them. Systems may radiate emissions through space as well as be susceptible to radiated noise. Noise conducted out antenna leads turns into radiated noise. By the same token, radiated noise picked up by the antenna is turned into conducted noise within the system. A perfect example is ground loops on a printed circuit board. These loops make excellent antennas. The system itself is capable of degrading performance due to its own internal generation of conducted and radiated noise and its susceptibility to it.

Some results of EMI within a system: Noise on power line causing false triggering of logic circuits, rapidly changing signals causing "glitches" on adjacent steady state signal lines (crosstalk) causing erratic operation, multiple simultaneously switching logic outputs propagating ground bounce noise throughout system, etc.

Coupling Paths

The modes of coupling an emitter source to a receptor victim can become very complicated. Remember, each EMI situation can be classified into two categories of coupling, conducted and radiated. Coupling can also result from a combination of paths. Noise can be conducted from an emitter to a point of radiation at the source antenna, then picked up at the receptor antenna by induction, and re-conducted to the victim. A further complication that multiple

coupling paths presents is that it makes it difficult to determine if eliminating a suspected path has actually done any good. If two or more paths contribute equally to the problem, eliminating only one path may provide little apparent improvement.

Conducted Interference

In order to discuss the various ways in which EMI can couple from one system to another, it is necessary to define a few terms. When dealing with conducted interference, there are two varieties that we are concerned with. The first variety is differential-mode interference. That is an interference signal that appears between the input terminals of a circuit. The other variety of conducted interference is called common-mode interference. A common-mode interference signal appears between each input terminal and a third point; that third point is called the common-mode reference. That reference may be the equipment chassis, an earth ground, or some other point.

Let's look at each type of interference individually. In *Figure 1* we show a simple circuit consisting of a signal source, V_S , and a load, R_L . In *Figure 2* we show what happens when differential-mode interference is introduced into the circuit by an outside source. As is shown, an interference voltage, V_D , appears between the two input terminals, and an interference current, I_D , flows in the circuit. The result is noise at the load. If, for instance, the load is a logic gate in a computer, and the amplitude of V_D is sufficiently high, it is possible for the gate to incorrectly change states.

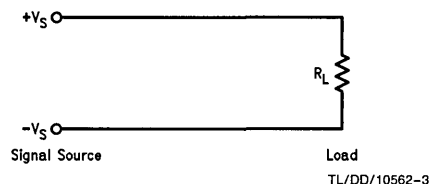
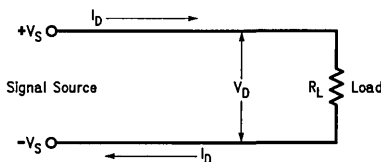


FIGURE 1

TL/DD/10562-3



TL/DD/10562-4

FIGURE 2. Differential-Mode Interference

Figure 3 shows what happens when a ground loop is added to our circuit. Ground loops, which are undesirable current paths through a grounded body (such as a chassis), are usually caused by poor design or by the failure of some component. In the presence of an interference source, common-mode currents, I_C , and a common-mode voltage, V_C , can develop, with the ground loop acting as the common-mode reference. The common-mode current flows on both input lines, and has the same instantaneous polarity and direction (the current and voltage are in phase), and returns through the common-mode reference. The common-mode voltage between each input and the common-mode reference is identical.

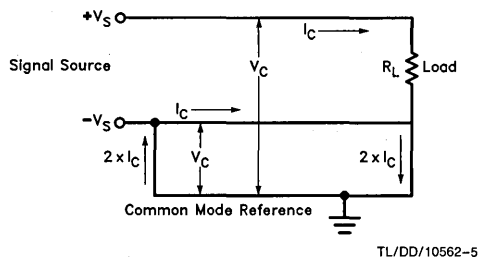


FIGURE 3. Common-Mode Interference

TL/DD/10562-5

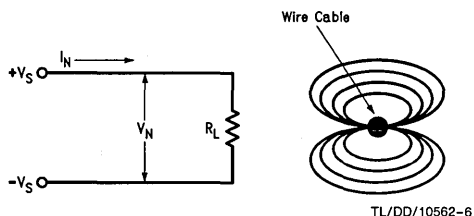


FIGURE 4. Field-to-Cable Coupling

TL/DD/10562-6

Radiated Interference

Radiated coupling itself can take place in one of several ways. Some of those include field-to-cable coupling, cable-to-cable coupling, and common-mode impedance coupling. Let's look at those types of coupling one at a time.

The principle behind field-to-cable coupling is the same as that behind the receiving antenna. That is, when a conductor is placed in a time-varying electromagnetic field, a current is induced in that conductor. That is shown in *Figure 4*. In this figure, we see a signal source, V_S , driving a load, R_L . Nearby there is a current carrying wire (or other conductor). Surrounding the wire is an electromagnetic field induced by the current flowing in the wire. The circuit acts like a loop antenna in the presence of this field. As such, an interference current, I_N , and an interference voltage, V_N , are induced in the circuit. The magnitude of the induced interference signal is roughly proportional to the frequency of the incoming field, the size of the loop, and the total impedance of the loop.

Cable-to-cable coupling occurs when two wires or cables are run close to one another. *Figure 5* shows how cable-to-cable coupling works. *Figure 5a* shows two lengths of cable (or other conductors) that are running side-by-side. Because any two conducting bodies have capacitance between them, called stray capacitance, a time-varying signal in one wire can couple via that capacitance into the other wire. That is referred to as capacitive coupling. This stray capacitance, as shown in *Figure 5c* makes the two cables behave as if there were a coupling capacitor between them. Another mechanism of cable-to-cable coupling is mutual inductance. Any wire carrying a time-varying current will develop a magnetic field around it. If a second conductor is placed near enough to that wire, that magnetic field will induce a similar current in the second conductor. That type of coupling is called inductive coupling. Mutual inductance, as shown in *Figure 5b*, makes the cables behave as if a poorly wound transformer were connected between them. In cable-to-cable coupling, either or both of those mechanisms may be

responsible for the existence of an interference condition. Though there is no physical connection between the two cables, the properties we have just described make it possible for the signal on one cable to be coupled to the other.

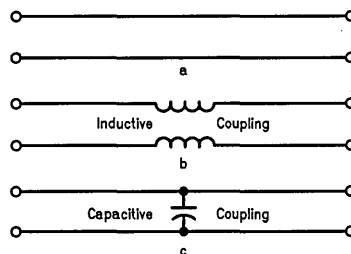


FIGURE 5. Cable-to-Cable Coupling

TL/DD/10562-7

Either or both of the above-mentioned properties cause the cables to be electromagnetically coupled such that a time-varying signal present on one will cause a portion of that signal to appear on the other. The "efficiency" of the coupling increases with frequency and inversely with the distance between the two cables. One example of cable-to-cable coupling is telephone "crosstalk", in which several phone conversations can be overheard at once. The term crosstalk is now commonly used to describe all types of cable-to-cable coupling.

Common-mode impedance coupling occurs when two circuits share a common bus or wire. In *Figure 6* we show a circuit that is susceptible to that type of coupling. In that figure a TL092 op-amp and a 555 timer share a common return or ground. Since any conductor (including a printed circuit board trace) is not ideal, that ground will have a non-zero impedance, Z . Because of that, the current, I , from pin 1 of the 555 will cause a noise voltage, V_N , to develop; that voltage is equal to $I \times Z$. That noise voltage will appear in series with the input to the op-amp. If that voltage is of sufficient amplitude, a noise condition will result.

While not all inclusive, these coupling paths account for, perhaps, 98% of all intra-system EMI situations.

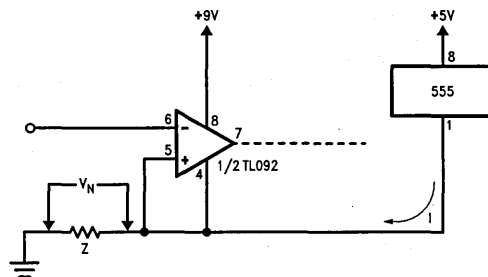


FIGURE 6. Common-Mode Impedance Coupling

TL/DD/10562-8

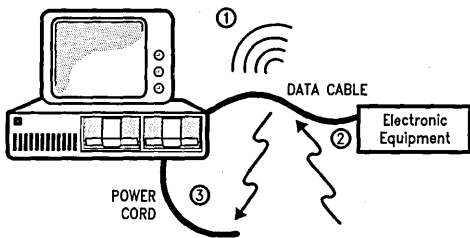
NOISE SOURCES

In this Application Note, we will look at sources of EMI which involve components that may conduct or radiate electromagnetic energy. These sources, component emitters, are different from the equipment and subsystems we have

been talking about. Component emitters are sources of EMI which emanate from a single element rather than a combination of components such as was previously described. Actually, these component emitters require energy and connecting wires from other sources to function. Therefore, they are not true sources of EMI, but are EMI Transducers. They convert electrical energy to electrical noise.

Cables and Connectors

The three main concerns regarding the EMI role of cables are conceptualized in *Figure 7*. They act as (1) radiated emission antennas, (2) radiated susceptibility antennas, and (3) cable-to-cable or crosstalk couplers. Usually, whatever is done to harden a cable against radiated emission will also work in reverse for controlling EMI radiated susceptibility. The reason for the word usually, is that when differential-mode radiated emission or susceptibility is the failure mode, twisting leads and shielding cables reduces EMI. If the failure mechanism is due to common-mode currents circulating in the cable, twisting leads has essentially no effect on the relationship between each conductor and the common-mode reference. Also cable shields may help or aggravate EMI depending upon the value of the transfer impedance of the cable shield. Transfer impedance is a figure of merit of the quality of cable shield performance defined as the ratio of coupled voltage to surface current in ohms/meter. A good cable shield will have a low transfer impedance. The effectiveness of the shield also depends on whether or not the shield is terminated and, if so, how it is terminated.



TL/DD/10562-9

FIGURE 7. Cables and Connectors

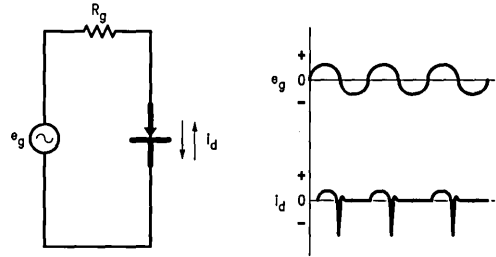
Connectors usually are needed to terminate cables. When no cable shields or connector filters or absorbers are used, connectors play essentially no role in controlling EMI. The influence of connector types, however, can play a major role in the control of EMI above a few MHz. This applies especially when connectors must terminate a cable shield and/or contain lossy ferrites or filter-pins.

Connectors and cables should be viewed as a system to cost-effectively control EMI rather than to consider the role of each separately, even though each offers specific interference control opportunities.

Components

Under conditions of forward bias, a semiconductor stores a certain amount of charge in the depletion region. If the diode is then reverse-biased, it conducts heavily in the reverse direction until all of the stored charge has been removed as shown in *Figure 8*. The duration, amplitude, and configuration of the recovery-time pulse (also called switching time or period) is a function of the diode characteristics and circuit parameters. These current spikes generate a broad spectrum of conducted transient emissions. Diodes with mechanical imperfections may generate noise when

physically agitated. Such diodes may not cause trouble if used in a vibration-free environment.



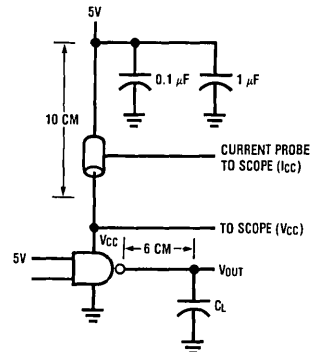
TL/DD/10562-10

FIGURE 8. Diode Recovery Periods and Spikes

Power Supply Noise

Power-supply spiking is perhaps the most important contributor to system noise. When any element switches logic states, it generates a current spike that produces a voltage transient. If these transients become too large, they can cause logic errors because the supply voltage drop upsets internal logic, or because a supply spike on one circuit's output feeds an extraneous noise voltage into the next device's input.

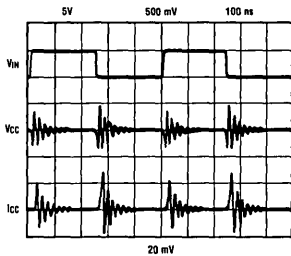
With CMOS logic in its quiescent state, essentially no current flows between V_{CC} and ground. But when an internal gate or an output buffer switches state, a momentary current flows from V_{CC} to ground. The switching transient caused by an unloaded output changing state typically equals 20 mA peak. Using the circuit shown in *Figure 9*, you can measure and display these switching transients under different load conditions.



TL/DD/10562-11

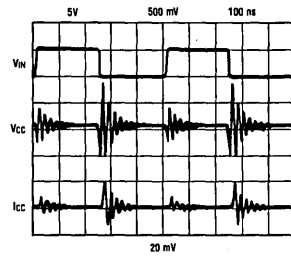
FIGURE 9

Figure 10a shows the current and voltage spikes resulting from switching a single unloaded ($C_L = 0$ in *Figure 9*) NAND gate. These current spikes, seen at the switching edges of the signal on V_{IN} , increase when the output is loaded. *Figures 10b*, *10c*, and *10d* show the switching transients when the load capacitance, C_L , is 15 pF, 50 pF, and 100 pF, respectively. The large amount of ringing results from the test circuit's transmission line effects. This ringing occurs partly because the CMOS gate switches from a very high impedance to a very low one and back again. Even for medium-size loads, load capacitance current becomes a major current contributor.



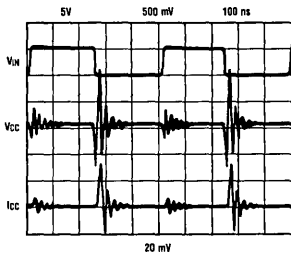
TL/DD/10562-12

a



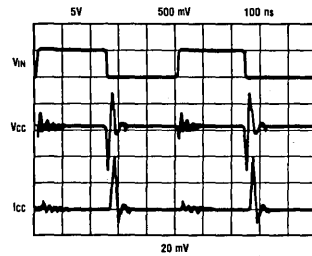
TL/DD/10562-13

b



TL/DD/10562-14

c

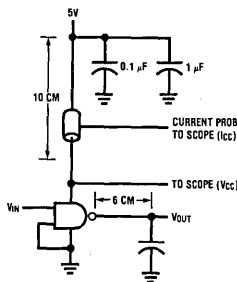


TL/DD/10562-15

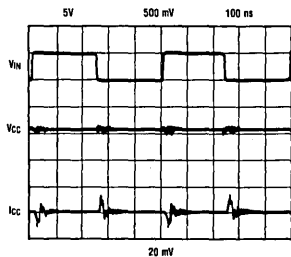
d

FIGURE 10

Although internal logic generates current spikes when switching, the bulk of a spike's current comes from output circuit transitions. *Figure 11* shows the I_{CC} current for a NAND gate, as shown in the test circuit, with one input switching and the other at ground resulting in no output transitions. Note the very small power-supply glitches provoked by the input-circuit transitions.



TL/DD/10562-16



TL/DD/10562-17

FIGURE 11

High-Speed CMOS Logic Switching

The magnitude of noise which can be tolerated in a system relates directly to the worst case noise immunity specified for the logic family. Noise immunity can be described as a device's ability to prevent noise on its input from being transferred to its output. It is the difference between the worst case output levels (V_{OH} and V_{OL}) of the driving circuit and the worst case input voltage requirements (V_{IH} and V_{IL} , respectively) of the receiving circuit.

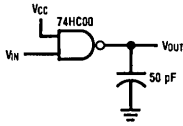
Using *Figure 12* as a guide, it can be seen that for TTL (LS or ALS) devices the worst case noise immunity is typically 700 mV for the high logic level and 300 mV for the low logic level. For HCMOS devices the worst case noise immunity is typically 1.75V for high logic levels and 800 mV for low logic levels. AC high speed CMOS logic families have noise immunity of 1.75V for high logic levels and 1.25V for low logic levels. ACT CMOS logic families have noise immunity of 2.9V for high logic levels and 700 mV for low logic levels.

Logic Family Comparisons

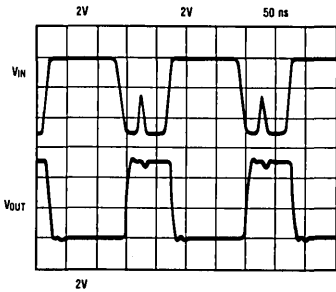
Characteristic	Symbol	LS/ALS TTL	HCMOS	AC	ACT
Input Voltage (Limits)	V_{IH} (Min)	2.0V	3.15V	3.15V	2.0V
	V_{IL} (Max)	0.8V	0.9V	1.35V	0.8V
Output Voltage (Limits)	V_{OH} (Min)	2.7V	$V_{CC}-0.1$	$V_{CC}-0.1$	$V_{CC}-0.1$
	V_{OL} (Max)	0.5V	0.1V	0.1V	0.1V

FIGURE 12

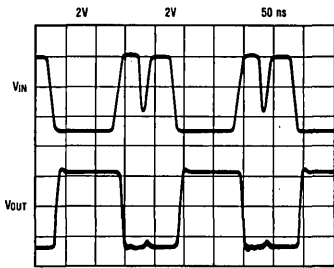
To illustrate noise margin and immunity, *Figure 13* shows the output that results when you apply several types of simulated noise to a 74HC00's input. Typically, even 2V or more input noise produces little change in the output. The top trace shows noise induced on the high logic level signal and the bottom trace shows noise induced on the low logic level signal.



TL/DD/10562-18



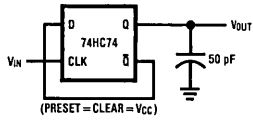
TL/DD/10562-19



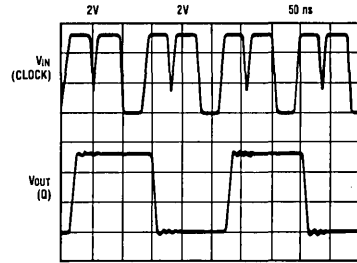
TL/DD/10562-20

FIGURE 13

Figure 14 shows how noise affects a 74HC74's clock input. Again, no logic errors occur with 2V or more of noise on the clock input.



TL/DD/10562-21



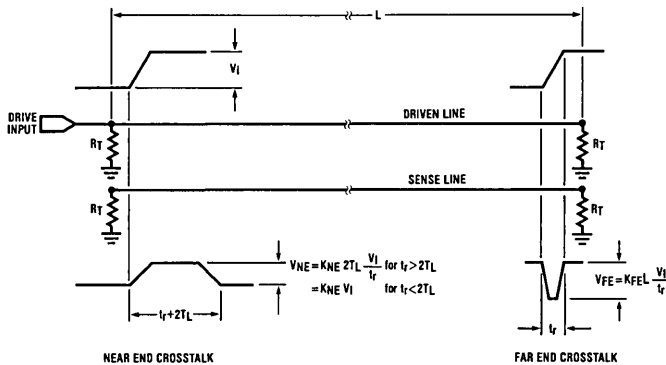
TL/DD/10562-22

FIGURE 14

Signal Crosstalk

The problem of crosstalk and how to deal with it is becoming more important as system performance and board densities increase. Our discussion on cable-to-cable coupling described crosstalk as appearing due to the distributed capacitive coupling and the distributed inductive coupling between two signal lines. When crosstalk is measured on an undriven sense line next to a driven line (both terminated at their characteristic impedances), the near end crosstalk and the far end crosstalk have quite distinct features, as shown in *Figure 15*. It should be noted that the near end component reduces to zero at the far end and vice versa. At any point in between, the crosstalk is a fractional sum of the near and far end crosstalk waveforms as shown in the figure. It also can be noted that the far end crosstalk can have either polarity whereas the near end crosstalk always has the same polarity as the signal causing it.

The amplitude of the noise generated on the undriven sense line is directly related to the edge rates of the signal on the driven line. The amplitude is also directly related to



TL/DD/10562-23

FIGURE 15. Crosstalk

the proximity of the two lines. This is factored into the coupling constants K_{NE} and K_{FE} by terms that include the distributed capacitance per unit length, the distributed inductance per unit length, and the length of the line. The lead-to-lead capacitance and mutual inductance thus created causes "noise" voltages to appear when adjacent signal paths switch.

Several useful observations that apply to a general case can then be made:

- The crosstalk always scales with the signal amplitude V_1 .
- Absolute crosstalk amplitude is proportional to slew rate V_1/t_r , not just $1/t_r$.
- Far end crosstalk width is always t_r .
- For $t_r < 2 T_L$, where t_r is the transition time of the signal on the driven line and T_L is the propagation or bus delay down the line, the near end crosstalk amplitude V_{NE} expressed as a fraction of signal amplitude V_1 is K_{NE} which is a function of physical layout only.
- The higher the value of ' t_r ' (slower transition times) the lower the percentage of crosstalk (relative to signal amplitude).

Although all circuit conductors have transmission line properties, these characteristics become significant when the edge rates of the drivers are equal to or less than about three times the propagation delay of the line. Significant transmission line properties may be exhibited, for example, where devices having edge rates of 3 ns are used to drive traces of 8 inches or greater, assuming propagation delays of 1.7 ns/ft for an unloaded printed circuit trace.

Signal Interconnects

Of the many properties of transmission lines, two are of major interest to the system designer: Z_{oe} , the effective equivalent impedance of the line, and t_{pde} , the effective propagation delay down the line. It should be noted that the intrinsic values of line impedance and propagation delay, Z_o and t_{pd} , are geometry-dependent. Once the intrinsic values are known, the effects of gate loading can be calculated. The loaded values for Z_{oe} and t_{pde} can be calculated with:

$$Z_{oe} = Z_o \cdot (1 + C_l/C_i) \cdot 0.5$$

$$t_{pde} = t_{pd} \cdot (1 + C_l/C_i) \cdot 0.5$$

where C_i = intrinsic line capacitance

C_l = additional capacitance due to gate loading.

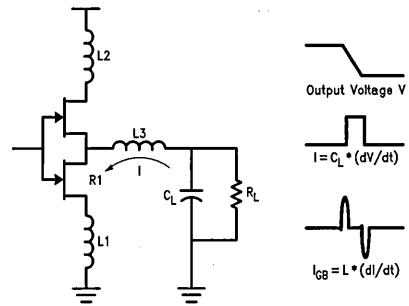
These formulas indicate that the loading of lines *decreases* the effective impedance of the line and *increases* the propagation delay. As was mentioned earlier, lines that have a propagation delay greater than one third the rise time of the signal driver should be evaluated for transmission line effects. When performing transmission line analysis on a bus, only the longest, most heavily loaded and the shortest, least loaded lines need to be analyzed. All lines in a bus should be terminated equally; if one line requires termination, all lines in the bus should be terminated. This will ensure similar signals on all of the lines.

Ground Bounce

Ground bounce occurs as a result of the intrinsic characteristics of the leadframes and bondwires of the packages used to house CMOS devices. As edge rates and drive capability increase in advanced logic families, the effects of these intrinsic electrical characteristics become more pronounced. One of these parasitic electrical characteristics is the inductance found in all leadframe materials.

Figure 16 shows a simple circuit model for a CMOS device in a leadframe driving a standard test load. The inductor L1

represents the parasitic inductance in the ground lead of the package; inductor L2 represents the parasitic inductance in the power lead of the package; inductor L3 represents the parasitic inductance in the output lead of the package; the resistor R1 represents the output impedance of the device output, and the capacitor and resistor C_L and R_L represent the standard test load on the output of the device.



TL/DD/10562-24

FIGURE 16. Ground Bounce

The three waveforms shown represent how ground bounce is generated. The top waveform shows the voltage (V) across the load as it is switched from a logic HIGH to a logic LOW. The output slew rate is dependent upon the characteristics of the output transistor, and the inductors L1 and L3, and the load capacitance. In order to change the output from a HIGH to a LOW, current must flow to discharge the load capacitance. The second waveform shows the current that is generated as the capacitor discharges [$I = -C_L \cdot (dV/dt)$]. This current, as it changes, causes a voltage to be generated across the inductances in the circuit. The formula for the voltage across an inductor is $V = L(dI/dt)$. The third waveform shows the voltage that is induced across the inductance in the ground lead due to the changing currents [$V_{GB} = L1 \cdot (dI/dt)$]. This induced voltage creates what is known as ground bounce.

Because the inductor is between the external system ground and the internal device ground, the induced voltage causes the internal ground to be at a different potential than the external ground. This shift in potential causes the device inputs and outputs to behave differently than expected because they are referenced to the internal device ground, while the devices which are either driving into the inputs or being driven by the outputs are referenced to the external system ground. External to the device, ground bounce causes input thresholds to shift and output levels to change.

Although this discussion is limited to ground bounce generated during HIGH-to-LOW transitions, it should be noted that the ground bounce is also generated during LOW-to-HIGH transitions. This ground bounce though, has a much smaller amplitude and therefore does not present the same concern.

There are many factors which affect the amplitude of the ground bounce. Included are:

- Number of outputs switching simultaneously: more outputs results in more ground bounce.
- Type of output load: capacitive loads generate two to three times more ground bounce than typical system traces. Increasing the capacitive load to approximately 60–70 pF, increases ground bounce. Beyond 70 pF, ground bounce drops off due to the filtering effect of the load itself. Moving the load away from the output also reduces the ground bounce.

- Location of the output pin: outputs closer to the ground pin exhibit less ground bounce than those further away due to effectively lower L1 and L3.
- Voltage: lowering V_{CC} reduces ground bounce.

Ground bounce produces several symptoms:

- Altered device states.
- Propagation delay degradation.
- Undershoot on active outputs. The worst-case undershoot will be approximately equal to the worst-case quiet output noise.

NOISE SUPPRESSION TECHNIQUES

EMI control techniques involve both hardware implementations and methods and procedures. They may also be divided into intra-system and inter-system EMI control. Our major concern in this Application Note is intra-system EMI control, however, an overview of each may be appropriate at this time.

Figure 17 illustrates the basic elements of concern in an intra-system EMI problem. The test specimen may be a single box, an equipment, subsystem, or system (an ensemble of boxes with interconnecting cables). From a strictly near-sighted or selfish point-of-view, the only EMI concern would appear to be degradation of performance due to self-jamming such as suggested at the top of the figure. While this might be the primary emphasis, the potential problems associated with either (1) susceptibility to outside conducted and/or radiated emissions or (2) tendency to pollute the outside world from its own undesired emissions, come under the primary classification of intra-system EMI. Corresponding EMI-control techniques, however, address themselves to both self-jamming and emission/susceptibility in accordance with applicable EMI specifications. The techniques that will be discussed include filtering, shielding, wiring, and grounding.

Inter-system EMI distinguishes itself by interference between two or more discrete and separate systems or platforms which are frequently under independent user control. Culprit emissions and/or susceptibility situations are divided into two classes: (1) antenna entry/exit and (2) back-door entry/exit. More than 95% of inter-system EMI problems involve the antenna entry/exit route of EMI. We can group inter-system EMI-control techniques by four fundamental categories: frequency management, time management, location management, and direction management.

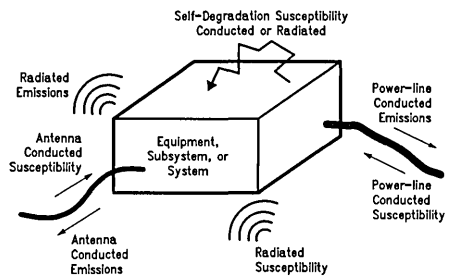


FIGURE 17. Intra-System EMI Manifestations

TL/DD/10562-25

The first step in locating a solution is to identify the problem as either an inter-system or intra-system EMI situation. Generally, if the specimen has an antenna and the problem develops from what exits or enters the antenna from another specimen or ambient, then the problem is identified as an inter-system EMI one. Otherwise, it is an intra-system EMI situation which we will discuss now.

Intra-System EMI-Control Techniques

Shielding

Shielding is used to reduce the amount of electromagnetic radiation reaching a sensitive victim circuit. Shields are made of metal and work on the principle that electromagnetic fields are reflected and/or attenuated by a metal surface. Different types of shielding are needed for different types of fields. Thus, the type of metal used in the shield and the shield's construction must be considered carefully if the shield is to function properly. The ideal shield has no holes or voids, and, in order to accommodate cooling vents, buttons, lamps, and access panels, special meshes and "EMI-hardened" components are needed.

Once a printed-circuit board design has been optimized for minimal EMI, residual interference can be further reduced if the board is placed in a shielded enclosure. A box's shielding effectiveness in decibels depends on three main factors: its skin, the control of radiation leakage through the box's apertures or open areas (like cooling holes), and the use of filters or shields at entry or exit spots of cables.

A box skin is typically fabricated from sheet metal or metalized plastic. Normally sheet metal skin that is 1 mm thick is more than adequate; it has a shielding effectiveness of more than 100 dB throughout the high-frequency spectrum from 1 MHz to 20 GHz. Conductive coatings on plastic boxes are another matter. Table I shows that at 10 MHz the shielding effectiveness can be as low as 27 dB if a carbon composite is used, or it can run as high as 106 dB for zinc sprayed on plastic by an electric arc process. Plastic filled materials or composites having either conductive powder, flakes, or filament are also used in box shielding; they have an effectiveness similar to that of metalized plastics.

TABLE I

Shielding Material	Surface Resistance,* Ohms/Square	Shielding Effectiveness, dB		
		At 10 MHz	At 100 MHz	At 1 GHz
Silver Acrylic Paint	0.004	67	93	97
Silver Epoxy Paint	0.1	59	81	87
Silver Deposition	0.05	57	82	89
Nickel Composite	3.0	35	47	57
Carbon Composite	10.0	27	35	41
Arc-Sprayed Zinc	0.002	106	92	98
Wire Screen (0.64 mm Grid)	N.A.	86	66	48

*Effectiveness of shielding materials with 25- μ m thickness and for frequencies for which the largest dimension of the shielding plate is less than a quarter of a wavelength.

In many cases shielding effectiveness of at least 40 dB is required of plastic housings for microcontroller-based equipment to reduce printed-circuit board radiation to a level that meets FCC regulations in the United States or those of the VDE in Europe. Such skin shielding is easy to achieve. The problem is aperture leakage. The larger the aperture, the greater its radiation leakage because the shield's natural attenuation has been reduced. On the other hand, multiple small holes matching the same area as the single large aperture can attain the same amount of cooling with little or no loss of attenuation properties.

Filtering

Filters are used to eliminate conducted interference on cables and wires, and can be installed at either the source or the victim. *Figure 18* shows an AC power-line filter. The values of the components are not critical; as a guide, the capacitors can be between 0.01 and 0.001 μF , and the inductors are nominally 6.3 μH . Capacitor C1 is designed to shunt any high-frequency differential-mode currents before they can enter the equipment to be protected. Capacitors C2 and C3 are included to shunt any common-mode currents to ground. The inductors, L1 and L2, are called common-mode chokes, and are placed in the circuit to impede any common-mode currents.

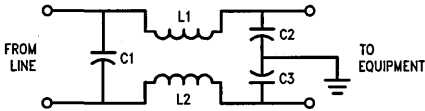


FIGURE 18. Filtering

TL/DD/10562-26

Wiring

Now that the equipment in each box can be successfully designed to combat EMI emission and susceptibility separately, the boxes may be connected together to form a system. Here the input and output cables and, to a lesser extent, the power cable form an "antenna farm" that greatly threatens the overall electromagnetic compatibility of the system. Most field remedies for EMI problems focus on the coupling paths created by the wiring that interconnects systems. By this time most changes to the individual equipment circuits are out of the question.

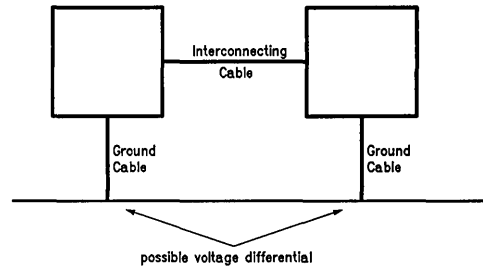
Let us address five coupling paths that are encountered in typical systems comprised of two or more pieces of equipment connected by cables. These should adequately cover most EMI susceptibility problems. They are:

- A *common ground impedance coupling*—a conducting path in which a common impedance is shared between an undesired emission source and the receptor.
- A *common-mode, radiated field-to-cable coupling*, in which electromagnetic fields penetrate a loop formed by two pieces of equipment, a cable connecting them, and a ground plane.
- A *differential-mode, radiated field-to-cable coupling*, in which the electromagnetic fields penetrate a loop formed by two pieces of equipment and an interconnecting transmission line or cable.
- A *crosstalk coupling*, in which signals in one transmission line or cable are capacitively or inductively coupled into another transmission line.

- A *conductive path* through power lines feeding the equipment.

The first coupling path is formed when two pieces of equipment are connected to the same ground conductor at different points, an arrangement that normally produces a voltage difference between the two points. If possible, connecting both pieces of equipment to a single-point ground eliminates this voltage. Another remedy is to increase the impedance along a loop that includes the path between the ground connections of the two boxes. Examples include the isolation of printed-circuit boards from their cabinet or case, the use of a shielded isolation transformer in the signal path, or the insertion of an inductor between one or both boxes and the ground conductor. The use of balanced circuits, differential line drivers and receivers, and absorbing ferrite beads and rods on the interconnecting cable can further reduce currents produced by this undesirable coupling path.

Common Ground Impedance Coupling

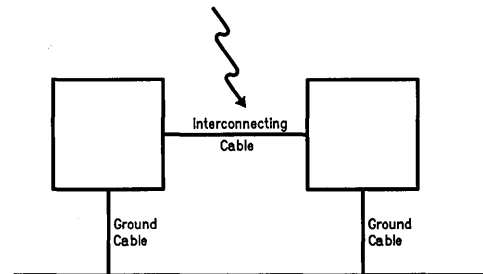


TL/DD/10562-27

A balanced circuit is configured so its two output signal leads are electrically symmetrical with respect to ground, as the signal increases on one output the signal on the other decreases. Differential line drivers produce a signal that is electrically symmetrical with respect to ground from a single-ended circuit in which only one lead is changing with respect to ground. Ferrite beads, threaded over electrical conductors, substantially attenuate electromagnetic interference by turning radio-frequency energy into heat, which is dissipated in them.

In the second coupling path, a radiated electromagnetic field is converted into a common-mode voltage in the ground plane loop containing the interconnect cable and both boxes. This voltage may be reduced if the loop area is trimmed.

Common-Mode, Radiated Field-to-Cable Coupling



TL/DD/10562-28

The third coupling path produces a differential-mode voltage that appears across the input terminals of the EMI receptor. One way of controlling this is to cancel or block the pickup of differential-mode radiation. In a balanced transmission line, this is done by use of twisted-wire pairs and a shielded cable.

As for crosstalk, the fourth coupling path—the reduction of capacitive coupling can be achieved by the implementation of at least one of these steps:

- Reducing the spacing between wire pairs in either or both of the transmission lines.
- Increasing the separation between the two transmission lines.
- Reducing the frequency of operation of the source, if possible.
- Adding a cable shield over either or both transmission lines.
- Twisting the source's or receptor's wire pairs.
- Twisting both wire pairs in opposite directions.

The fifth coupling path conductively produces both common-mode and differential-mode noise pollution on the power mains. Among several remedies that can suppress the EMI here are the filters and isolation transformers.

There are only about 50 common practical remedies that can be used in most EMI situations. Of these, about 10 suffice in 80 percent of the situations. Most engineers are aware of at least some of these remedies—for example, twisting wires to reduce radiation pickup.

In order to attack the EMI problem, one can make use of the information contained in Table II. First, decide what coupling path has the worst EMI interference problem. From the 11 most common coupling paths listed at the top of the table, find the problem coupling path. Using the numbers found in that table entry, locate the recommended remedy or remedies from the 12 common EMI fixes identified at the bottom of the table. This procedure should be repeated until all significant coupling paths have been properly controlled and the design goal has been met.

Inter-System EMI Control Techniques

There are many EMI controls that may be carried out to enhance the chances of inter-system EMC. They can be grouped into four categories which we will discuss briefly. The following discussion is not intended to be complete but merely provide an overview of some EMI control techniques available to the intersystem designer and user.

Frequency management suggests both transmitter emission control and improvement of receptors against spurious responses. The object is to design and operationally maintain transmitters so that they occupy the least frequency spectrum possible in order to help control electromagnetic pollution. For example, this implies that long pulse rise and fall times should be used. Quite often one of the most convenient, economic and rapid solutions to an EMI problem in the field, is to change frequency of either the victim receiver or the culprit source.

In those applications where information is passed between systems, a possible time management technique could be utilized where the amount of information transferred is kept to a minimum. This should reduce the amount of time that the receptor is susceptible to any EMI. In communication protocols, for example, essential data could be transmitted in short bursts or control information could be encoded into fewer bits.

Location management refers to EMI control by the selection of location of the potential victim receptor with respect to all other emitters in the environment. In this regard, separation distance between transmitters and receivers is one of the most significant forms of control since interfering source emissions are reduced greatly with the distance between them. The relative position of potentially interfering transmitters to the victim receiver are also significant. If the emitting source and victim receiver are shielded by obstacles, the degree of interference would be substantially reduced.

Direction management refers to the technique of EMI control by gainfully using the direction and attitude of arrival of electromagnetic signals with respect to the potential victim's receiving antenna.

TABLE II. Electromagnetic Interference Coupling Paths

Radiated Field to Interconnecting Cable (Common-Mode)	2, 7, 8, 9, 11	Radiated Field to Box	12, 13
Radiated Field to Interconnecting Cable (Differential-Mode)	2, 5, 6	Box to Radiated Field	12, 13
Interconnecting Cable to Radiated Field (Common-Mode)	1, 3, 9, 11	Box-to-Box Radiation	12, 13
Interconnecting Cable to Radiated Field (Differential-Mode)	1, 3, 5, 6, 7	Box-to-Box Conduction	1, 2, 7, 8, 9
Cable-to-Cable Crosstalk	1, 2, 3, 4, 5, 6, 10, 11	Power Mains to Box Conduction	4, 11
		Box to Power Mains Conduction	4

Electromagnetic Interference Fixes

1. Insert Filter In Signal Source
2. Insert Filter in Signal Receptor
3. Insert Filter in Power Source
4. Insert Filter in Power Receptor
5. Twist Wire Pair
6. Shield Cable
7. Use Balanced Circuits
8. Install Differential Line Drivers and Receivers
9. Float Printed Circuit Board(s)
10. Separate Wire Pair
11. Use Ferrite Beads
12. Use a Multilayer Instead of a Single-Layer Printed Circuit Boards

DESIGN GUIDELINES

The growth of concern over electromagnetic compatibility (EMC) in electronic systems continues to rise in the years since the FCC proclaimed that there shall be no more pollution of the electromagnetic spectrum. Still, designers have not yet fully come to grips with a major source and victim of electromagnetic interference—the printed circuit board. The most critical stage for addressing EMI is during the circuit board design. Numerous tales of woe can be recounted about the eleventh hour attempt at solving an EMI problem by retrofit because EMC was given no attention during design. This retrofit ultimately costs much more than design stage EMC, holds up production, and generally makes managers unhappy. With these facts in mind, let's address electromagnetic compatibility considerations in printed circuit board design.

Logic Selection

Logic selection can ultimately dictate how much attention must be given to EMC in the total circuit design. The first guideline should be: use the slowest speed logic that will do the job. Logic speed refers to transition times of output signals and gate responses to input signals. Many emissions and susceptibility problems can be minimized if a slow speed logic is used. For example, a square wave clock or signal pulse with a 3 ns rise time generates radio frequency (100 MHz and higher) energy that is gated about on the PC board. It also means that the logic can respond to comparable radio frequency energy if it gets onto the boards.

The type of logic to be used is normally an early design decision, so that control of edge speeds and, hence, emissions and susceptibility is practical early. Of course, other factors such as required system performance, speed, and timing considerations must enter into this decision. If possible, design the circuit with a slow speed logic. The use of slow speed logic, however, does not guarantee that EMC will exist when the circuit is built; so proper EMC techniques should still be implemented consistently during the remainder of the circuit design.

Component Layout

Component layout is the second stage in PC board design. Schematics tell little or nothing about how systems will perform once the board is etched, stuffed, and powered. A circuit schematic is useful to the design engineer, but an experienced EMC engineer refers to the PC board when troubleshooting. By controlling the board layout in the design stage, the designer realizes two benefits: (1) a decrease in EMI problems when the circuit or system is sent for EMI or quality assurance testing; and (2) the number of EMI coupling paths is reduced, saving troubleshooting time and effort later on.

Some layout guidelines for arranging components according to logic speed, frequency, and function are shown in *Figure 19*. These guidelines are very general. A particular circuit is likely to require a combination and/or tradeoffs of the above arrangements. Isolation of the I/O from digital circuitry is important where emissions or susceptibility may be a problem. For the case of emissions, a frequently encountered coupling path involves a digital energy coupling through I/O circuitry and signal traces onto I/O cables and wires, where the latter subsequently radiate. When susceptibility is a problem, it is common for the EMI energy to couple from I/O circuits onto sensitive digital lines, even though the I/O lines may be "opto-coupled" or otherwise supposedly isolated. In both situations, the solution often lies in the proper

electrical and physical isolation of analog and low speed digital lines from high speed circuits. When high speed signals are designed to leave the board, the reduction of EMI is usually performed via shielding of I/O cables and is not considered here.

Therefore, a major guideline in laying out boards is to isolate the I/O circuitry from the high speed logic. This method applied even if the logic is being clocked at "only" a few MHz. Often, the fundamental frequency is of marginal interest, with the harmonics generated from switching edges of the clock being the biggest emission culprits. Internal system input/output PCB circuitry should be mounted as close to the edge connector as possible and capacitive filtering of these lines may be necessary to reduce EMI on the lines.

High speed logic components should be grouped together. Digital interface circuitry and I/O circuitry should be physically isolated from each other and routed on separate connectors, if possible as shown in *Figure 19d*.

- No High Frequency Signals to the Backplane

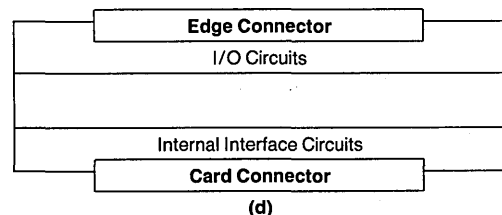
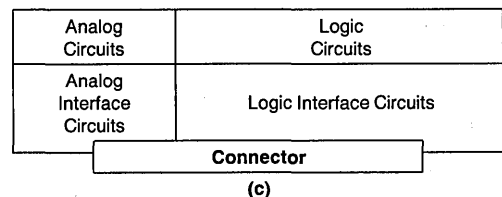
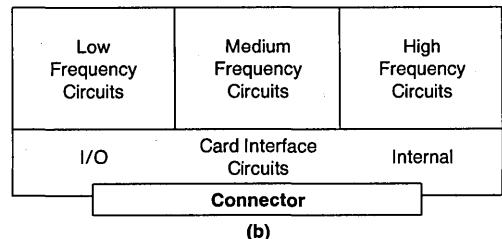
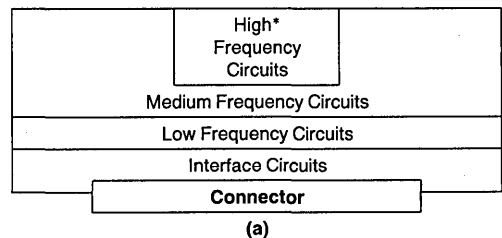
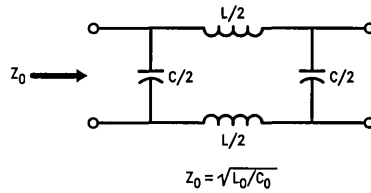


FIGURE 19. Board Layout

Power Supply Bussing

Power supply bussing is the next major concern in the design phase. Isolated digital and analog power supplies must be used when mixing analog and digital circuitry on a board. The design preferably should provide for separate power supply distribution for both the analog and digital circuitry. Single point common grounding of analog and digital power supplies should be performed at one point and one point only—usually at the motherboard power supply input for multi-card designs, or at the power supply input edge connector on a single card system. The fundamental feature of good power supply bussing, however, is low impedance and good decoupling over a large range of frequencies. A low impedance distribution system requires two design features: (1) proper power supply and return trace layout and (2) proper use of decoupling capacitors.

At high frequencies, PCB traces and the power supply buses (+V_{CC} and 0V) are viewed as transmission lines with associated characteristic impedance, Z₀, as modeled in Figure 20. The goal of the designer is to maximize the capacitance between the lines and minimize the self-inductance, thus creating a low Z₀. Table III shows the characteristic impedance of various two-trace configurations as a function of trace width, W, and trace separation, h.



TL/DD/10562-29

where L₀ and C₀ are, respectively, the distributed inductance and capacitance per unit length of the line

FIGURE 20

Any one of the three configurations may be viewed as a possible method of routing power supply (or signal) traces. The most important feature of Table III is the noticeable difference in impedance between the parallel strips and strip over ground plane compared with the side-by-side configurations.

As an example of the amount of voltage that can be generated across the impedance of a power bus, consider TTL logic which pulls a current of approximately 16 mA from a supply that has a 25Ω bus impedance (this assumes no decoupling present). The transient voltage is approximately $dV = 0.016 \times 25\Omega = 400 \text{ mV}$, which is equal to the noise immunity level of the TTL logic. A 25Ω (or higher) impedance is not uncommon in many designs where the supply and return traces are routed on the same side of the board in a side-by-side fashion. In fact, it is not uncommon to find situations where the power supply and return traces are routed quite a distance from each other, thereby increasing the overall impedance of the distribution system. This is obviously a poor layout.

Power and ground planes offer the least overall impedance. The use of these planes leads the designer closer to a multi-layer board. At the very least, it is recommended that all open areas on the PC board be "landfilled" with a 0V reference plane so that ground impedance is minimized.

Multi-layer boards offer a considerable reduction in power supply impedance, as well as other benefits. As shown in Table III, the impedance of a multi-layer power/ground plane bus grows very small (on the order of an ohm or less), assuming a W/h ratio greater than 100. Multi-layer board designs also pay dividends in terms of greatly reduced EMI, and they provide close control of line impedances where impedance matching is important. In addition, shielding benefits can be realized. For high-density, high-speed logic applications, the use of a multi-layer board is almost mandatory. The problem with multi-layer boards is the increased cost of design and fabrication and increased difficulty in board repair.

Decoupling

High-speed CMOS has special decoupling and printed circuit board layout requirements. Adhering to these requirements will ensure the maximum advantages are gained with CMOS devices in system performance and EMC performance.

Local high frequency decoupling is required to supply power to the chip when it is transitioning from a LOW to a HIGH value. This power is necessary to charge the load capacitance or drive a line impedance.

For most power distribution networks, the typical impedance can be between 50 and 100Ω. This impedance appears in series with the load impedance and will cause a droop in the

TABLE III

W/h or D/W	#1 $t = \frac{h}{W}$ Parallel Strips*	#2 $\frac{W}{h}$ Strip Over Ground Plane*	#3 $t = \frac{D}{W}$ Strips Side by Side**
	Z ₀₁	Z ₀₂	Z ₀₃
0.5	377	377	NA
0.6	281	281	NA
0.7	241	241	NA
0.8	211	211	NA
0.9	187	187	NA
1.0	169	169	0
1.1	153	153	25
1.2	140	140	34
1.5	112	112	53
1.7	99	99	62
2.0	84	84	73
2.5	67	67	87
3.0	56	56	98
3.5	48	48	107
4.0	42	42	114
5.0	34	34	127
6.0	28	28	137
7.0	24	24	146
8.0	21	21	153
9.0	19	19	160
10.0	17	17	166
12.0	14	14	176
15.0	11.2	11.2	188
20.0	8.4	8.4	204
25.0	6.7	6.7	217
30.0	5.6	5.6	227
40.0	4.2	4.2	243
50.0	3.4	3.4	255

TL/DD/10562-30

*Mylar dielectric assumed: DC = 5.0 D > nearby ground plane

**Paper base phenolic or glass epoxy assumed: DC = 4.7

$Z_{01} = (377/\sqrt{DC}) \times (h/W)$, for $W > 3h$ and $h > 3t$

$Z_{02} = (377/\sqrt{DC}) \times (h/W)$, for $W > 3h$

$Z_{03} = (120/\sqrt{DC}) \ln(D/W + \sqrt{D^2/W^2 - 1})$ for $W \gg t$

V_{CC} at the part. This limits the available voltage swing at the local node, unless some form of decoupling is used. This drooping of rails will cause the rise and fall times to become elongated. Consider the example presented in *Figure 21* used to help calculate the amount of decoupling necessary. This circuit utilizes an octal buffer driving a 100Ω bus from a point somewhere in the middle.

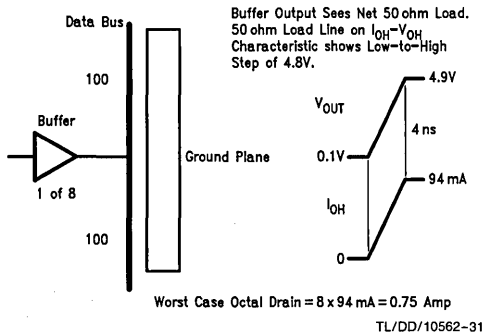
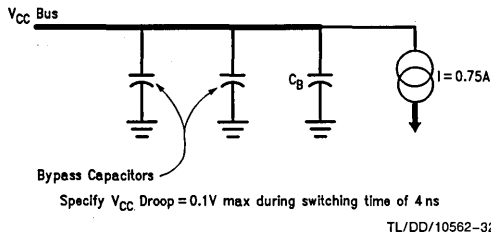


FIGURE 21

Being in the middle of the bus, the driver will see two 100Ω loads in parallel, or an effective impedance of 50Ω. To switch the line from rail to rail, a drive of 94 mA is needed ($4.8\text{V}/50\Omega$) and more than 750 mA will be required if all eight lines switch at once. This instantaneous current requirement will generate a voltage drop across the impedance of the power lines, causing the actual V_{CC} at the chip to droop. This droop limits the voltage swing available to the driver. The net effect of the voltage droop will be to lengthen device rise and fall times and slow system operation. A local decoupling capacitor is required to act as a low impedance supply for the driver chip during high current demands. It will maintain the voltage within acceptable limits and keep rise and fall times to a minimum. The necessary values for decoupling capacitors can be calculated with the formula given in *Figure 22*.

In this example, if the V_{CC} droop is to be kept below 0.1V and the edge rate equals 4 ns, we can calculate the value of the decoupling capacitor by use of the charge on a capacitor equation: $Q = CV$. The capacitor must supply the high demand current during the transition period and is represented by $I = C (dV/dt)$. Rearranging this somewhat yields $C = I (dt/dV)$.



$$Q = CV \text{ charge on capacitor}$$

$$I = C \text{ dV/dt}$$

$$C = I \text{ dt/dV} = 750 \text{ mA} \times 4 \text{ ns} / 0.1\text{V} = 0.030 \mu\text{F}$$

Select $C_B = 0.047 \mu\text{F}$ or greater

FIGURE 22

Now, $I = 750 \text{ mA}$ assuming all 8 outputs switch simultaneously for worst case conditions, $dt =$ switching period or 4 ns, and dV is the specified V_{CC} droop of 0.1V. This yields

a calculated value of $0.030 \mu\text{F}$ for the decoupling capacitor. So, a selection of $0.047 \mu\text{F}$ or greater should be sufficient. It is good practice to distribute decoupling capacitors evenly throughout the logic on the board, placing one capacitor for every package as close to the power and ground pins as possible. The parasitic inductance in the capacitor leads can be greatly reduced or eliminated by the use of surface mount chip capacitors soldered directly onto the board at the appropriate locations. Decoupling capacitors need to be of the high K ceramic type with low equivalent series resistance (ESR), consisting primarily of series inductance and series resistance. Capacitors using 5ZU dielectric have suitable properties and make a good choice for decoupling capacitors; they offer minimum cost and effective performance.

Proper Signal Trace Layout

Although crosstalk cannot be totally eliminated, there are some design techniques that can reduce system problems resulting from crosstalk. In any design, the distance that lines run adjacent to each other should be kept as short as possible. The best situation is when the lines are perpendicular to each other. Crosstalk problems can also be reduced by moving lines further apart or by inserting ground lines or planes between them.

For those situations where lines must run parallel as in address and data buses, the effects of crosstalk can be minimized by line termination. Terminating a line in its characteristic impedance reduces the amplitude of an initial crosstalk pulse by 50%. Terminating the line will also reduce the amount of ringing.

There are several termination schemes which may be used. They are series, parallel, AC parallel and Thevenin terminations. AC parallel and series terminations are the most useful for low power applications since they do not consume any DC power. Parallel and Thevenin terminations experience high DC power consumption.

Series terminations are most useful in high-speed applications where most of the loads are at the far end of the line. Loads that are between the driver and the end of the line will receive a two-step waveform. The first wave will be the incident wave. The amplitude is dependent upon the output impedance of the driver, the value of the series resistor and the impedance of the line according to the formula:

$$V_W = V_{CC} * Z_{oe} / (Z_{oe} + R_S + Z_S)$$

Series Termination



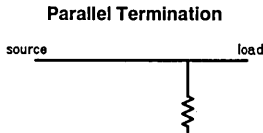
$$V_W = V_{CC} \times Z_{oe} / (Z_{oe} + R_S + Z_S)$$

where R_S is the series resistor
 Z_S is the output impedance of the driver
 Z_{oe} is the equivalent line impedance

The amplitude will be one-half the voltage swing if R_S (the series resistor) plus the output impedance (Z_S) of the driver is equal to the line impedance (Z_{oe}). The second step of the waveform is the reflection from the end of the line and will have an amplitude equal to that of the first step. All devices on the line will receive a valid level only after the wave has

propagated down the line and returned to the driver. Therefore, all inputs will see the full voltage swing within two times the delay of the line.

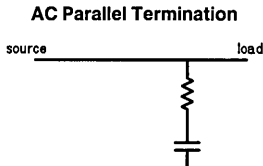
Parallel terminations are not generally recommended for CMOS circuits due to their power consumption, which can exceed the power consumption of the logic itself. The power consumption of parallel terminations is a function of the resistor value and the duty cycle of the signal. In addition, parallel termination tends to bias the output levels of the driver towards either V_{CC} or ground depending on which bus the resistor is connected to. While this feature is not desirable for driving CMOS inputs because the trip levels are typically $V_{CC}/2$, it can be useful for driving TTL inputs where level shifting is desirable in order to interface with CMOS devices.



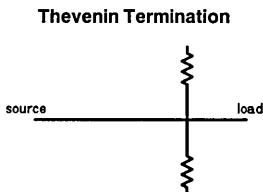
TL/DD/10562-34

AC parallel terminations work well for applications where the increase in bus delays caused by series terminations are undesirable. The effects of AC parallel terminations are similar to the effects of standard parallel terminations. The major difference is that the capacitor blocks any DC current path and helps to reduce power consumption.

Thevenin terminations are not generally recommended due to their power consumption.



TL/DD/10562-35



TL/DD/10562-36

Like parallel terminations, a DC path to ground is created by the terminating resistors. The power consumption of a Thevenin termination, though, will generally be independent of the signal duty cycle. Thevenin terminations are more applicable for driving CMOS inputs because they do not bias the output levels as paralleled terminations do. It should be noted that output lines with Thevenin terminations should not be left floating since this will cause the undriven input levels to float between V_{CC} and ground, increasing power consumption.

Ground Bounce

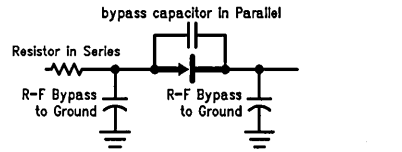
Observing either one of the following rules is sufficient to avoid running into any of the problems associated with ground bounce:

- First, use caution when driving asynchronous TTL-level inputs from CMOS octal outputs. Ground bounce glitches may cause spurious inputs that will alter the state of non-locked logic.
 - Second, use caution when running control lines (set, reset, load, clock, chip select) which are glitch-sensitive through the same devices that drive data or address lines.
- When it is not possible to avoid the above conditions, there are simple precautions available which can minimize ground bounce noise. These are:
- Choose package outputs that are as close to the ground pin as possible to drive asynchronous TTL-level inputs.
 - Use the lowest V_{CC} possible or separate the power supplies.
 - Use board design practices which reduce any additive noise sources, such as crosstalk, reflections, etc.

Components

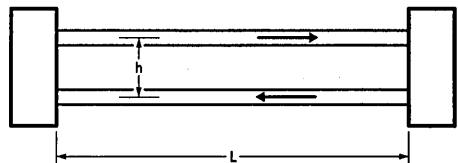
The interference effect by rectifier diodes, typically found in power supply sections of PC boards, can be minimized by one or more of the following measures:

- Placing a bypass capacitor in parallel with each rectifier diode.
- Placing a resistor in series with each rectifier diode.
- Placing an R-F bypass capacitor to ground from one or both sides of each rectifier diode.
- Operating the rectifier diodes well below their rated current capability.



TL/DD/10562-37

Connectors



TL/DD/10562-38

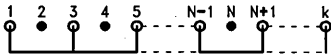
Cables and Connectors

Several options are available to reduce EMI from a typical ribbon cable used to interconnect pieces of equipment. These include:

- Reduce spacing between conductors (h in the figure) by reducing the size of wires used and reducing the insulation thickness.
- Join alternate signal returns together at the connectors at each end of the cable.
- Twist parallel wire pairs in ribbon cables.
- Shield ribbon cable with metal foil cover (superior to braid).
- Replace discrete ribbon cable with stripline flexprint cable.

In the case of joining alternate signal returns, wire N is carrying the signal current, i_n , whereas its mates, N-1 and N+1 wires are each carrying one half of the return currents, i_{n-1} and i_{n+1} , respectively. Thus, radiation from pair N and N-1 is out of phase with radiation from pair N and N+1 and will tend to cancel. In practice, however, the net radiation is reduced by 20-30 dB with 30 dB being a good default value.

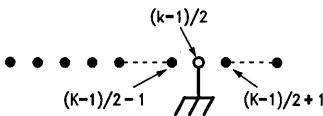
Alternating Signal Returns Minimizes Radiation



TL/DD/10562-39

The opposite of this is to conserve signal returns by only using one, or two, wires to service N data lines in a ribbon cable. For data lines farther from the return line, the differential mode radiation becomes so great that this cable tends to maximize EMI radiation. Another disadvantage of this approach is poor impedance control in the resulting transmission line. This could result in distortion of pulses and cause reflections, especially for high-speed logic, and common return impedance noise in this single ground wire.

Single Signal Return Maximizes Radiation



TL/DD/10562-40

Ideally, connectors should have negligible resistance for obvious reasons other than EMI control. They should provide foolproof alignment to minimize the possibility of contact damage over time and use which would increase the resistance and be prone to vibration and shock. Adequate force to provide good mating between contacts which will insure low resistance and limit likelihood of damage. Connectors should mate with little friction to minimize the effects of continual disconnections and connections increasing the contact resistance with use as the contacts wear out. A contamination free design should be used to avoid corrosion and oxidation increasing resistance and susceptibility to shock and vibration causing intermittent contact.

Special Considerations with Development Tools

The following set of guidelines have been compiled from the experiences of the Development Systems Group and the Microcontroller Applications Group in Santa Clara. They should be considered *additional* techniques and guidelines to be followed concurrently with the standard ones already presented. Some are general and some may be specific to development systems use.

Ground bounce prevention and minimization techniques presented in this Application Note should be strictly adhered to when using '373 type transparent latches on the HPC's external address/data bus. Multiple simultaneously switching outputs could produce ground bounce significant enough to cause false latching. Observe good EMI planning by locating the latches as close to the HPC as possible. The use of multi-layer printed circuit boards with good ground planes and following appropriate layout techniques is

also essential, especially if emulation will be done at frequencies above 10 MHz. With the foregoing discussions about "antenna farms", radiated noise, and ideal connector characteristics, it becomes obvious that wire-wrap boards and the use of IC sockets is absolutely out of the question. The concern here is not so much EMI affecting the outside world but EMI strangling the operation of the module itself.

The inputs to the buffers in a '244 type octal buffer package are placed adjacent or side-by-side outputs of other buffers in the package. This configuration would tend to maximize the crosstalk or noise coupling from the inputs to the outputs. On the other hand, the buffer inputs in a '544 type package are on one side of the package and the outputs are on the other. The use of these package types in high speed designs can facilitate board layout to help reduce the effects of crosstalk.

Use extra heavy ground wires between emulator and target board. Rely on the ground returns in the emulator cable for reduction of differential-mode noise radiated from the cable but heavy-duty help is required for reducing power line impedance in the integrated development system.

Unused HPC inputs, most importantly NMI and RDY/HLD, must be tied to V_{CC} directly or through a pull-up resistor. This not only tends to reduce power consumption, but will avoid noise problems triggering an unwanted action.

In order to reduce the effects of noise generated by high speed signal changes, a sort of Frequency Management technique might be applied. If possible, develop application hardware and software at a slower crystal operating frequency. If ringing, crosstalk, or other combinations of radiated and conducted noise problems exist, the result may be to move the problem from one point in the affected signal waveform to a different point. Thus, apparent "noise glitches" that caused a latch to erroneously trigger when the input data was still changing, may now come at a time when they are non-destructive such as at a point when the input data is now stable.

Some applications require driving the HPC clock input, CKI, with an external signal. The emulator tools are all clocked using a crystal network with the HPC so that the generation of the system timing is contained on the tool itself. Consequently, there is no connection between the emulator cable connector on the tool and the CKI pin at the HPC. However, when the emulator cable is now inserted into the target board, the target board's clock signal travelling along the cable couples noise onto adjacent signal lines causing symptoms pointing to an apparent failure of the emulator tool. The recommendation is to disable the clock drive to the CKI pin at the HPC pad on the target board whenever the emulator tool is connected. The emulator tools supply the system clock so there is no need for the clock on the target and signal crosstalk on the emulator cable can be greatly reduced with minimal implementation. If one insists that the emulator tool and the target be synchronous, then bring the clock signal from the target to the emulator tool external to the emulator cable via twisted wire pair or coax cable. Remove the clock drive connection to CKI at the target to prevent the signal from entering the cable. Finally, remove crystal components on emulator tool to prevent problems with the signal.

Connecting boards and modules together to make a totally unique system in which EMC was practiced is necessary to ensure little problem with the environment. But, connecting

an emulator tool makes it an entirely new and unique system, both in physical and electrical properties. Treat the emulator tool as part of the system during the design phase and development phase.

NOISE MEASUREMENT

The basic purpose of FCC Part 15J is to minimize the jamming of commercial broadcasting systems by computer devices. Toward this end, the FCC has established test limits, for both conducted and radiated emissions, which must be met. These two tests together span the frequency range from 450 kHz to 1000 MHz. To accomplish FCC Part 15J testing requires the following equipment and associated support items:

- EMC Receivers or Spectrum Analyzers to cover the frequency range from 450 kHz to 1000 MHz.
- Dipole antennas (2) to cover the frequency range from 30 MHz to 1000 MHz.
- Masts or supports which will allow antenna elevation to be increased to at least 4 meters and also allow the polarization to be changed.
- Line impedance stabilization networks (LISN) built in accordance with CISPR requirements. These are 50 Ω , 50 μ H devices and are inserted between power mains and test item to permit making repeatable conducted EMI measurements.
- Power line filters.
- An appropriate test site.

Environment

The most controversial item on the test requirement list is the appropriate test site. The FCC required emission limits are comparable with the ambient RF level. These low limits and the noisy ambient would indicate that the tests should be made in a shielded enclosure. Unfortunately, all shielded enclosures introduce significant errors into the radiated measurements because of room reflections, room resonances, and antenna loading. To reduce the magnitude of these problems, the FCC has specified that measurements should be made at an open-field test site. Open-field test sites frequently have high ambient levels especially in the FM broadcast band. They may also have ground reflection variations as a function of soil moisture.

The FCC will permit the use of anechoic shielded enclosures which have reduced reflections, provided an error analysis is made to show correlation of interior RF levels with those of an open-field test site. The cost of an anechoic enclosure is its major drawback. For measurements other than for certification, the test site does not have to be in accordance with government regulations. There are also alternatives where an agency or private company will perform the tests for you at their facility for a nominal fee.

Many manufacturers are using shielded enclosures that they have constructed on site or purchased from one of the shielded enclosures manufacturers. The measurement requirement is that the RF ambient levels should be 6 dB or more below the specifications limits. This may require 20 dB worth of aluminum foil or 160 dB worth of electrical seals. Only a site survey can provide that answer. In any case, some margin of safety should be made, 6–10 dB, plus periodic check for reflection problems.

Instrumentation

After the appropriate test site has been obtained, whether a room or a quiet open field, then the testing can begin. If the

equipment to be tested is not floor standing, the test sample is placed on a non-conducting stand 80 cm high and at least 40 cm from the wall of the enclosure. Antennas are then set up so that radiated emission levels can be measured. The test sample should be loaded with full electrical and mechanical loads and operated in a manner that closely approximates normal operation. During operation of the equipment under test, the EMI measuring equipment is used to determine the amplitude of the radiated emission.

At NSC, we have a spectrum analyzer that can be attached to a Personal Computer that runs software to control experiments and report results. It automatically marks the computer display with FCC limits for quick comparison with the amplitude of the emissions signal. This setup is outside the shielded enclosure and can be used to determine if the equipment under test is failing any FCC requirements.

If the test sample fails, we can move inside the room and use near-field probes to help pinpoint the source of emissions. The spectrum analyzer samples the signal generated by the source at many different frequencies. The scale across the bottom of the screen is frequency and the scale along the side is signal amplitude in dBuV/m. Thus, we can quickly determine where the peak amplitude of the generated noise is located, read what level that is, and at what frequency it is being generated.

A little analysis and thought should then allow you to determine what signal could be the culprit. For example, if the noise problem is at 16 MHz and the system clock is 16 MHz, then the basic clock signal is causing the problem. If the noise problem is at even multiples of 16 MHz it could be caused by rise and fall times on the 16 MHz clock or overshoot and undershoot on that clock. In the case of the HPC, since it generates a clock output that is the system clock divided by 2 (CK2 = CK1/2), the noise frequency generated at the multiple of the 16 MHz signal could also be due to CK2 or any device that is clocked by that signal. Unfortunately for the investigator, everything else inside the part is clocked by CK2, which includes bus transitions and input sampling.

Cost

Basically, the risks of no EMI control will include the following:

- Vehicle/System Performance Degradation
- Degradation to outside world equipment
- Personal Hazards
- Ordinance Hazards
- Acceptance Delays

The sum which can mean anything from a minor system or equipment performance compromise to the total cancellation of a project.

The cost of EMI control will vary and include the following:

- Government procurement requirements
- Company proposal preparation
- EMI Control Plan
- Test Plan
- EMI Tests and Reports

A rough guideline that can be used might be:

- 1%–3% of \$100 Million projects
- 3%–7% of \$1 Million to \$10 Million projects
- 7%–12% of small items

SUMMARY

The design and construction of an electromagnetically compatible printed circuit board does not necessarily require a big change in current practices. On the contrary, the implementation of EMC principles during the design process can fit in with the ongoing design. When EMC is designed into the board, the requirements to shield circuitry, cables, and enclosures, as well as other costly eleventh hour surprises, will be drastically reduced or even eliminated. Without EMC in the design stage, production can be held up and the cost of the project increases.

REFERENCES

1. Atkinson, Kenn-Osburn, John-White, Donald, *Taming EMI in Microprocessor Systems*, **IEEE Spectrum**, December 1985, pp 30-37
2. Balakrishnan, R. V., *Reducing Noise on Microcomputer Buses*, **National Semiconductor Application Note 337**, May 1983
3. Brewer, Ron W., *Test Methodology to Determine Levels of Conducted and Radiated Emissions from Computer Systems*, **EMC Technology**, July 1982, p 10
4. Engineering Staff, Don White Consultants, Inc., *The Role of Cables & Connectors in Control of EMI*, **EMC Technology**, July 1982, pp 16-26
5. Fairchild Semiconductor, *Design Considerations, Fairchild Advanced CMOS Technology Logic Data Book*, 1987, pp 4-3 to 4-12
6. Fairchild Semiconductor, *Understanding and Minimizing Ground Bounce*, **Application Note DU-6**, August 1987
7. Interference Control Technology, *Introduction to EMI/RFI/EMC*, **Course Notebook on Electromagnetic Compatibility**, August 1988
8. Violette, Michael F. and J.L., *EMI Control in the Design and Layout of Printed Circuit Boards*, **EMC Technology**, March-April 1986, pp 19-32
9. Violette, Michael F., *Curing Electromagnetic Interference*, **Radio-Electronics**, November 1985, pp 50-56
10. Wakeman, Larry, *High-Speed-CMOS Designs Address Noise and I/O Levels*, **National Semiconductor Application Note 375**, September 1984
11. White, Donald R. J., *EMI Control Methods and Techniques*, **Electromagnetic Interference and Compatibility — Vol 3**, copyright 1988 by Interference Control Technologies

Interfacing the HPC46064 to the DP83200 FDDI Chip Set

National Semiconductor
Application Note 736
Simon Stanley



AN-736

TABLE OF CONTENTS

- 1.0 INTRODUCTION
- 2.0 FDDI INTELLIGENT STATION ARCHITECTURE
- 3.0 DP83200 FDDI CHIPSET
- 4.0 HPC46064 HIGH PERFORMANCE MICROCONTROLLER (HPCTM)
- 5.0 CONTROL BUS INTERFACE

1.0 INTRODUCTION

Fiber Distributed Data Interface (FDDI) is a high bandwidth (100 Mbits per second) local area network (LAN) which uses a dual redundant ring architecture. The network consists of a number of point to point links connected to form a ring. The physical and electrical characteristics of the ring protocol are covered by the Physical Media Dependent (PMD), Physical (PHY) and Media Access Control (MAC) Standards as defined by the American National Standards Institute (ANSI) X3T9.5 committee, and can be implemented using the DP83200 FDDI chipset from National Semiconductor. The on-line verification of these point to point links, and the formation of a ring is covered by the FDDI Station Management (SMT) Standard.

This application note covers the use of the HPC46064 High Performance Microcontroller from National Semiconductor to provide the local processing power required within a Fiber Distributed Data Interface (FDDI) node for Station Management (SMT). The note covers the interface between the HPC and the FDDI chipset from National Semiconductor and a possible system architecture.

2.0 FDDI INTELLIGENT STATION ARCHITECTURE

In FDDI, the Station Management (SMT) service is split into three main sections, SMT Frame Services, Ring Management (RMT) and Connection Management (CMT). Within the National Semiconductor implementation of FDDI, any controller handling RMT and CMT services need only access the Control Bus directly, although a communications channel to the host is also required. An architecture using the HPC from National Semiconductor is shown in *Figure 1*. This can be implemented directly using the interface shown in *Figure 2*. Using this architecture, the SMT frame services are provided by the host.

The architecture shown is one of several that could be implemented with the HPC and the National FDDI chipset. This architecture connects the HPC multiplexed address/data bus to the control bus of the FDDI devices, allowing the HPC to access these devices directly with single instructions. The 16 Kbyte ROM of the HPC46064 is used to minimize chip count, though this architecture allows external ROM or RAM to be used if additional functions are implemented.

The HPC has sufficient performance to handle all of SMT, including frame based management, but this would require that the HPC have access to the frame buffer memory. This requires an arbitration scheme to resolve conflicts between the host and the HPC in accessing the buffer RAM. The arbitration scheme could be simply implemented using the HOLD and HLDA (Hold Acknowledge) pins of the HPC.

An advantage of putting all of SMT on the board is that the SMT function need not be resident in the host memory. The removal of TSRs or daemons from host memory leaves more room for applications.

An alternative architecture would run the HPC in single chip mode, reducing the chip count and allowing the Universal Peripheral Interface (UPI) port to be used as the host interface. This would require software drivers to simulate the FDDI control bus signals using the I/O ports of the HPC.

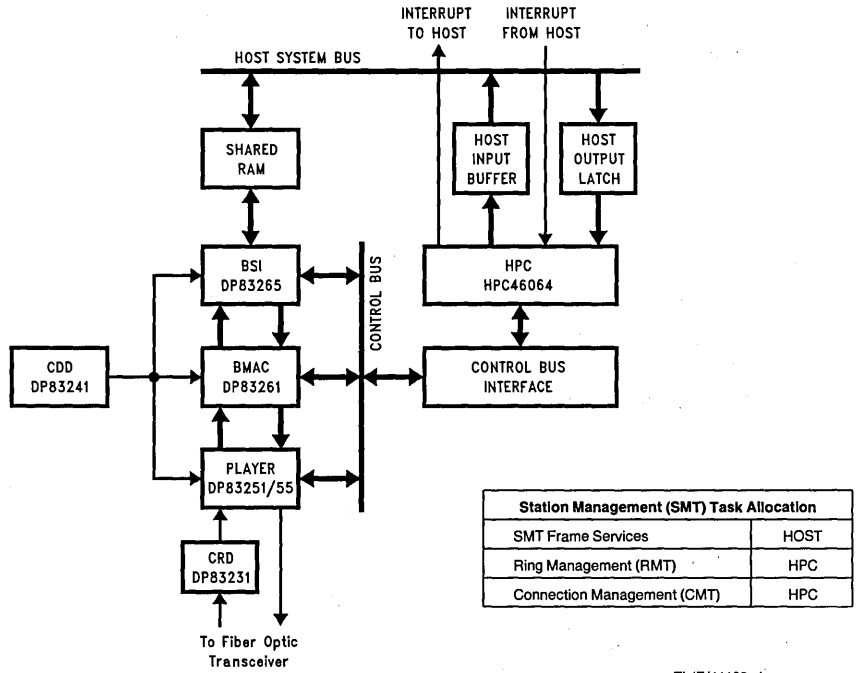


FIGURE 1. FDDI Station Architecture Using HPC for Partial Station Management

TL/F/11103-1

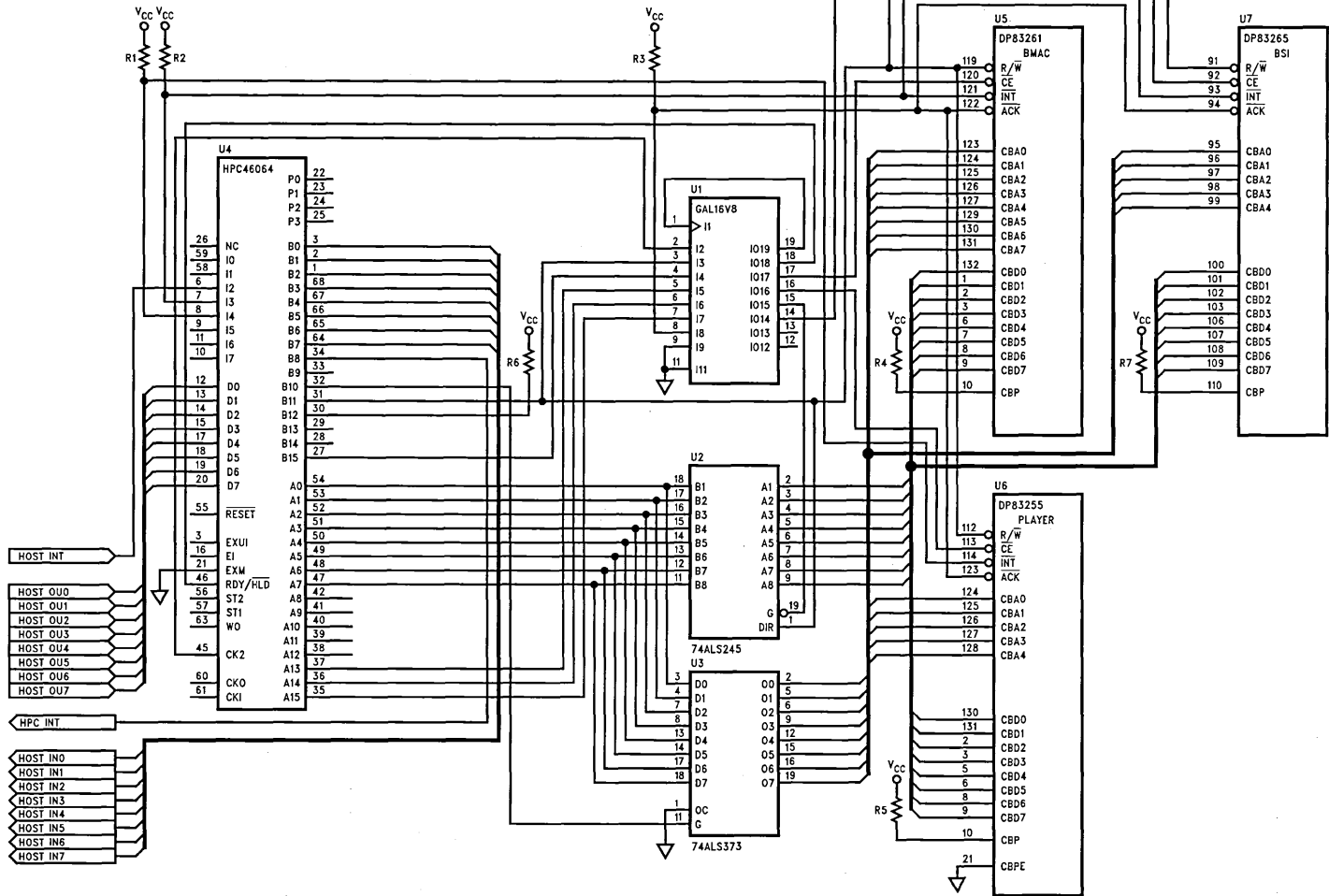


FIGURE 2. HPC to FDDI Chipset Interface

3.0 DP83200 FDDI CHIPSET

The DP83200 FDDI chipset from National Semiconductor implements the PHY and MAC Standards as defined by the American National Standards Institute (ANSI) X3T9.5 committee. The chipset includes the following devices.

DP83231 Clock Recovery Device (CRD™ DEVICE)

The Clock Recovery Device extracts a 125 MHz clock from the incoming data of the upstream station. Its features include on-chip loopback control, on-chip PLL, ability to lock to a Master Line State in less than 100 μ s, and a single +5V supply.

DP83241 Clock Distribution Device (CDD™ DEVICE)

From a 12.5 MHz reference, the CDD generates the 125 MHz, 25 MHz and 12.5 MHz clocks required by the PLAYER™ and BMAC™ devices.

DP83251/DP83255 Physical Layer Controller (PLAYER™ DEVICE)

The PLAYER device converts the BMAC 12.5 Mbyte/s stream into a 125 Mbaud 4B/5B encoded bit stream as specified in the FDDI PHY Standard. It synchronizes the received bit stream to the local 12.5 MHz clock and decodes the 4B/5B data into internal code. The DP83255 PLAYER device also contains a configuration switch for use in dual attachment stations and concentrators.

DP83261 Basic Media Access Controller (BMAC™ DEVICE)

The BMAC implements the functions defined by ANSI X3T9.5 FDDI Media Access Control Standard. The BMAC consists of the transmit and receive state machines, an address magnitude compare unit, a CRC checker, a CRC generator, protocol timers and diagnostic counters.

DP83265 BMAC System Interface (BSI™ DEVICE)

The BSI device provides a multiframe, multiple channel interface between the BMAC device and a host system. The BSI device interfaces directly to the system bus or through low-cost DRAMs. The efficient data structures employed provide high throughput with minimal host intervention.

For more information on these and other devices in the chipset please consult the appropriate data sheets and application notes.

4.0 HPC46064 High Performance Microcontroller (HPC DEVICE)

The HPC46064 is a member of the HPC family of High Performance Microcontrollers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC46064 has 16 Kbytes of on-chip ROM and is fabricated in National's microCMOS technology. This process combined with the advanced architecture provides fast, flexible I/O, efficient data manipulation, and high speed computation.

The HPC46064 features include 16 bit internal architecture, 16- or 8-bit external data bus, 16 bit address and up to 52 general purpose I/O lines. The HPC46064 also includes a full duplex UART, four 16-bit timers, 16 Kbytes of ROM and 512 bytes of RAM.

The HPC46064 is used in this design because its 16 Kbytes of on-chip ROM remove the need for an external EPROM.

For prototyping or small production runs the pin compatible HPC467064 may be used, which has 16 Kbytes of on-chip EPROM.

Devices in the HPC family can directly address 64 Kbytes of external memory. This address range can be expanded to over 500 Kbytes without additional devices by using I/O pins to implement a simple bank switching technique.

This additional addressing range may be needed if the HPC implements the whole SMT function. The HPC development tools support this bank switching technique, allowing this low-cost microcontroller to handle the whole SMT function while minimizing the software development effort.

The performance of the system will not be affected by using this bank switching approach because many of the SMT functions are never performed simultaneously. The HPC could, for example, run the Physical Connection Management (PCM) from one memory bank, then switch to the bank containing the frame-based management functions at its leisure.

5.0 CONTROL BUS INTERFACE

The FDDI chipset from National Semiconductor provides the PHY and MAC services as detailed in the FDDI Standard. The chipset does not provide the Station Management (SMT) services, but does provide access to all the necessary data through a large array of 8-bit registers located on-board the PLAYER, BMAC, and BSI devices. The interface to these registers is via the Control Bus defined in the data sheet for the BMAC Device (Media Access Controller). The interface consists of an 8-bit address bus, 8-bit data bus and several control lines. The control lines consist of a read/write signal, a chip enable signal and an acknowledge signal. There is also an interrupt signal and a parity bit, which for this application, has been disabled. The interrupt signal and the acknowledge signal are open drain signals and can be wire-ORed.

A transfer cycle on the bus is started when the processor drives the Chip Enable (\overline{CE}) signal low. Within 20 ns, the data (write cycle only), address and read/write signals must all be valid. The device being accessed will respond by driving the Acknowledge (\overline{ACK}) signal low when data is valid (read cycle) or when the data has been clocked in (write cycle). The processor can now end the transfer by driving \overline{CE} high and the device will complete the cycle by driving \overline{ACK} TRI-STATE®.

The HPC provides a multiplexed 16 bit bus for interfacing to external memory. As only 8 bits are required for the Control Bus, the interface can be achieved with minimal components (see Figure 2). The multiplexed address is latched by the flow-through latch, U3. The data bus is buffered by the bi-directional buffer, U2, which is enabled whenever either read or write signals is asserted by the HPC. The direction is controlled by the Write (WR) signal. Two chip enables are provided (PCE and BCE) by decoding the read, write and upper address bits. The chip enables are delayed until the next Clock (CK2) falling edge so that during a write cycle, the data is available within 20 ns of the chip enable. The Ready (RDY) signal forces the HPC to insert wait states until an \overline{ACK} is received from the Control Bus. (See Figures 3 and 4 for interface waveforms.)

No polling of the Control Bus is required as the interrupt signals for each device are brought into the HPC separately. A simple host interface is provided consisting of two interrupt signals and two eight bit ports, one for read and one for write.

Device U1 is a GAL16V8 (a programmable logic device) and performs all of the logic functions required to generate the control signals used by this system. The programming of this device using the ABEL language is straightforward, as shown by Figure 5.

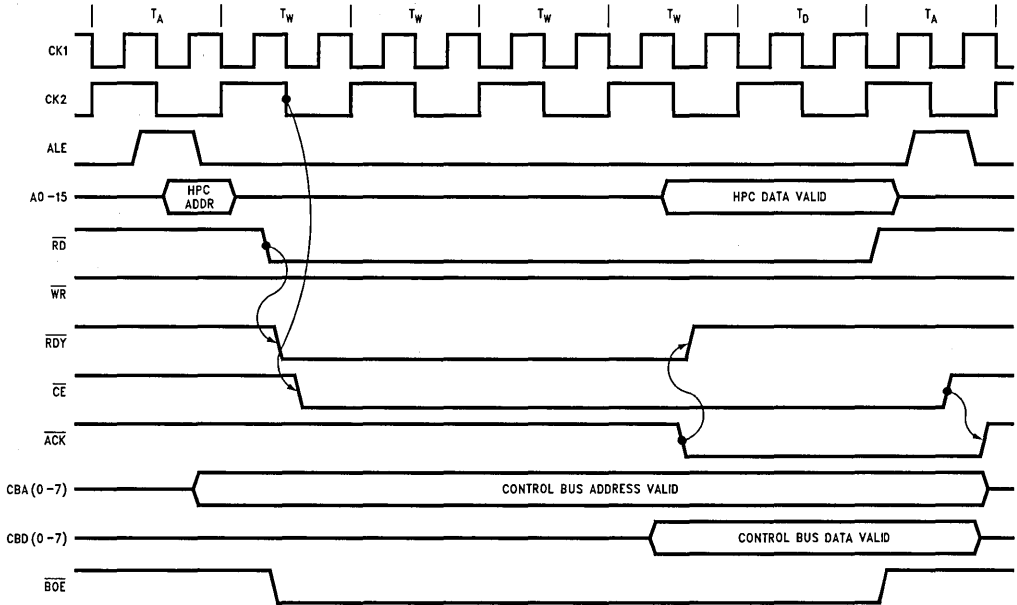
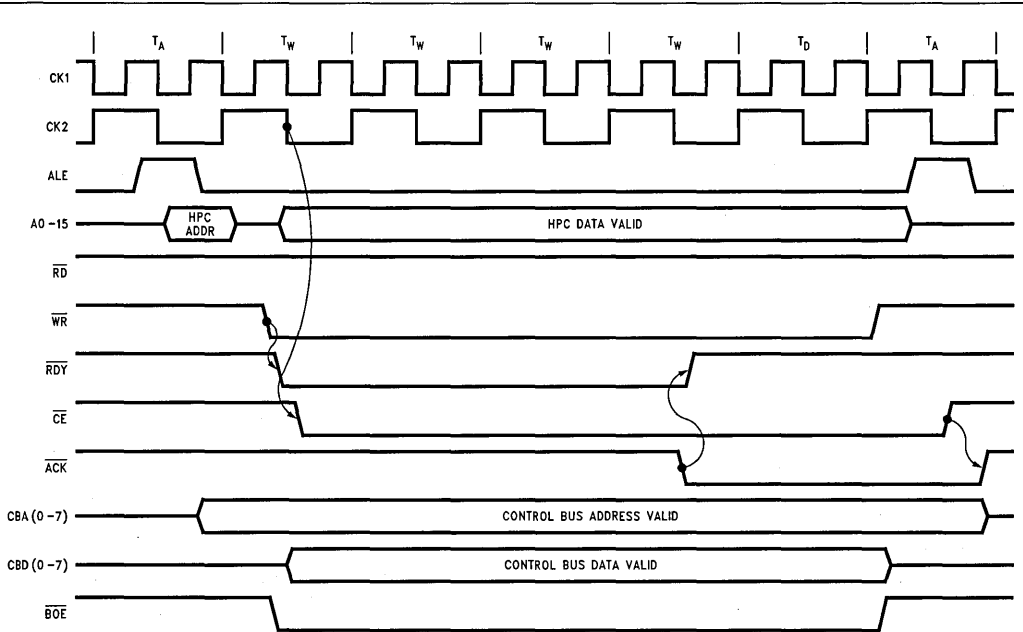


FIGURE 3. Control Bus Interface READ Cycle

TL/F/11103-3



TL/F/11103-4

FIGURE 4. Control Bus Interface WRITE Cycle

Title 'HPC to C Bus Interface

Simon Stanley

National Semiconductor 12 Dec. 1990

U01

device 'GAL16V8';

CK2,WR,RD,ACK

pin 2,3,4,8;

A15,A14,A13

pin 5,6,7;

RDY,NCK2,PCE,BCE, BSICE, BOE

pin 18,19,16,17,14,15;

Address = [A15,A14,A13,x,x,x,x,x,x,x,x,x,x,x,x];

equations

!PCE := (!RD# !WR) & (Address >= h8000) & (Address <= h9FFF);

!BCE := (!RD# !WR) & (Address >= hA000) & (Address <= hBFFF);

RDY = (WR & RD)# !ACK;

!BOE = !WR# !RD;

NCK2 = !CK2;

FIGURE 5. ABEL Source File for HPC FDDI Interface PAL

LCD Direct Drive Using HPC

National Semiconductor
Application Note 786
Santanu Roy



AN-786

INTRODUCTION

Liquid Crystal Displays (LCD) are used in a wide variety of applications. They are extremely popular because of their low power consumption. Manufacturers of Automobiles to Measuring Equipment have taken advantage of these low power displays. Driving LCDs has always been done with dedicated driver chips which not only increase the system cost, but also increase the chip count and board space. This note is developed to demonstrate a low cost solution using the HPC to directly drive LCDs without any driver interface in applications involving LCD display control. A customized 2-way multiplexed LCD (I3420) is being used to illustrate the above capability of HPC microcontrollers in the form of a simple decimal counter.

DRIVING AN LCD

An LCD consists of a backplane and any number of segments which will be used to form the image being displayed. Applying a voltage (nominally 4V–5V) between any segment and the backplane causes the segment to darken. The only catch is that the polarity of the applied voltage has to be periodically reversed, or else a chemical reaction takes place in the LCD which causes deterioration and eventual failure of the liquid crystal. (DC components higher than 100 mV can cause electrochemical reactions in LCDs). To prevent this from happening, the backplane and all the segments are driven with an AC signal, which is derived from a rectangular waveform. To turn a segment *OFF*, it is driven by the same waveform as the backplane. Thus it is always at backplane potential. If a segment is to be *ON*, it is driven with a waveform that is the inverse of the backplane waveform. Thus it has periodically changing polarity between it and the backplane.

MULTIPLEXED LCDs

Today a wide variety of LCDs ranging from static to multiplex rates of 1:64 are available on the market. The *MULTIPLEX* rate of an LCD is determined by the number of backplanes. The higher the multiplex rate the more individual segments can be controlled using only one line e.g., a static LCD has only one backplane and hence only one segment can be controlled using one line. A two way multiplexed LCD has two backplanes and two segments can be controlled with one line. In general if the multiplex ratio of the LCD is N and the number of available outputs is M, the number of segments that can be driven is:

$$S = (M - N) * N$$

i.e., N lines out of M outputs will be used to drive N backplanes, the rest (M – N) outputs are available for segment control. Each line can control N segments, so (M – N) lines can drive (M – N)*N segments. So the maximum number of segments in a 2-way MUX LCD that can be driven with an HPC (if all outputs—16 PortA, 16 PortB, and 4 PortP are used) is:

$$S = (36 - 2) * 2 = 68$$

The number of backplanes in the LCD also determines the number of levels to be generated for their control signals,

e.g., three different voltage levels V, 1/2V, and 0 are to be generated for a 1:2 LCD device (V = operating voltage of the LCD). A *Refresh Cycle* of LCDs (also known as "*Scan Frequency*") is the time period during which all backplanes and segments have to be updated. Typically this is between 39 Hz–208 Hz. During each half of the refresh cycle (*Frame Time*), the polarities of the voltages driving the backplanes and the segments are reversed because of the reason stated above. The current consumption of typical LCDs is in the range of 3 μ A–4 μ A (at V = 4.5, refresh rate 60 Hz) per square centimeter of activated area. Thus the backplane and segment terminals can be treated as Hi-Z loads. At high refresh rates the current consumption of LCDs increases dramatically, a reason why many LCD manufacturers recommend not to exceed a refresh rate of 60 Hz.

LCD CONTROL AND HPC

Figure 1 shows the schematic of the system. With the HPC, each I/O pin can be set individually to TRI-STATE®, "HI" or "LO". Here, in this application, B4 and B5 on the HPC's PortB are selected for backplane control of a 1:2 multiplexed customized LCD—I3420. The three different voltage levels viz. V, V/2, and 0 required for backplane control are achieved through an external voltage divider circuit. The procedure is to set B4 and B5 to "LO" for 0, Hi-Z (configuring them as inputs) for 0.5V, and "HI" for V at the backplane electrodes. For segment control: 8 PortA lines (A0–A7), 4 PortP lines (P0–P3) and 3 PortB lines (B0–B2) are used. All are used as outputs to drive individual segments of the LCD. The HPC in this application is used in single-chip mode to maximize the I/O pin count for LCD control.

TIMING CONSIDERATIONS

Figure 2 shows the backplane and segment waveforms of a typical 1:2 multiplexed LCD. One Refresh Cycle T_{scan} is subdivided into four equally spaced time slots t_a , t_b , t_c and t_d during which the backplane and segment terminals have to be updated in order to switch a specific segment "ON" or "OFF". The voltage waveform during BP– is the mirror image of the waveform during BP+ which satisfies polarity reversal every T_{frame} . Considering a refresh frequency of 50 Hz i.e., $T_{scan} = 20$ ms: t_a , t_b , t_c and t_d are each equal to 5 ms. The timer T2 is used to mark off one time phase ($1/4$ of T_{scan}) of the driving voltage waveform. The timer and autoreload value to get 5 ms time-out is 4999 (decimal) at an operating frequency of 16.0 MHz.

SEGMENT CONTROL

In *Figure 2a*, BP1 and BP2 are the typical backplane waveform of a 2-way multiplexed LCD. During BP+ time, backplane outputs are *ON* for driving voltage level V and *OFF* for the level $1/2$ V. Again for BP– frame time, backplane outputs are *ON* for "0" and *OFF* for " $1/2$ V". Voltage at a particular LCD segment is the resultant of the backplane output and voltage at the line driving that segment. *Figure 2(b)* shows the waveform at an LCD segment. *Figure 2(c)* and *2(d)* are the resultant waveforms with respect to BP1 and BP2 obtained by subtracting the segment waveform in *Figure 2(b)* from the backplane waveforms BP1 and BP2 respectively.

Figure 3 shows the four different waveforms which must be generated at the segments to meet all possible combinations ON and OFF sequence viz. OFF-OFF, ON-ON, ON-OFF, and OFF-ON. A segment is ON if the resultant voltage across it periodically oscillates between +V and -V and is OFF if the swing is between +V/2 and -V/2. The result of the combination is showed in form of white and black circles, representing OFF and ON segments respectively e.g., a waveform pattern "1" will always turn a segment OFF with respect to both the backplanes. However, the waveform "2" will keep it ON with respect to BP1 and BP2. Figurea 4a and Figure 4b show the resultant voltage waveforms at an LCD segment for the above possible combinations and the status of the segment during display operation. Figures 5 and 6 shows the internal segment and backplane connections for a typical 2-way LCD. Figure 7 gives the details of the LCD used in this application.

LCD DRIVE SUBROUTINE

The software for the LCD drive is provided at the end of this application note. The drive subroutine DISPL converts a 16-bit binary value to a 20-bit BCD value for easier display data fetch. This subroutine itself is comprised of a main routine for backplane refresh and seven subroutines (SEGTA, SEGTB, SEGTC, SEGTD, SEGOUT, TMPND, and DISPD). The subroutines SEGTA through SEGTD are used to fetch LCD segment data from a lookup table in ROM for time phases ta, tb, tc, and td respectively. In the table, the subroutine SEGOUT writes these data for each time phase to the respective ports of the HPC connected to the LCD

device. For a refresh cycle of 50 Hz (20 ms), each time phase (1/4 of T_{scan}) is equal to 5.0 ms. This time base is generated by the HPC timer T2 with the associated autoreload register R2. The polling routine TMPND checks for timer underflow flag at the end of each time phase. If the flag is set, it is reset and the program returns to the calling routine. This way a 5 ms time delay is created before the segment and backplane data for the next time phase is updated. The DISPD subroutine switches the LCD OFF by driving the segment and backplane ports to logic "LO". In this application, the display is initialized with "399.9" (which uses all LCD segments) for a BCD down counter. Each count is displayed for a fixed period of time (here a present time of 100 ms is chosen) which is user programmable. The special segments e.g., "m", "A", "V" ... etc. which are not used are all connected together to a common port pin (B2) of the HPC and kept turned OFF throughout the display. It is mandatory to drive any unused segment lines to the OFF state rather than leaving them open or grounded which might result in ghost images.

Note: Selecting the resistors for the voltage divider circuits on B4 and B5 will depend on the type of LCD used.

TYPICAL APPLICATIONS

- Automotive test and control systems
- Weighing scales
- Control Panel
- Microwave
- Clocks and watches etc.

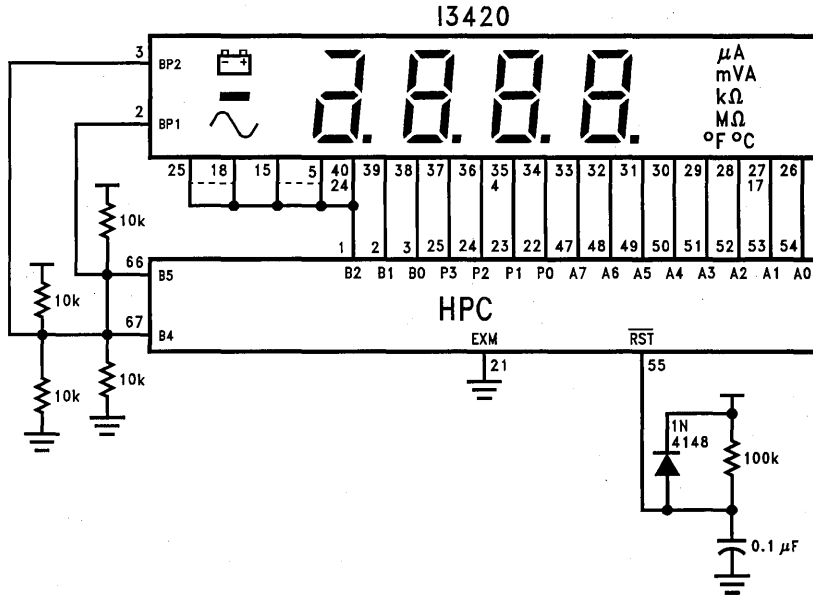
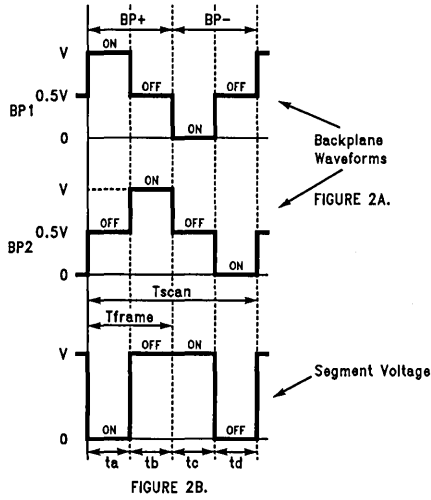


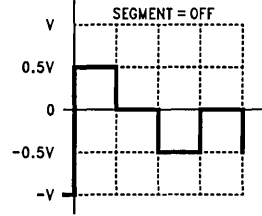
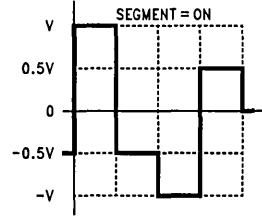
FIGURE 1

TL/DD/11250-1

LCD Waveforms

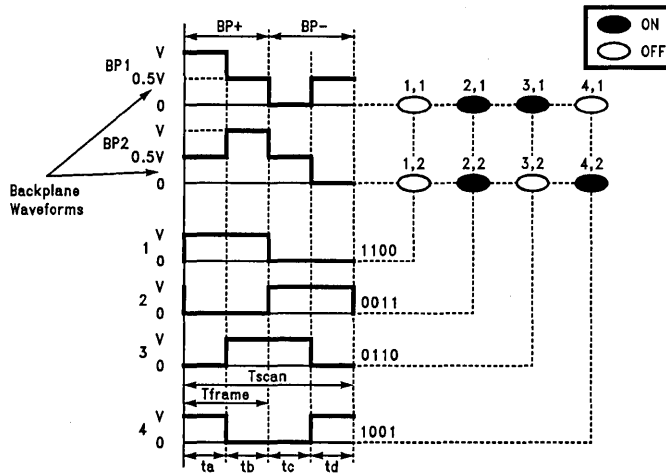


TL/DD/11250-2



TL/DD/11250-3

Segment and Backplane Waveforms



TL/DD/11250-4

Resultant Waveforms at Segments

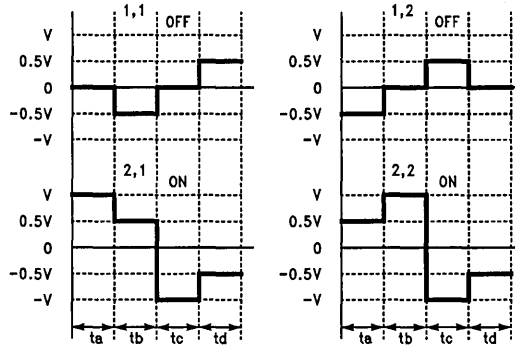


FIGURE 4a

TL/DD/11250-5

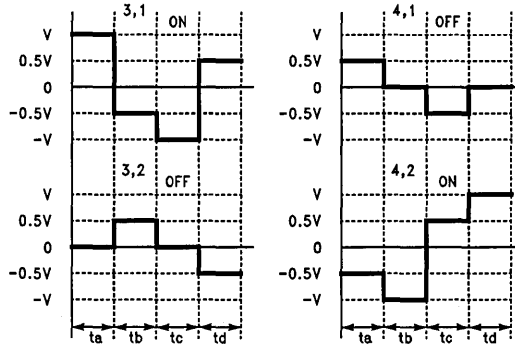


FIGURE 4b

TL/DD/11250-6

Segment and Backplane Distribution

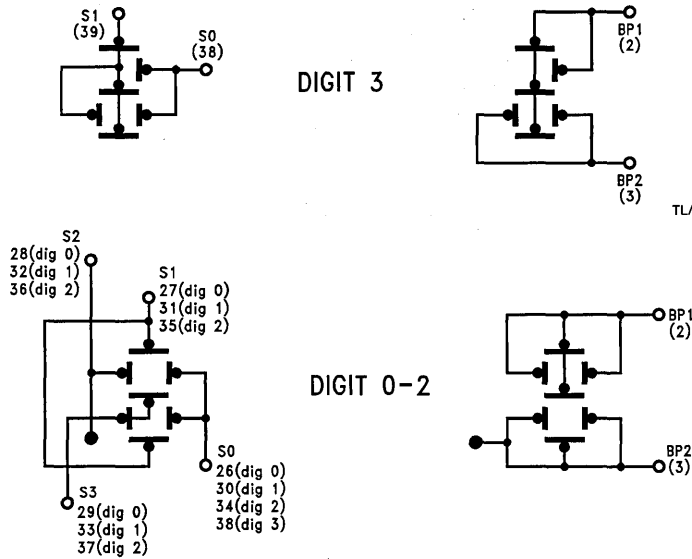
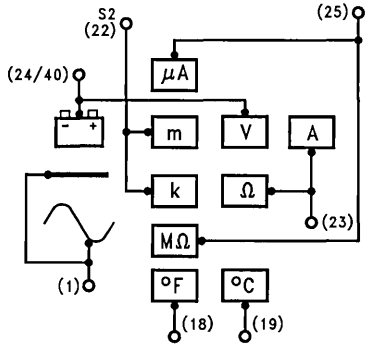


FIGURE 5

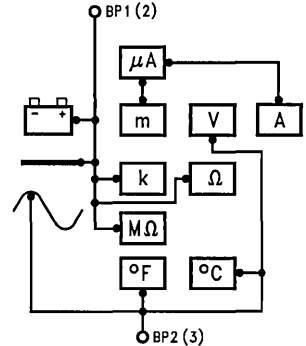
TL/DD/11250-7

TL/DD/11250-8

Special Segments
Segment and Backplane Distribution



TL/DD/11250-9



TL/DD/11250-10

FIGURE 6

```
;MEMAP.INC
```

```
;This is the memory map of different RAM areas used in the
;LCD program.
```

```
;***** RAM DEFINITIONS *****
```

```
BCDLO  = 02:b           ;Measured period in BCD (lo byte)
BCDHI  = 03:b           ;High Byte

MWBUF0 = 05:b           ;A-port data (7-segment)
MWBUF1 = 06:w           ;P-port data
MWBUF2 = 08:b           ;B-port data

OFF1   = 0a:b           ;offset reg. for 7-seg code table
OFF2   = 0b:b           ;
OFF3   = 0c:b           ;

EVAL   = 0e:w           ;end value lo-byte (period)
SVAL   = 010:w          ;hi-byte

COUNT = 020:b          ;counter #1
COUNT2 = 021:b         ;counter #2
BCNT   = 022:w
```

```
;***** BIT MAP *****
```

```
BP1    = 05             ;Backplane 1
BP2    = 04             ;Backplane 2
```

TL/DD/11250-12

```
;File Name:CNTR.ASM
;Function: Counter displayed on a 2-way muxed LCD display
;directly driven by the HPC16083.
```

```
.includ reg16083.inc           ;HPC register def. file
                               ;Chip - HPC16083

.includ memap.inc

.extrn DISPL,DISPD,COPY

.sect cntr,rom16

BEGIN:
    ld    sp,#01c0             ;set the stack pointer
BINIT:
    jsr   DISPD                ;define port config,
                               ;switch diplay OFF
OK1:
    ld    BCNT,#0f9f          ;set counter to 3999 decimal
BLOOP:
    ld    b,#BCNT             ;copy the decimal value
    ld    x,#EVAL             ;to the location which
    jsr   COPY                ;undergoes conversion
    ld    COUNT2,#01          ;display time=100 ms
    jsr   DISPL               ;Display 399.9 first

BLOOP2:
    decsz BCNT                ;display till
    jp    BLOOP               ;counter=0
    ld    b,#BCNT             ;display "0" also
    ld    x,#EVAL             ;and then restart
    jsr   COPY                ;the session
    ld    COUNT2,#01          ;
    jsr   DISPL               ;go back and start
    jp    BINIT

.endsect
.end BEGIN
```

TL/DD/11250-13

```

;Title: DISPL.ASM

;COUNT2 = Contains display time in seconds e.g if "1" ->
;display time is 1 second.

;SEGTA: Gets LCD segment data for time phase Ta
;SEGTB: .... Time phase Tb
;SEGTC: .... Time phase Tc
;SEGTD: .... Time phase Td

;OFF1: .... Offset register for DIGIT 0+1
;OFF2: .... Offset register for DIGIT 2
;OFF3: .... Offset register for DIGIT 3

.inclld reg16083.inc
.inclld memap.inc

.extrn TMPND,TBL,BINBCD
.public DISPL,DISPD

.sect drive,rom16

SEGTA:
    ld    OFF1.w,#0           ;clear OFF1 and OFF2
    ld    a,#042             ;point to DIG3 data

$APORT:
    st    a,OFF3             ;put it in OFF3 reg.
    ld    x,#BCDLO          ;point to BCDLO byte
    ld    b,#MWBUFF0        ;point to MWBUFO
    ld    a,[x].b           ;get the bcd lo byte
    and   a,#0f             ;get low nibble
    add   a,OFF1            ;add to the offset reg1
    ld    a,TBL[a].b        ;get the 7-seg code
    st    a,[b].b          ;save the data in MWBUFO
    ld    a,[x+].b         ;x reg points to BCDLO+1
    and   a,#0f0           ;upper nibble of lower
    swap  a                 ;byte of BCDLO
    and   a,#00f           ;clear other bits
    add   a,OFF2            ;add to the OFF2 reg
    ld    a,TBL[a].b        ;
    swap  a                 ;position upper nibble
    and   a,#0f0           ;clear all other bits
    or    MWBUFO,a         ;data (+ dec. point)

$PPORT:
    ld    b,#MWBUFF1        ;point to MWBUFF1
    ld    a,[x].b          ;get BCDLO+1 data
    and   a,#0f            ;get the lower nibble
    add   a,OFF2            ;add the reqd. offset
    ld    a,TBL[a].b        ;get the 7-seg data
    st    a,[b].w          ;
    ifbit 1,[b].b         ;rearrange as PORTP

```

```

sbit 4,MWBUF1 ;data bits are 0,4,7,15
ifbit 2,[b].b ;locations in portP
sbit 0,MWBUF1+1.b ;
ifbit 3,[b].b ;
sbit 4,MWBUF1+1.b ;

$BPORT:
ld b,#MWBUF2 ;point to MWBUF2
ld a,[x].b ;get digit3 data
and a,#0f0 ;get the higher nibble
swap a ;
and a,#0f ;position it right
add a,OFF3 ;add the reqd. offset
ld a,TBL[a].b ;from the table
or a,#0f8 ;sbit 3...7 and save
st a,[b].b ;save it in MWBUF2
ret

SEGTB:
ld OFF1,#016 ;with dec. pt
ld OFF2,#00b ;without dec. pt
ld a,#046 ;
jp $APORT ;

SEGTC:
ld OFF1,#021 ;
ld OFF2,#021 ;
ld a,#04a ;
jp $APORT ;

SEGTD:
ld OFF1,#037 ;
ld OFF2,#02c ;
ld a,#04e ;
jp $APORT ;

DISPL:
jsr BINBCD ;convert bin to BCD
ld COUNT,#05 ;50*20 ms = 1 sec
;10*1 = 10 sec display
ld irpd,#0 ;clear all pending bits
ld tmmode,#04440 ;timer ckt. initialize
ld pwmode,#04444 ;stop all timers
ld tmmode,#0ccc8 ;and acknowledge all
ld pwmode,#0cccc ;interrupts
ld divby,#02222 ;select T2 clock=CKI/16
ld t2reg,#01387 ;LCD refresh rate of
ld r2reg,#01387 ;50 Hz (20 ms) -> 5ms
;per time slot (5000
;counts @ 16.0 Mhz)
rbit 2,tmmodeh ;start timer T2

```

DISP1:

```

        jsr  SEGTA                ;get 7 seg. dat for
        jsr  TMPND                ;refresh time phase Ta
        ;test pending T2

TPO:
        sbit BP1,portbl          ;backplane refresh Ta
        rbit BP2,dirbl          ;make it i/p (Hi-z)
        sbit BP1,dirbl          ;
        rbit BP2,portbl         ;BP1=1, BP2=.5
        jsr  SEGOUT              ;
        jsr  SEGTB               ;time phase Tb
        jsr  TMPND              ;

TP1:
        sbit BP2,portbl          ;BP2 data = 1
        rbit BP1,dirbl          ;make BP1 i/p
        sbit BP2,dirbl          ;send BP2=1
        rbit BP1,portbl         ;Hi-z
        jsr  SEGOUT              ;BP1=.5, BP2=1
        jsr  SEGTC               ;
        jsr  TMPND              ;

TP2:
        rbit BP1,portbl          ;BP1 data=0
        rbit BP2,dirbl          ;BP2 i/p
        sbit BP1,dirbl          ;o/p "0" on BP1
        rbit BP2,portbl         ;BP2 = 0.5
        jsr  SEGOUT              ;
        jsr  SEGTD               ;
        jsr  TMPND              ;

TP3:
        rbit BP2,portbl          ;BP1 data=0
        rbit BP1,dirbl          ;BP2 data=0
        sbit BP2,dirbl          ;make BP1 Hi-z (0.5)
        rbit BP1,portbl         ;BP1=.5, BP2=0
        jsr  SEGOUT              ;
        decsz COUNT              ;do the loop N times
        jp   DISP1               ;
        ld   COUNT,#5            ;
        decsz COUNT2             ;COUNT2 = X*N = set time
        jp   DISP1               ;
        ret                      ;

DISPD:
        ld   portal,#00          ;switch display OFF
        ld   diral,#0ff          ;as o/p
        ld   portbl,#0           ;
        ld   dirbl,#037          ;B0-B2,B5,B4 = outputs
        ld   portp,#0           ;
        ret

SEGOUT:
        ld   porta,MWBUFO        ;portA data (DIG 4+5)

```

```
ld portp,MWBUF1 ;portP data (16-bit reg)
ld b,#MWBUF2 ;
ld x,#portbl ;read portb low byte
ld a,[x].b ;and it with MWBUF2
and a,[b].b ;save original MWBUF2 in
ld k,MWBUF2 ;K register
st a,[b].b ;store MWBUF2&PORTBL in
ld a,k ;MWBUF2
and a,#007 ;get orig. MWBUF2 and
or a,[b].b ;extract B0-B2, OR it
st a,portbl ;with new MWBUF2 and
ret ;send it
```

.endsect

TL/DD/11250-17

```

;Title :   BINBCD.ASM

;Function: This program takes a 16-bit binary number and
;converts into a 20-bit BCD number.
;
;***** RAM MAP *****
;INPUT DATA  -> BINLO+1  BINLO
;
;BCD OUTPUT   -> BCDLO+2  BCDLO+1  BCDLO

.incl memap.inc

BINLO = EVAL

.public BINBCD

.sect code,rom8

BINBCD:
    ld    COUNT,#16           ;Number of left shifts
    ld    bk,#BCDLO,#BCDLO+2 ;
;
$CBCD:
    clr   a                   ;clear BCD ram space
    xs   a,[b+].b             ;
    jp   $CBCD                ;
;
$LSH:
                                ;left shift binary
                                ;routine
    ld   bk,#BINLO,#BINLO+1  ;
    rc   ;reset carry
;
$LSHFT:
    ld   a,[b].b               ;start shifting
    adc  a,[b].b               ;if MSB=1, set C
    xs  a,[b+].b               ;do for all 4 nibbles
    jp  $LSHFT                 ;of the Binary data
;
    ld   bk,#BCDLO,#BCDLO+2  ;
;
$BCDADD:
    ld   a,[b].b               ;get the BCD data
    dadc a,[b].b               ;decimal add with carry
    xs  a,[b+].b               ;put it back
    jp  $BCDADD                ;loop for all 3 bytes
;
    decsz COUNT                ;is shift =16?
;
COUNTER:
    jp  $LSH                    ;no - go back
    ret
;
.endsect

```



```
;Lookup table for customized 2-way MUX LCD I3420
```

```
;
```

```
.includ reg16083.inc
```

```
.public TBL,TMPND,COPY
```

```
.sect table,rom8
```

```
TBL:
```

```
;Timephase Ta ---- 7 segment data
```

```
.byte      08                ;'0' and '.0'  
.byte      0e                ;'1' and '.1'  
.byte      04                ;'2' and '.2'  
.byte      04                ;'3' and '.3'  
.byte      02                ;'4' and '.4'  
.byte      01                ;'5' and '.5'  
.byte      01                ;'6' and '.6'  
.byte      0c                ;'7' and '.7'  
.byte      00                ;'8' and '.8'  
.byte      00                ;'9' and '.9'  
.byte      0f                ;' ' and '.'
```

```
;Timephase Tb ---- 7 segment data
```

```
.byte      04                ;'0'  
.byte      0e                ;'1'  
.byte      05                ;'2'  
.byte      0c                ;'3'  
.byte      0e                ;'4'  
.byte      0c                ;'5'  
.byte      04                ;'6'  
.byte      0e                ;'7'  
.byte      04                ;'8'  
.byte      0c                ;'9'  
.byte      0f                ;' '
```

TL/DD/11250-19

```
.byte 00 ;'.0'  
.byte 0a ;'.1'  
.byte 01 ;'.2'  
.byte 08 ;'.3'  
.byte 0a ;'.4'  
.byte 08 ;'.5'  
.byte 00 ;'.6'  
.byte 0a ;'.7'  
.byte 00 ;'.8'  
.byte 08 ;'.9'  
.byte 0b ;'.'
```

;Timephase Tc ---- 7 segment data

```
.byte 07 ;'0' and '.0'  
.byte 01 ;'1' and '.1'  
.byte 0b ;'2' and '.2'  
.byte 0b ;'3' and '.3'  
.byte 0d ;'4' and '.4'  
.byte 0e ;'5' and '.5'  
.byte 0e ;'6' and '.6'  
.byte 03 ;'7' and '.7'  
.byte 0f ;'8' and '.8'  
.byte 0f ;'9' and '.9'  
.byte 00 ;' ' and '.'
```

TL/DD/11250-20

```
;Timephase Td ---- 7 segment data
```

```
.byte    0b          ;'0'
.byte    01          ;'1'
.byte    0a          ;'2'
.byte    03          ;'3'
.byte    01          ;'4'
.byte    03          ;'5'
.byte    0b          ;'6'
.byte    01          ;'7'
.byte    0b          ;'8'
.byte    03          ;'9'
.byte    00          ;' '

.byte    0f          ;'.0'
.byte    05          ;'.1'
.byte    0e          ;'.2'
.byte    07          ;'.3'
.byte    05          ;'.4'
.byte    07          ;'.5'
.byte    0f          ;'.6'
.byte    05          ;'.7'
.byte    0f          ;'.8'
.byte    07          ;'.9'
.byte    04          ;'. '
```

```
;
```

```
;Digit '3' codes
```

```
;Time phase Ta
```

```
.byte    07          ;''
.byte    06          ;'1'
.byte    04          ;'2'
.byte    04          ;'3'
```

```
;Timephase Tb
```

```
.byte    07          ;''
.byte    06          ;'1'
.byte    05          ;'2'
.byte    06          ;'3'
```

```
;Timephase Tc
```

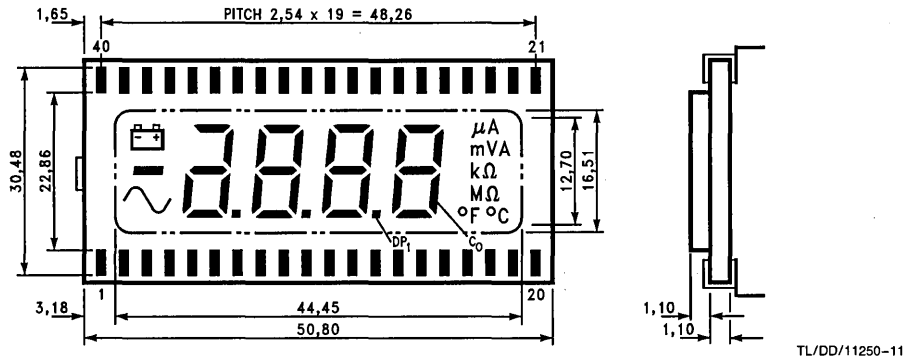
```
.byte    00          ;''
.byte    01          ;'1'
.byte    03          ;'2'
.byte    03          ;'3'
```

```
;Timephase Td
```

```
.byte    00          ;''
.byte    01          ;'1'
.byte    02          ;'2'
.byte    01          ;'3'

TMPND:
        ld      b,#tmmoden
$LOOP:
        ifbit   1,[b].b
        jp      $SEND
        jp      $LOOP
$SEND:
        sbit   3,[b].b
        ret
COPY:
        ld      a,[b].w
        x      a,[x].w
        ret
.endsect
```

TL/DD/11250-22



Segment	/	Backplane	Pin Assignment (Proposal)
BP1		BP2	23
Minus		~	1
K		m	22
Ω		A	23
nΩ		μA	25
±		V	40/24
B ₀		C ₀	26
A ₀		D ₀	27/17
G ₀		E ₀	29
F ₀		DP ₁	28
B ₁		C ₁	30
A ₁		D ₁	31/16
G ₁		E ₁	33
F ₁		DP ₂	32
B ₂		C ₂	34
A ₂		D ₂	35/4
G ₂		E ₂	37
F ₂		DP ₃	36
B ₃		C ₃	38
ADG ₃		E ₃	39
		°C	19
		°F	18

The pairing and the annunciators may be slightly rearranged by the LCD manufacturer.

ADG₃ = A₃, D₃, G₃
(One segment)

FIGURE 7

Improved UART Cloning Techniques on New Generation HPCs

National Semiconductor
Application Note 798
Ravi Kumar



The new generation HPCs have on-chip UARTs with much better baud rate generation techniques and better status reporting capabilities. This article explains in detail, accurate baud rate generation on HPC46400E and HPC+ UARTs with appropriate examples.

UART implemented on the HPC46400E and HPC+ is an upward compatible enhancement of the UART present on the HPC46083. Unlike the UART on HPC46083, the operating mode may be selected as either Asynchronous or Synchronous. Here we can also select the baud rate through software in conjunction with both prescaler and baud select registers.

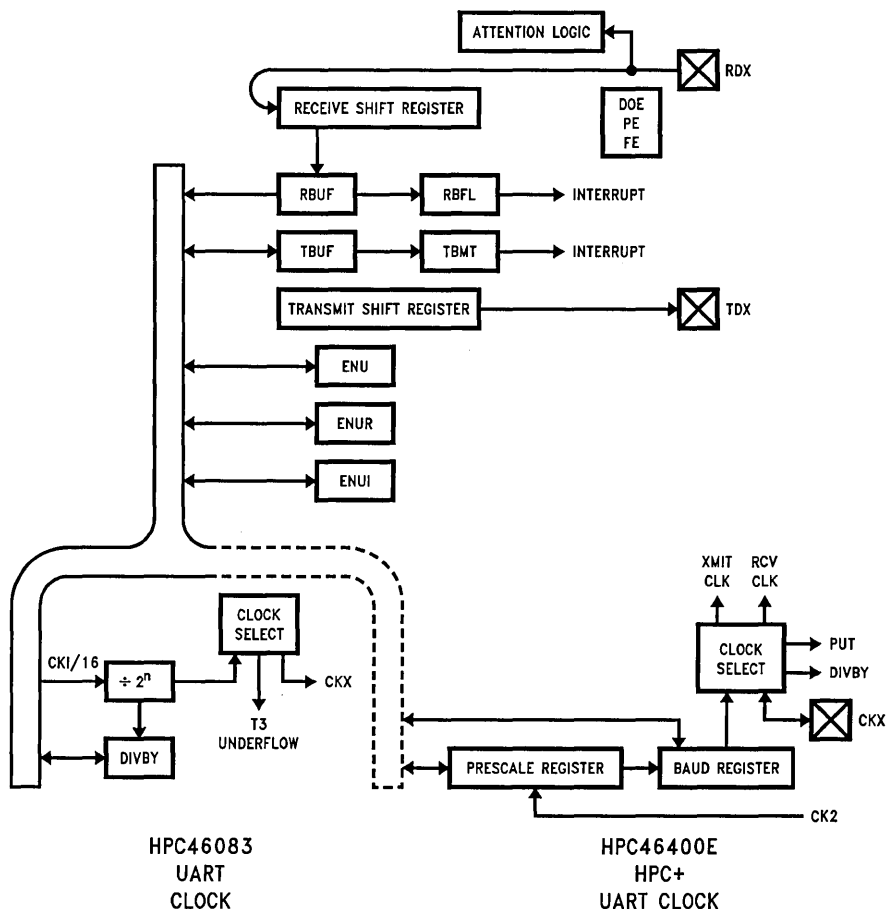


FIGURE 1

TL/DD/11292-1

COMMON FEATURES SUPPORTED BY HPC46083 UART AND THE NEWER VERSION OF UART ON HPC46400E AND HPC+

- Fully programmable serial interface characteristics, including:
 - 8- or 9-bit characters
 - 1 or 2 stop bits
- Two interrupt sources (Receiver buffer full, Transmit buffer empty)
- Independent clock inputs (either on-chip or off-chip) for the transmitter and receiver
- Error reporting capabilities (Data overrun error, framing error)
- Attention or wake up mode for receiver to enhance networking capability

ADDITIONAL UART FEATURES AVAILABLE ON HPC46400E AND HPC+

- Upwardly compatible from earlier HPC UARTs such as HPC16083

- Fully programmable serial interface characteristics, including:
 - Accurate baud rate generation without the penalty of using an expensive crystal up to 625k baud
 - 7-bit characters possible
 - $\frac{7}{8}$, $1\frac{1}{8}$ stop bit lengths
 - Odd, Even, Mark, Space or no parity bit generation and detection
- Selectable Asynchronous or Synchronous mode of operation
- Loopback Diagnostic test capability

Now lets see various methods of BAUD Rate generation. First we shall discuss how DIVBY can be used to generate required baud-rate.

1.0 UART CLOCK SOURCE FROM DIVBY REGISTER

Clock for DIVBY register can be generated using precise value crystals or T3 underflow. Referring to *Figure 2*, we see that baud rate is from internal source for DIVBY register.

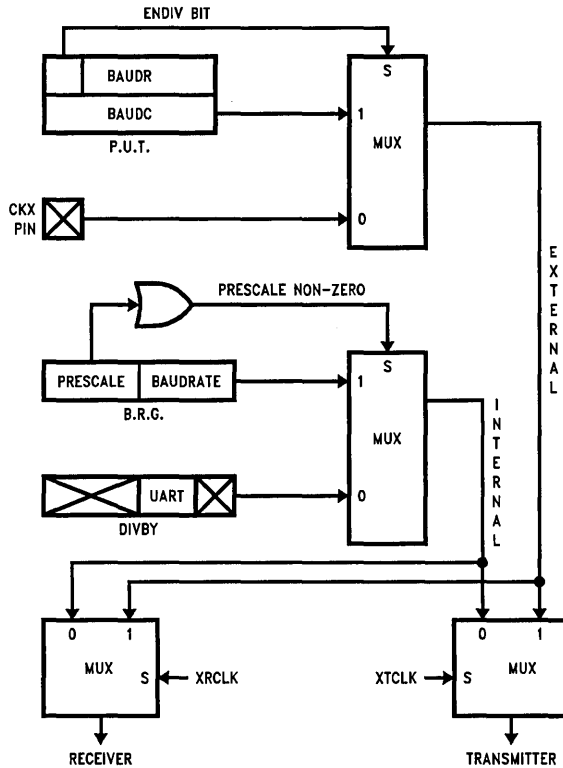


FIGURE 2. Simplified UART Clock Routing

TL/DD/11292-2

The following is a sample assembly language routine illustrating BAUD Rate generation using DIVBY register through precise value crystal.

```

;This program will test the HFCL6400E UART for 9600 baud.
;using 10.0 MHz crystal and DIVBY (baud clock from internal source).
;*****
;The power-up default setup is:
;a) Baud clock from internal source DIVBY
;b) Frame format is 1 start, 8 data, and 1 stop bit.
;The clock should be a 10.0 MHz crystal

.sect uart, rom16

begin:
    sbit 0,Of2.b           ;DIRB reg pin 1 outward direction
    sibit 0,Of4.b         ;BFUNL reg, turns on TDX bit
    rbit 2,0x122.b        ;xtclk
    rbit 3,0x122.b        ;xrclk
    ld 018e.b,#040        ;Load DIVBY from table to generate
                        ;9600 baud (CKI/64)

;The baud clock = baud rate * 16
;So, for 9600 baud, belk = 9600 * 16 = 153600 Hz
;With 10.0 MHz clock → 10.0 MHz/64 = 156250 Hz (within 5%)

xmit:    ld a,#041        ;load char "A"
        st a,0126.b      ;Load TBUF reg to transmit
chk:     ifbit 0,0120.b
        jp xmit          ;continue to xmit
        jp chk

.endsect
.end begin

```

Hence we see the percentage error of Baud Rate produced is:

$$\begin{aligned} \% \text{ error} &= (156250 - 153600)/153600 \\ &= 1.72 \end{aligned}$$

which is within the error limits.

A) Baud Rate Calculation Using DIVBY Register through Precise Value Crystal

Table I gives the bit values to be loaded into UART section of the DIVBY register. This table defines the baud rates for two different crystals at 9.304 MHz and 19.6608 MHz.

We see that more care in selecting the crystal frequency is necessary to generate exact baud rates. Obviously the baud rate generation is restricted by the crystal frequency.

TABLE I

Bit 7	Bit 6	Bit 5	Bit 4	Baud Clock (x16 Clock)	Baud Rate 9.8304 MHz Crystal	Baud Rate 17.6603 MHz Crystal
0	0	0	0	← Not Allowed →		
0	0	0	0	← Defined by Timer T3 Underflow →		
0	0	1	0	CKI/16	38400	65536
0	0	1	1	CKI/32	19200	32768
0	1	0	0	CKI/64	9600	16384
0	1	0	1	CKI/128	4800	8192
0	1	1	0	CKI/256	2400	4096
0	1	1	1	CKI/512	1200	2048
1	0	0	0	CKI/1024	600	1024
1	0	0	1	CKI/2048	300	512
1	0	1	0	CKI/4096	150	256
1	0	1	1	CKI/8192	75	128
1	1	0	0	CKI/16384	38	64
1	1	0	1	CKI/32768	19	32
1	1	1	0	CKI/65536	9.4	16
1	1	1	1	CKI/131072	4.7	8

B) Baud Rate Calculation Using DIVBY Register and Timer T3 Underflow

Suppose we want to generate 9600 baud. In the DIVBY register, load the UART bits with value 0001, which means Baud Clock is defined by T3 underflow (refer to Table I). Once again referring to *Figure 2*, we see BAUD clock is from internal source.

Let's calculate the Pre-Scale value to be loaded into T3 register (0X018C) and R3 register (0X018A)

Baud Clock = Required baud rate \times 16

$$\text{Clock Input} = \frac{\text{Crystal Freq}}{16}$$

$$\text{Pre-Scale Value} = \frac{\text{Clock Input}}{\text{Baud Clock}}$$

In our specific case

required BAUD Rate = 9600

crystal freq = 20 MHz

$$\rightarrow \text{Baud Clock} = 9600 \times 16 = 15360$$

$$\text{Clock Input} = 20/16 = 1.25 \text{ MHz} = 1.25 \times 10^6 \text{ Hz}$$

$$\text{Pre-Scale Value} = \frac{1.25 \times 10^6}{153600} \approx 8$$

Pre-Scale Value = 8

Actual value to be loaded into T3 and R3 register is (Pre-scale value - 1) i.e., 7 in this case.

Percentage error of Baud Rate produced is:

Pre-Scale Value = 8

Baud Rate = Baud Clock/16

Baud Clock = Clock Input/Pre-Scale Value

Clock Input = CKI/16 = 20 MHz/16

= 1.25 MHz

Baud Clock = 1.25 MHz/8 = 156250

Therefore Baud Rate = 156250/16 = 9765.62

Hence % error = (9765.62 - 9600)/9600
= 1.72

Which is within the error limits.

The following is a sample assembly language routine illustrating BAUD Rate generation through DIVBY and T3 underflow.

```

;***** Generation of BAUD clock through timer3 without triggering timer intrpt *****

.sect uart, rom16

tmr:          ld_tmmode.w,#0xcccc          ;stop timers t3, t2, t1
              ld_divby.w,#0x2010         ;Clk to T3 thru DIVBY as CKI/16
                                            ;with reload val Ton & Toff as 7
              ld_t3reg.w,#0x7           ;Reload Ton as 7
              ld_r3reg.w,#0x7           ;Reload Toff as 7
                                            ;and BAUD rate = 9600
              rbit 2,0x122.b            ;uart internal xmit clk
              rbit 3,0x122.b            ;uart internal rcv clk
              sbit 0,0x0f2.b           ;config BFUN pin as TDX
              sbit 0,0x0f4.b           ;config DIRB pin 1 outward
              ld_tmmode.w,#0x8ccc       ;Start timer T3 & stop T1, T2 & Ack'em.

;Loop to continuously xmit char "A" at specified baud rate

xmit:         ld a,#041                  ;load char "A"
              st a,0126.b               ;load TBUF reg to transmit
chk:          ifbit 0,_enu.b
              jp xmit
              jp chk

.endsect
.end tmr

```

2.0 BAUD RATE CALCULATION USING PUT (PRECISION UART TIMER)

The Precision UART Timer (PUT) is now obsolete and kept only for compatibility with software developed for those earlier components. PUT has two registers i.e., BAUDR with 15-bit divisor field and BAUDC, a 15-bit free-running down counter. These can be programmed to divide the CK2 (CKI/2) clock by a factor of from 3 to 32767, in units of CK2, thus yielding a time base to the UART of higher resolution than that available through the DIVBY register.

Referring to *Figure 2* we see that BAUD clock source for PUT is external.

Suppose the Clock input is 16 MHz and the required baud rate is 9600, then the value to be loaded into BAUDR register will be

$$\text{Required Baud Rate} = \frac{(\text{CK2}/16)}{(\text{BAUDR} + 1)}$$

Where CK2 = CKI/2
Given CKI = 16 MHz
Hence CK2 = 8 MHz

$$(\text{BAUDR} + 1) = \frac{\text{CK2}/16}{\text{Required Baud Rate}}$$

$$(\text{BAUDR} + 1) = \frac{8 \text{ MHz}/16}{(9600)}$$

The following is a sample assembly language routine illustrating BAUD Rate generation through PUT.

```

;This program will test the HPC16400E UART for 9600 baud.
;Using PUT for generating 9600 baud at 20 MHz
.sect code, rom 16
This is for 20 MHz CKI
;
;Using PUT for generating 9600 baud at 20 MHz
.sect code, rom 16
main:

        ld 0x017e.w,#0x0000      ;for 9600 baud @ 20 MHz
        ld 0x017c.w,#0x8033      ;UDIV w/xtclk or xrelk (baud count)
        sbit 2, 0x122.b          ;baud div value to generate 9600 baud
        sbit 3, 0x122.b          ;UDIVR (baud div) register
                                   ;xtclk
                                   ;xrelk

        ld 0f2.b,#0x05           ;DIRB reg pin 1 outward direction.
        ld 0f4.b,#0x05           ;BFUNL reg, turns on TDX bit

;char xmission
ld a,#041                        ;Load char "A"
xmit:
st a,0126.b                      ;Load TBUF reg to transmit

jp xmit                          ;Continue to xmit

.endsect
.end main

```

$$\therefore \text{BAUDR} \approx 52 - 1 \quad 51 \text{ in decimal}$$

and here value to be loaded into BAUDR register will be 33 hex.

Now to select PUT timer as external clock source MSB of BAUDR register must be 1.

$$\begin{array}{r} 1000 \ 0000 \ 0011 \ 0011 \text{ — Binary} \\ 8 \quad 0 \quad 3 \quad 3 \quad \text{— Hex} \end{array}$$

Note: BAUDC must also be loaded with same value (Reload Value).

Percentage error of Baud Rate produced is:

$$\text{BAUDR} = 51$$

$$\text{Therefore Baud Rate} = \frac{8 \text{ MHz}/16}{(51 + 1)}$$

$$= 9615.38$$

$$\text{Required Baud Rate} = 9600$$

$$\text{Hence \% Error} = (9615.38 - 9600)/9600$$

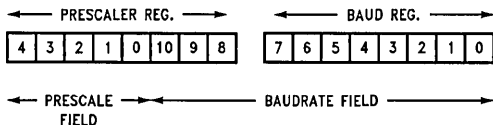
$$= 0.16$$

Which is well within the error limits.

3.0 BAUD RATE CALCULATIONS USING BRG (BAUD RATE GENERATOR).

The most flexible and accurate on-chip clocking is provided by the BAUD Rate generator and (BRG). The BAUD Rate generator is controlled by the register pair PSR and BAUD, shown below. The Prescale factor is selected by the upper 5 bits of the PSR register (the PRESCALE field), in units of the CK2 clock from 1 to 16 in 1/2 step increments. The lower 3

bits of the PSR register, in conjunction with the 8 bits of the baud register, form the 11-bit BAUDRATE field, which defines a baud rate divisor ranging from 1 to 2048, in units of the prescaled clock selected by the PRESCALE field. In Asynchronous Mode, the resulting baud rate is 1/16 of the clocking rate selected through the BRG circuit. The maximum baud rate generated using BRG is 625 kbaud.



TL/DD/11292-3

Formula:

$$\text{Required Baud Rate} = \frac{\text{CKI}}{32 * N * P}$$

where CK = Input Clock

N = Baud Rate Divisor

P = Prescaler Division Factor

Note: This calculation is for Asynchronous mode of UART operation.

Suppose we need 9600 Baud with given Clock i.e., CKI = 20 MHz

then

$$\text{Required Baud Rate} = 9600$$

$$\text{CKI} = 20 \text{ MHz}$$

From formula stated earlier for required baud rate, we have

$$9600 = \frac{20 \text{ MHz}}{32 * N * P}$$

$$\rightarrow N * P = \frac{20 * 10^6}{32 * 9600}$$

$$N * P = 65.1$$

$$\text{or } N = 65.1/P$$

P, which is a prescaler factor, should be selected from Table II in such a way that "N" should be close to an integer. Therefore substituting values of P in the table and calculating N we have the following table.

TABLE II

P Prescaler	N N = (65.104/P)
1	65.104
1.5	43.402
2	32.552
2.5	26.041
3	21.701
3.5	18.601
4	16.276
4.5	14.467
5	13.020
5.5	11.837
6	10.850
6.5	10.016
7	9.300
7.5	8.680
8	8.138
8.5	7.659
9	7.233
9.5	6.853
10	6.510
10.5	6.200
11	5.918
11.5	5.661
12	5.425
12.5	5.203
13	5.008
13.5	4.822
14	4.650
14.5	4.489
15	4.340
15.5	4.200
16	4.069

← Value Closest
to an Integer

Now choose N in such a way that it's closest to an integer. Obviously N = 5.008 is the closest to being an integer therefore, the value of P when N = 5.008 is 13

$$\rightarrow P = 13 \text{ and } N = 5$$

Now from the table "UART Prescaler Factors" select the binary "Prescale field" using the value of N derived above.

Percentage error of the Baud Rate produced is:

from the above table P = 13 and N = 5.008

$$\therefore \text{Baud Rate} = \frac{20 \text{ MHz}}{32 \times N \times P}$$

$$\frac{20 \times 10^6}{32 \times 5.008 \times 12} = 9600.02$$

$$\% \text{ error} = (9600.02 - 9600)/9600$$

$$= 0.0002\%$$

Which is obviously negligible.

UART Prescaler Factors

Prescale Field (Binary)	Prescaler Factor
00000	(Compatibility Mode)
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11010	13.5
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

in Binary format

$$P = 11001 \quad (N - 1) = 100$$

Therefore Prescaler field is $P = 11001$ and baud rate divisor or baud rate field $N = 100$

Referring to BRG register format in page 7 we can combine 5 bits of P and 11 bits of baud rate field to load Prescaler bits (PSR) and Baud Rate generate bits (BRG) respectively.

$$PSR = 11001$$

$$\text{Baud Rate field } (N - 1) = 00000000100$$

Combined value in binary format is

$$1100 \ 1000 \ 0000 \ 0100$$

which in hex is

$$\text{C} \quad \text{8} \quad \text{0} \quad \text{4}$$

therefore load BRG register with C804.

The following is a sample assemble language routine illustrating BAUD Rate generation through BRG.

```

;Baud rate generation using BRG register
;BAUD RATE = CKI/(32 * N * P) where P = 5 bit prescalar value and N = 11 bit
;baud rate filed. For 9600 baud at 20 MHz → NP = 52.083 and so P = 13 and N = 4
;
;At 16 MHz crystal (CKI) for PSR use #0c8 and for BAUD use #07
;At 20 MHz crystal (CKI) for PSR use #0c8 and for BAUD use #04
;*****
.sect code, roml6
main:
                                ;First exit compatibility mode
                                ;by writing to PSR register
                                ;load prescalar i.e., PSR reg
                                ;load baudrate field i.e., BAUD at 20 MHz
                                ;8 bit data, space (0) parity in ENU register.
                                ;ENUI register, 2 stop bits
                                ;DIRB register pin 1 outward direction
                                ;BFUNL register, turns on TDX bit
                                ;Loop to continuously xmit chars at specified baud rate.
xmit:   ld a,#041                ;load char "A"
                                ;Load TBUF reg to transmit
                                ;Continue to xmit.
                                st a,0126.b
                                jp xmit
.endsect
.end main

```

Performance Comparison of PUT and BRG Regarding Higher Baud Rate Generation.

Let's take a case where the required Baud rate is 625k baud at 20 MHz.

PUT:

$$\text{BAUDR} + 1 = \frac{\text{CK2/16}}{\text{Required Baud Rate}}$$

$$\text{Therefore BAUR} + = \frac{10 \times 10^6 / 16}{625 \times 10^3}$$

$$\text{BAUDR} + 1 = 0.1$$

$$\text{BAUDR} = -(0.9)$$

Therefore we see that, PUT can not be used to generate 6.25k baud limit on PUT is 208.3 baud.

BRG:

$$\text{Baud Rate Required} = \frac{\text{CKI}}{32 * N * P}$$

$$625k = \frac{20 \times 10^6}{32 * N * P}$$

$$N \times P = 1$$

$$N = 1 \quad P = 1$$

i.e. Prescale field = 00001 N - 1 = 000

i.e., 000 1000 0000 0000 = 0 × 0800

Therefore load BRG register with 0x 0800 to generate 625k baud @ 20 MHz

Conclusion:

Thus we see that the clocking techniques on new generation HPCs are more accurate and very flexible. Generation of higher rates can be done with relative ease. We can also observe that, using newer UART clocking techniques the percentage error i.e., difference between the required baud rate and the actual baud rate produced goes down significantly.



Section 6
**MICROWIRE and
MICROWIRE/PLUS
Peripherals**



Section 6 Contents

MICROWIRE and MICROWIRE/PLUS: 3-Wire Serial Interface	6-3
COP472-3 Liquid Crystal Display Controller	6-7

MICROWIRE™ and MICROWIRE/PLUS™: 3-Wire Serial Interface

National's MICROWIRE and MICROWIRE/PLUS provide for high-speed, serial communications in a simple 3-wire implementation.

Originally designed to interface COP400 microcontrollers to peripheral devices, the MICROWIRE protocol has been extended to both the COP800 and HPCTM families with the enhanced version, MICROWIRE/PLUS.

Because the shift clock in MICROWIRE/PLUS can be internal or external, the interface can be designated as either bus master or slave, giving it the flexibility necessary for distributed and multiprocessing applications.

With its simple 3-wire interface, MICROWIRE/PLUS can connect a variety of nodes in a serial-communication network.

This simple 3-wire design also helps increase system reliability while reducing system size and development time.

MICROWIRE/PLUS consists of an 8-bit serial shift register (SIO), serial data input (SI), serial data output (SO), and a serial shift clock (SK).

Because the COP800 and HPC families have memory-mapped architectures, the contents of the SIO register can be accessed through standard memory-addressing instructions.

The control register (CNTRL) is used to configure and control the mode and operation of the interface through user-selectable bits that program the internal shift rate. This greatly increases the flexibility of the interface.

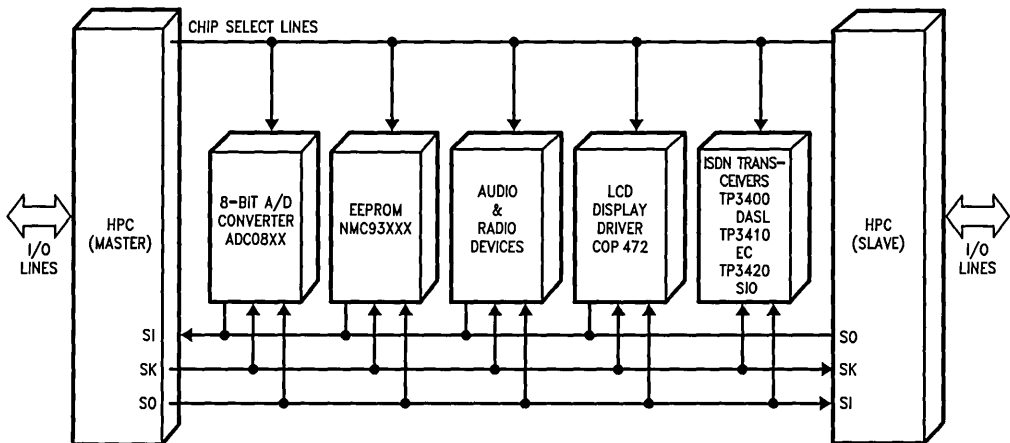
MICROWIRE/PLUS can also provide additional I/O capability for COP800 and HPC microcontrollers by connecting, for example, external 8-bit parallel-to-serial shift registers to 8-bit serial-to-parallel shift registers.

And it can interface a wide variety of peripherals:

- Memory (CMOS RAM and EEPROM)
- A/D converters
- Timers/counters
- Digital phase locked-loops
- Telecom peripherals
- Vacuum fluorescent display drivers
- LED display drivers
- LCD display drivers

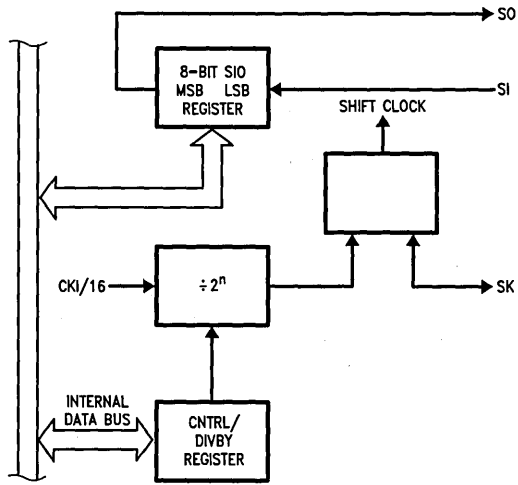
Both MICROWIRE and MICROWIRE/PLUS give all the members of National's microcontroller families the flexibility and design-ease to implement a solution quickly, simply, and cost-effectively.

MICROWIRE/PLUS System Block



TL/XX/0074-1

MICROWIRE/PLUS Block Diagram



TL/XX/0074-2

MICROWIRE and MICROWIRE/PLUS Peripherals

Part Number	Description	Databook
A/D CONVERTERS AND COMPARATORS		
ADC0811	11 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0819	19 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0831	1 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0838	8 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0832	2 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0833	4 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0834	4 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0852	Multiplexed Comparator with 8-Bit Reference Divider	Linear
ADC0854	Multiplexed Comparator with 8-Bit Reference Divider	Linear
DISPLAY DRIVERS		
COP472-3	3 x 12 Multiplexed Expandable LCD Display Driver	Microcontroller
MM5450	35 Output LED Display Driver	Interface
MM5451	34 Output LED Display Driver	Interface
MM5483	31 Segment LCD Display Driver	Interface
MM5484	16 Segment LED Display Driver	Interface
MM5486	33 Output LED Display Driver	Interface
MM58201	8 Backplane and 24 Segment Multiplexed LCD Driver	Interface
MM58241	32 Output High Voltage Display Driver	Interface
MM58242	20 Output High Voltage Display Driver	Interface
MM58248	35 Output High Voltage Display Driver	Interface
MM58341	32 Output High Voltage Display Driver	Interface
MM58342	20 Output High Voltage Display Driver	Interface
MM58348	35 Output High Voltage Display Driver	Interface
MEMORY DEVICES		
NMC9306	16 x 16 NMOS EEPROM	Memory
NMC9313B	16 x 16 NMOS EEPROM	Memory
NMC9314B	64 x 16 NMOS EEPROM	Memory
NMC9346	64 x 16 NMOS EEPROM	Memory
NMC93C06	16 x 16 CMOS EEPROM	Memory
NMC93C46	64 x 16 CMOS EEPROM	Memory
NMC93CS06	16 x 16 CMOS EEPROM with Write Protect	Memory
NMC93CS46	64 x 16 CMOS EEPROM with Write Protect	Memory
NMC93CS56	128 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C56	128 x 16 CMOS EEPROM	Memory
NMC93CS66	256 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C66	256 x 16 CMOS EEPROM	Memory

MICROWIRE and MICROWIRE/PLUS Peripherals (Continued)

Part Number	Description	Databook
TELECOM DEVICES		
TP3420	S Interface Device (SID)	Telecom
AUDIO AND RADIO DEVICES		
DS8906	AM/FM Digital PLL Synthesizer	Interface
DS8907	AM/FM Digital PLL Frequency Synthesizer	Interface
DS8908	AM/FM Digital PLL Frequency Synthesizer	Interface
DS8911	AM/FM/TV Sound Up-Conversion Frequency Synthesizer	Interface
LMC1992	Stereo Volume/Tone/Fade with Source Select	Linear
LMC1993	Stereo Volume/Tone/Fade/Loudness with Source Select	Linear
LMC835	7 Band Graphic Equalizer	Linear

COP472-3 Liquid Crystal Display Controller

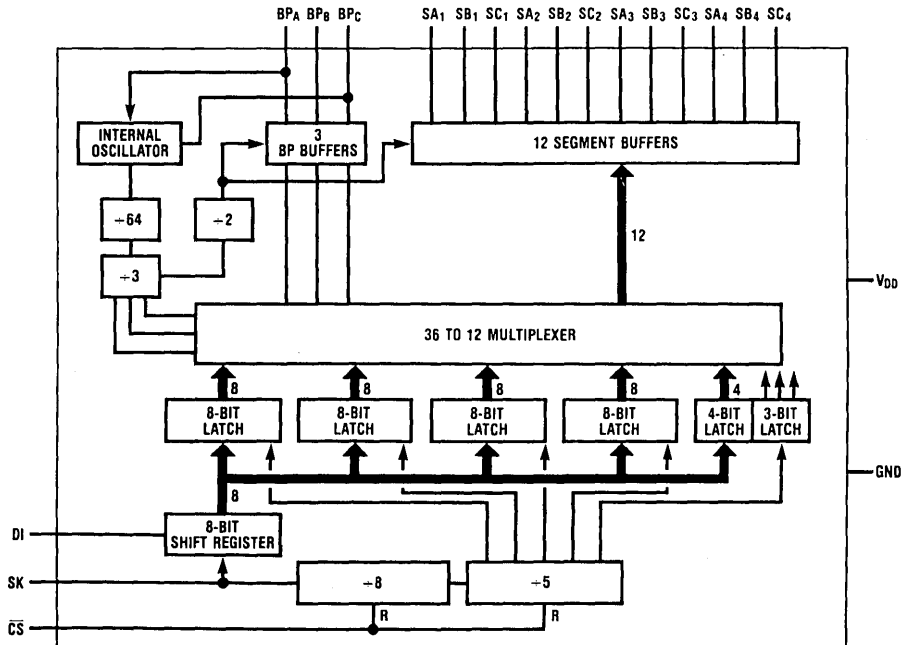
General Description

The COP472-3 Liquid Crystal Display (LCD) Controller is a peripheral member of the COPSTM family, fabricated using CMOS technology. The COP472-3 drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472-3 contains an on-chip oscillator and generates all the multi-level waveforms for backplanes and segment outputs on a triplex display. One COP472-3 can drive 36 segments multiplexed as 3 x 12 (4½ digit display). Two COP472-3 devices can be used together to drive 72 segments (3 x 24) which could be an 8½ digit display.

Features

- Direct interface to TRIPLEX LCD
- Low power dissipation (100 μ W typ.)
- Low cost
- Compatible with all COP400 processors
- Needs no refresh from processor
- On-chip oscillator and latches
- Expandable to longer displays
- Software compatible with COP470 V.F. Display Driver chip
- Operates from display voltage
- MICROWIRE™ compatible serial I/O
- 20-pin Dual-In-Line package

Block Diagram



TL/DD/6932-1

Absolute Maximum Ratings

Voltage at CS, DI, SK pins	-0.3V to +9.5V	Storage Temperature	-65°C to +150°C
Voltage at all other Pins	-0.3V to $V_{DD} + 0.3V$	Lead Temp. (Soldering, 10 Seconds)	300°C
Operating Temperature Range	0°C to 70°C		

DC Electrical Characteristics

GND = 0V, V_{DD} = 3.0V to 5.5V, T_A = 0°C to 70°C (depends on display characteristics)

Parameter	Conditions	Min	Max	Units
Power Supply Voltage, V_{DD}		3.0	5.5	Volts
Power Supply Current, I_{DD} (Note 1)	$V_{DD} = 5.5V$		250	μA
	$V_{DD} = 3V$		100	μA
Input Levels DI, SK, CS V_{IL} V_{IH}		0.7 V_{DD}	0.8 9.5	Volts Volts
BPA (as Osc. in) V_{IL} V_{IH}		$V_{DD} - 0.6$	0.6 V_{DD}	Volts Volts
Output Levels, BPC (as Osc. Out) V_{OL} V_{OH}		$V_{DD} - 0.4$	0.4 V_{DD}	Volts Volts
Backplane Outputs (BPA, BPB, BPC) $V_{BPA, BPB, BPC}$ ON $V_{BPA, BPB, BPC}$ OFF	During BP+ Time	$V_{DD} - \Delta V$ $\frac{1}{3} V_{DD} - \Delta V$	V_{DD} $\frac{1}{3} V_{DD} + \Delta V$	Volts Volts
	During BP- Time	0 $\frac{2}{3} V_{DD} - \Delta V$	ΔV $\frac{2}{3} V_{DD} + \Delta V$	Volts Volts
Segment Outputs ($SA_1 \sim SA_4$) V_{SEG} ON V_{SEG} OFF	During BP+ Time	0 $\frac{2}{3} V_{DD} - \Delta V$	ΔV $\frac{2}{3} V_{DD} + \Delta V$	Volts Volts
	During BP- Time	$V_{DD} - \Delta V$ $\frac{1}{3} V_{DD} - \Delta V$	V_{DD} $\frac{1}{3} V_{DD} + \Delta V$	Volts Volts
Internal Oscillator Frequency		15	80	kHz
Frame Time (Int. Osc. \div 192)		2.4	12.8	ms
Scan Frequency ($1/T_{SCAN}$)		39	208	Hz
SK Clock Frequency		4	250	kHz
SK Width		1.7		μs
DI Data Setup, t_{SETUP} Data Hold, t_{HOLD}		1.0 100		μs ns
\overline{CS} t_{SETUP} t_{HOLD}		1.0 1.0		μs μs
Output Loading Capacitance			100	pF

Note 1: Power supply current is measured in stand-alone mode with all outputs open and all inputs at V_{DD} .

Note 2: $\Delta V = 0.05V_{DD}$.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at CS, DI, SK Pins	-0.3V to +9.5V
Voltage at All Other Pins	-0.3V to $V_{DD} + 0.3V$
Operating Temperature Range	-40°C to +85°C

Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

DC Electrical Characteristics

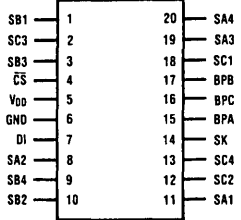
$GND = 0V$, $V_{DD} = 3.0V$ to $5.5V$, $T_A = -40^\circ C$ to $+85^\circ C$ (depends on display characteristics)

Parameter	Conditions	Min	Max	Units
Power Supply Voltage, V_{DD}		3.0	5.5	Volts
Power Supply Current, I_{DD} (Note 1)	$V_{DD} = 5.5V$		300	μA
	$V_{DD} = 3V$		120	μA
Input Levels DI, SK, CS V_{IL} V_{IH}		0.7 V_{DD}	0.8 9.5	Volts Volts
BPA (as Osc. In) V_{IL} V_{IH}		$V_{DD} - 0.6$	0.6 V_{DD}	Volts Volts
Output Levels, BPC (as Osc. Out) V_{OL} V_{OH}		$V_{DD} - 0.4$	0.4 V_{DD}	Volts Volts
Backplane Outputs (BPA, BPB, BPC) $V_{BPA, BPB, BPC}$ ON $V_{BPA, BPB, BPC}$ OFF	During BP+ Time	$V_{DD} - \Delta V$ $\frac{1}{3} V_{DD} - \Delta V$	V_{DD} $\frac{1}{3} V_{DD} + \Delta V$	Volts Volts
	During BP- Time	0 $\frac{2}{3} V_{DD} - \Delta V$	ΔV $\frac{2}{3} V_{DD} + \Delta V$	Volts Volts
Segment Outputs ($SA_1 \sim SA_4$) V_{SEG} ON V_{SEG} OFF	During BP+ Time	0 $\frac{2}{3} V_{DD} - \Delta V$	ΔV $\frac{2}{3} V_{DD} + \Delta V$	Volts Volts
	During BP- Time	$V_{DD} - \Delta V$ $\frac{1}{3} V_{DD} - \Delta V$	V_{DD} $\frac{1}{3} V_{DD} + \Delta V$	Volts Volts
Internal Oscillator Frequency		15	80	kHz
Frame Time (Int. Osc. \div 192)		2.4	12.8	ms
Scan Frequency ($1/T_{SCAN}$)		39	208	Hz
SK Clock Frequency		4	250	kHz
SK Width		1.7		μs
DI Data Setup, t_{SETUP} Data Hold, t_{HOLD}		1.0 100		μs ns
\overline{CS} t_{SETUP} t_{HOLD}		1.0 1.0		μs μs
			100	pF

Note 1: Power supply current is measured in stand-alone mode with all outputs open and all inputs at V_{DD} .

Note 2: $\Delta V = 0.05 V_{DD}$.

Dual-In-Line Package

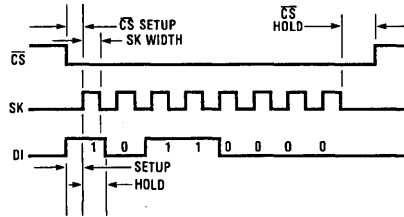


TL/DD/6932-2

Pin	Description
\overline{CS}	Chip select
V _{DD}	Power supply (display voltage)
GND	Ground
DI	Serial data input
SK	Serial clock input
BPA	Display backplane A (or oscillator in)
BPB	Display backplane B
BPC	Display backplane C (or oscillator out)
SA1 ~ SC4	12 multiplexed outputs

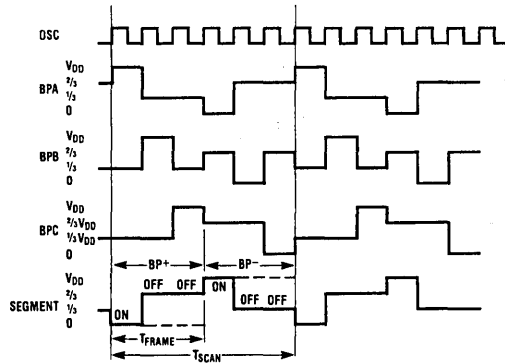
Order Number COP472MW-3 or COP472N-3
See NS Package Number M20A or N20A

FIGURE 2. Connection Diagram



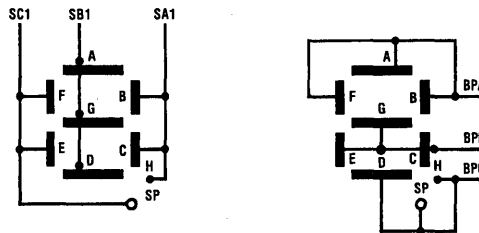
TL/DD/6932-3

FIGURE 3. Serial Load Timing Diagram



TL/DD/6932-4

FIGURE 4. Backplane and Segment Waveforms



TL/DD/6932-5

FIGURE 5. Typical Display Internal Connections
Epson LD-370

Functional Description

The COP472-3 drives 36 bits of display information organized as twelve segments and three backplanes. The COP472-3 requires 40 information bits: 36 data and 4 control. The function of each control bit is described below. Display information format is a function of the LCD interconnections. A typical segment/backplane configuration is illustrated in *Figure 5*, with this configuration the COP472-3 will drive 4 digits of 9 segments.

To adapt the COP472-3 to any LCD display configuration, the segment/backplane multiplex scheme is illustrated in Table I.

Two or more COP472-3 chips can be cascaded to drive additional segments. There is no limit to the number of COP472-3's that can be used as long as the output loading capacitance does not exceed specification.

TABLE I. COP472-3 Segment/Backplane Multiplex Scheme

Bit Number	Segment, Backplane	Data to Numeric Display	
1	SA1, BPC	SH	
2	SB1, BPB	SG	
3	SC1, BPA	SF	
4	SC1, BPB	SE	Digit 1
5	SB1, BPC	SD	
6	SA1, BPB	SC	
7	SA1, BPA	SB	
8	SB1, BPA	SA	
9	SA2, BPC	SH	
10	SB2, BPB	SG	
11	SC2, BPA	SF	
12	SC2, BPB	SE	Digit 2
13	SB2, BPC	SD	
14	SA2, BPB	SC	
15	SA2, BPA	SB	
16	SB2, BPA	SA	
17	SA3, BPC	SH	
18	SB3, BPB	SG	
19	SC3, BPA	SF	
20	SC3, BPB	SE	Digit 3
21	SB3, BPC	SD	
22	SA3, BPB	SC	
23	SA3, BPA	SB	
24	SB3, BPA	SA	
25	SA4, BPC	SH	
26	SB4, BPB	SG	
27	SC4, BPA	SF	
28	SC4, BPB	SE	Digit 4
29	SB4, BPC	SD	
30	SA4, BPB	SC	
31	SA4, BPA	SB	
32	SB4, BPA	SA	
33	SC1, BPC	SPA	Digit 1
34	SC2, BPC	SP2	Digit 2
35	SC3, BPC	SP3	Digit 3
36	SC4, BPC	SP4	Digit 4
37	not used		
38	Q6		
39	Q7		
40	SYNC		

SEGMENT DATA BITS

Data is loaded in serially, in sets of eight bits. Each set of segment data is in the following format:

SA	SB	SC	SD	SE	SF	SG	SH
----	----	----	----	----	----	----	----

Data is shifted into an eight bit shift register. The first bit of the data is for segment H, digit 1. The eighth bit is segment A, digit 1. A set of eight bits is shifted in and then loaded into the digit one latches. The second set of 8 bits is loaded into digit two latches. The third set into digit three latches, and the fourth set is loaded into digit four latches.

CONTROL BITS

The fifth set of 8 data bits contains special segment data and control data in the following format:

SYNC	Q7	Q6	X	SP4	SP3	SP2	SP1
------	----	----	---	-----	-----	-----	-----

The first four bits shifted in contain the special character segment data. The fifth bit is not used. The sixth and seventh bits program the COP472-3 as a stand alone LCD driver or as a master or slave for cascading COP472-3's. BPC of the master is connected to BPA of each slave. The following table summarizes the function of bits six and seven:

Q7	Q6	Function	BPC Output	BPA Output
1	1	Slave	Backplane Output	Oscillator Input
0	1	Stand Alone	Backplane Output	Backplane Output
1	0	Not Used	Internal Osc. Output	Oscillator Input
0	0	Master	Internal Osc. Output	Backplane Output

The eighth bit is used to synchronize two COP472-3's to drive an 8½-digit display.

LOADING SEQUENCE TO DRIVE A 4½-DIGIT DISPLAY

Steps:

1. Turn \overline{CE} low.
2. Clock in 8 bits of data for digit 1.
3. Clock in 8 bits of data for digit 2.
4. Clock in 8 bits of data for digit 3.
5. Clock in 8 bits of data for digit 4.
6. Clock in 8 bits of data for special segment and control function of BPC and BPA.

0	0	1	1	SP4	SP3	SP2	SP1
---	---	---	---	-----	-----	-----	-----

7. Turn \overline{CS} high.

Note: \overline{CS} may be turned high after any step. For example to load only 2 digits of data, do steps 1, 2, 3, and 7.

\overline{CS} must make a high to low transition before loading data in order to reset internal counters.

LOADING SEQUENCE TO DRIVE AN 8½-DIGIT DISPLAY

Two or more COP472-3's may be connected together to drive additional segments. An eight digit multiplexed display is shown in Figure 7. The following is the loading sequence to drive an eight digit display using two COP472-3's. The right chip is the master and the left the slave.

Steps:

1. Turn \overline{CS} low on both COP472-3's.
2. Shift in 32 bits of data for the slave's four digits.
3. Shift in 4 bits of special segment data: a zero and three ones.

1	1	1	0	SP4	SP3	SP2	SP1
---	---	---	---	-----	-----	-----	-----

This synchronizes both the chips and BPA is oscillator input. Both chips are now stopped.

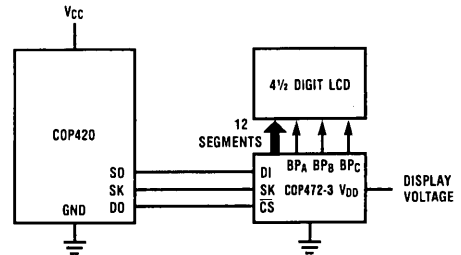
4. Turn \overline{CS} high to both chips.
5. Turn \overline{CS} low to master COP472-3.
6. Shift in 32 bits of data for the master's 4 digits.
7. Shift in four bits of special segment data, a one and three zeros.

0	0	0	1	SP4	SP3	SP2	SP1
---	---	---	---	-----	-----	-----	-----

This sets the master COP472-3 to BPA as a normal backplane output and BPC as oscillator output. Now both the chips start and run off the same oscillator.

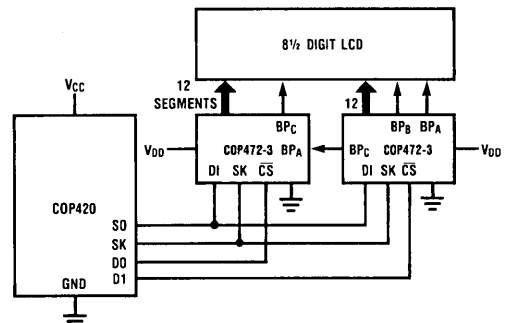
8. Turn \overline{CS} high.

The chips are now synchronized and driving 8 digits of display. To load new data simply load each chip separately in the normal manner, keeping the correct status bits to each COP472-3 (0110 or 0001).



TL/DD/6932-6

FIGURE 6. System Diagram - 4½ Digit Display



TL/DD/6932-7

FIGURE 7. System Diagram - 8½ Digit Display

Example Software

Example 1

COP420 Code to load a COP472-3 [Display data is in M(0, 12)-M(0, 15), special segment data is in M(0, 0)]

```

LOOP:      LBI 0, 12                ; POINT TO FIRST DISPLAY DATA
           OBD                    ; TURN CS LOW (DO)
           CLRA
           LQID                   ; LOOK UP SEGMENT DATA
           CQMA                   ; COPY DATA FROM Q TO M & A
           SC                     ; SET C TO TURN ON SK
           XAS                    ; OUTPUT LOWER 4 BITS OF DATA
           NOP                    ; DELAY
           NOP                    ; DELAY
           LD                     ; LOAD A WITH UPPER 4 BITS
           XAS                    ; OUTPUT 4 BITS OF DATA
           NOP                    ; DELAY
           NOP                    ; DELAY
           RC                     ; RESET C
           XAS                    ; TURN OFF SK CLOCK
           XIS                    ; INCREMENT B FOR NEXT DATA
           JP LOOP                ; SKIP THIS JUMP AFTER LAST DIGIT
           SC                     ; SET C
           LBI 0, 0               ; ADDRESS SPECIAL SEGMENTS
           LD                     ; LOAD INTO A
           XAS                    ; OUTPUT SPECIAL SEGMENTS
           NOP                    ;
           CLRA                   ;
           AISC 12                ; 12 to A
           XAS                    ; OUTPUT CONTROL BITS
           NOP                    ;
           LBI 0, 15              ; 15 to B
           RC                     ; RESET C
           XAS                    ; TURN OFF SK
           OBD                    ; TURN CS HIGH (DO)

```

Example Software (Continued)**Example 2**

COP420 Code to load two COP472-3 parts [Display data is in M(0, 12)-M(0, 15) and M(1, 12)-M(1, 15), special segment data is in M(0, 0) and M(1, 0)]

```

INIT:      LBI          0, 15
           OBD
           LEI          8           ; TURN BOTH CS'S HIGH
           RC
           XAS          ; TURN OFF SK CLOCK
           LBI          3, 15      ; USE M(3, 15) FOR CONTROL BITS
           STII         7         ; STORE 7 TO SYNC BOTH CHIPS
           LBI          0, 12      ; SET B TO TURN BOTH CS'S LOW
           JSR          OUT        ; CALL OUTPUT SUBROUTINE

MAIN DISPLAY SEQUENCE
DISPLAY   LBI          3, 15
           STII         8         ; SET CONTROL BITS FOR SLAVE
           LBI          0, 13      ; SET B TO TURN SLAVE CS LOW
           JSR          OUT        ; OUTPUT DATA FROM REG. 0
           LBI          3, 15
           STII         6         ; SET CONTROL BITS FOR MASTER
           LBI          1, 14      ; SET B TO TURN MASTER CS LOW
           JSR          OUT        ; OUTPUT DATA FROM REG. 1

OUTPUT SUBROUTINE
OUT:      OBD          ; OUTPUT B TO CS'S
           CLRA
           AISC         12        ; 12 TO A
           CAB          ; POINT TO DISPLAY DIGIT (BD = 12)
LOOP     CLRA
           LQID         ; LOOK UP SEGMENT DATA
           CQMA         ; COPY DATA FROM Q TO M & A
           SC
           XAS          ; OUTPUT LOWER 4 BITS OF DATA
           NOP          ; DELAY
           NOP          ; DELAY
           LD           ; LOAD A WITH UPPER 4 BITS
           XAS          ; OUTPUT 4 BITS OF DATA
           NOP          ; DELAY
           NOP          ; DELAY
           RC          ; RESET C
           XAS          ; TURN OFF SK
           XIS          ; INCREMENT B FOR NEXT DISPLAY DIGIT
           JP           LOOP      ; SKIP THIS JUMP AFTER LAST DIGIT
           SC          ; SET C
           NOP
           LD           ; LOAD SPECIAL SEGS. TO A (BD=0)
           XAS          ; OUTPUT SPECIAL SEGMENTS
           NOP
           LBI          3, 15
           LD           ; LOAD A
           XAS          ; OUTPUT CONTROL BITS
           NOP
           NOP
           RC
           XAS          ; TURN OFF SK
           OBD          ; TURN CS'S HIGH (BD = 15)
           RET

```



Section 7
**Microcontroller
Development Support**

7 Contents

Development Support	7-3
HPC16400 Microcontroller Development Support	7-5
HPC164800 Microcontroller Development System	7-12
HPC16400 Microcontroller Development System	7-22
HPC Software Support Package	7-35
ISDN Basic Rate Interface Software for the HPC16400 High Performance Data Communications Microcontroller	7-45



Section 7
**Microcontroller
Development Support**



Section 7 Contents

Development Support	7-3
COP400 Microcontroller Development Support	7-5
COP800 Development System	7-12
HPC Microcontroller Development System	7-22
HPC Software Support Package	7-35
ISDN Basic Rate Interface Software for the HPC16400 High Performance Data Communications Microcontroller	7-45

Development Support

Our job doesn't end when you buy a National microcontroller, it only begins.

The next step is to help you put that microcontroller to work—delivering real-world performance in a real-world application.

That's why we offer you such a comprehensive, powerful, easy-to-use package of development tools.

Microcontroller Development Support COP400 Family

The COP^{STM} Microcontroller Development system is a complete, inexpensive system, designed to support both hardware and software development of the COP400 family of microcontrollers.

Using a standard IBM® PC® platform as a host, this system provides the tools to write, assemble, debug and emulate software for user target design.

The development system itself consists of two circuit boards that interface with each other and to the host computer using a software package. The first board is called the Brain Board. It provides the major functional features of the system, linking the various elements of the host system. The other board is called the Personality Board and it is common for all members of the COP400 family of microcontrollers.

Microcontroller Development Support COP800 Family

MetaLink Corporation's iceMASTER™ COP8 Model 400 In-Circuit Emulator provides complete real-time full speed emulation of all COP8 family devices. It consists of a base unit and interchangeable probe cards, which support various configurations and packages. The source symbolic debugger with a window based user interface is a powerful tool to accomplish software and hardware debug and integration tasks.

COP800 code development is supported by a macro cross-assembler running DOS on the IBM compatible PC.

COP800 development is also supported with a low cost Designer's Kit. The Designer's Kit includes a simulator with a window based menu driven user interface and the COP8 cross-assembler. It is a tool designed for product evaluation and code development and debug. It comes equipped with complete debug capability and full assembler. The host for the designer kit is an IBM PC/XT/AT or compatible running DOS.

Microcontroller Development Support HPC™ Family

HPC-MDS is a complete packaged system for all members of the HPC family except for HPC46100. The host system is IBM PC/AT® (PC-DOS, MS-DOS) and Sun® SPARCstation (SunOS™). It provides true real time in-system emulation with support tools such as ANSI compatible C-Compiler, assembler, Linker and Source/Symbolic debugger. The debugger interface is based on MS-Windows 3.0 for IBM PC/AT and a line debugger for Sun SPARCstation users.

HPC-MDS gives the user the flexibility to symbolically debug his code and download it to the target hardware. The user can set breakpoints and traces, can execute time measurements and examine and modify internal registers and I/O.

A low cost HPC designer's kit is also available. The kit has complete in-system emulation capability and is packaged with an evaluation version of C compiler and full package of Assembler/Linker.

The HPC46100 DSP-Microcontroller, is supported by a development kit for ROM emulation, logic and timing analysis, code debug with inverse assembly and PC based debug monitor. The kit consists of a Logic Analyzer Interface Board, a Target Board, Assembler/Linker/Librarian software, an inverse assembler to run on Hewlett-Packard 1650 and 16500A/B logic analyzers and PC based debug monitor, "The Serial Hook".

Third Party development support is also available for various sources for the HPC family.

Hewlett Packard offers HP64775 emulator/analyzer for 30 MHz HPC 16083/16064 and 20 MHz 16400E emulation. The stand alone HP system provides a very fast serial link to the host system and offers complete emulation and timing and logic analysis capability. The software tools for HP emulator are provided by National Semiconductor®.

Signum System offers a USP-HPC in-circuit emulator for the HPC46100 with 40 MHz 1 wait state real time emulation. This system is supported with 256 kbyte overlay emulation memory, 32k frames deep trace buffer memory, complex breakpoints, high level language source/symbolic debugger, fast serial download and a window based menu driven user interface.

The language tools hosted on the IBM PC/AT and compatibles and Sun SPARCstation are available from National Semiconductor to support third party emulation systems.

Emulation Technology offers a passive preprocessor and inverse assembler package for HP1650 and 16500A series of Logic analyzers. The preprocessor provides a low cost and convenient way of doing timing and state analysis of the HPC based design.

Emulation Technology also offers debug tool accessories for 68-pin PLCC and 80-pin (QFP) Quad Flat Packages. This includes PLCC to QFP adapter, QFP test clip and a QFP surface mount replacement base.

Programming support for the HPC emulator devices is available from Data I/O on their Unisite models.

For more details on the third party support tools for NSC's microcontroller products, please contact the third party office in your area or the National Semiconductor sales office.

Dial-A-Helper On-Line Applications Support

Dial-A-Helper lets you communicate directly with the Microcontroller Applications Engineers at National.

Using standard computer communications software, you can dial into the automated Dial-A-Helper Information System 24 hours a day.

You can leave messages on the electronic bulletin board for the Applications Engineers, then retrieve their responses.

You can select and then download specific applications data.

Dial-A-Helper

Voice: (408) 721-5582 (8 a.m.-5 p.m. PST)

Modem: (408) 739-1162 (24 Hrs./day)

Setup: Baud rate 300 bps or 1200 bps 8 bits, no parity, 1 stop

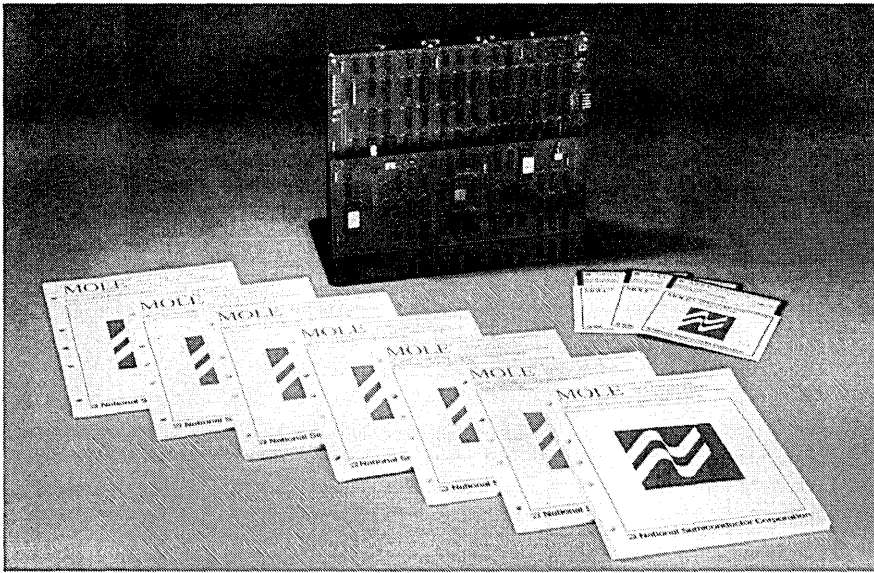
Dedicated Applications Engineers

We've assembled a dedicated team of highly trained, highly experienced engineering professionals to help you implement your solution quickly, effectively, efficiently and to ensure that it's the best solution for your specific application.

At National, we believe that the best technology is also the most usable technology. That's why our microcontrollers provide such practical solutions to such real design problems. And that's why our microcontroller development support includes such comprehensive tools and such powerful engineering resources.

No one makes more microcontrollers than National and no one does more to help you put those microcontrollers to work.

COP400 (COPS™ DS) Microcontroller Development Support



TL/DD/8630-14

Development Tools

The NSC Microcontroller On Line Emulator Development System is designed to support the development of NSC COPS Microcontroller products. This system provides effective support for the development of both software and hardware in Microcontroller-based applications.

A system consists of three components: a Brain Board, a Personality Board, and software for a host computer. The host may be an IBM®-PC, or one of a number of inexpensive PC compatibles. The cross-assemblers and debugger provided by National Semiconductor will run under control of the host computer MS-DOS operating system.

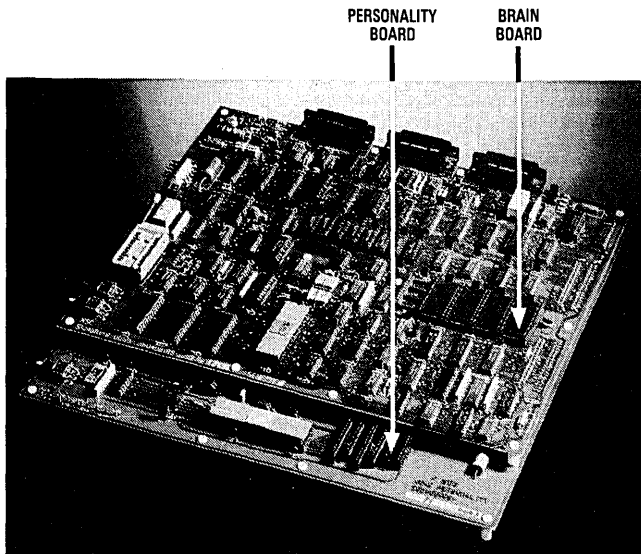
The Brain Board provides the development system with the capability of communicating with the user's Host CPU. Resident firmware on the Brain Board allows the user to download assembled load modules over the RS-232 link from the host computer, display and alter code initiate Breakpoints, Traces, and timing on addresses and external events, examine and modi-

fy the internal resources of the microcontroller being emulated. The Brain Board also provides all the hardware and firmware to program standard EPROMs up to 27256's (32k x 8).

The Personality Board supports the emulation of the COP 400 family of microcontrollers.

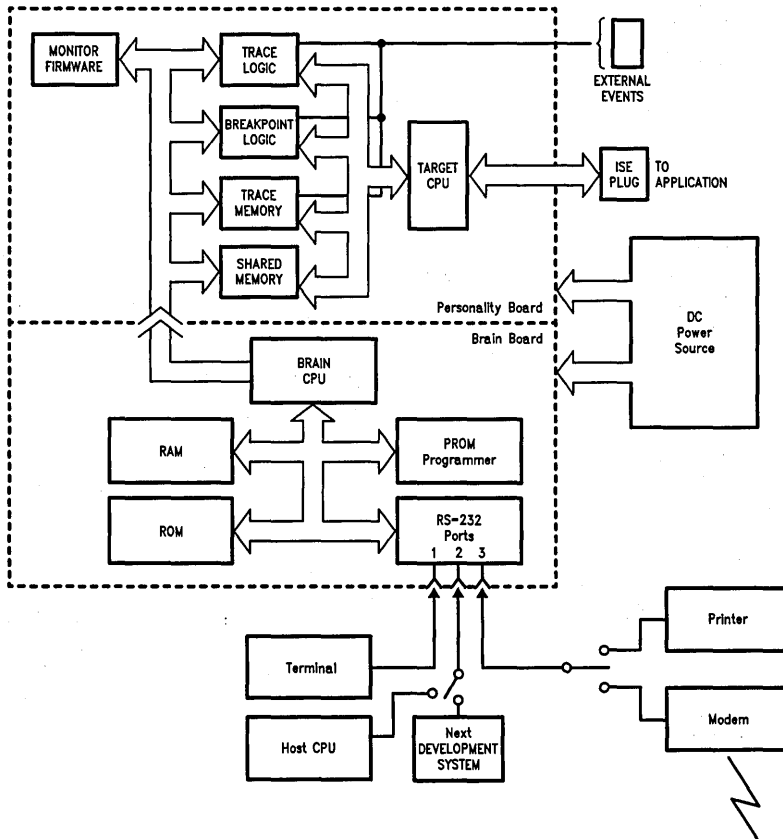
The host CPU contributes cost effective bulk storage and high speed processing. Disk editing and assembly operations are controlled by the host CPU. The results are down loaded to the Brain Board over the RS-232 link.

The Microcontroller On-Line Emulator Development System concept provides the user with a powerful development system based around a familiar host. The Brain Board/Personality Board/Host combination provides FULL emulation capability. This modular design provides maximum flexibility and maximum utility for the development of Microcontroller based systems.



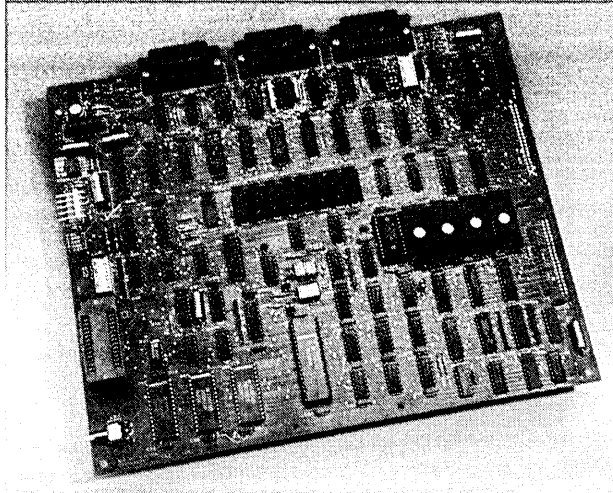
TL/DD/8830-3

System Block Diagram



TL/DD/8830-2

Brain Board



TL/DD/6830-19

General Description

The Brain Board is the pivotal component of the development system concept. In conjunction with a terminal and Personality Board it provides the user with a freestanding workstation for Microcontroller emulation. It ties the system together by communicating with the Personality Board, printers, modems, optional host computer, and other Brain Boards. Multiple Brain Boards, tied to a common host, can function as emulators for individual projects where each Brain Board is a separate workstation. They can also function as individual Microcontroller emulators within a multicontroller system.

The Brain Board utilizes a NSC800™ Microprocessor with 64k RAM and firmware ROM. It has an EPROM programmer for on-line changes. There are three RS-232 ports and a bus to connect the Brain to the Personality Board for actual emulation of code in the user's application system.

The RS-232 ports are used via the communication routines in firmware to interface with a host computer, terminal, modem, printer, or other development systems, for greater flexibility during system development.

The development system firmware is controlled by an EXEC. There are three major sets of EXEC commands. The first set of commands are calls to other main programs. These are:

COMM	Invoke Communications Program
DIAG	Invoke Diagnostics Program
MONITOR	Invoke Personality Emulation Monitor
PROG	Invoke PROM Programming Program

The second set of EXEC commands are:

CALC	Adds/Subtracts decimal and hex numbers
COMPARE	Compares one buffer with another
ERASE	Used to erase all or part of a buffer
HELP	Prints a summary of EXEC commands
MOVE	Moves data from one buffer to another
STATUS	Display status of buffers, display and alter RS-232 parameters

The third set of commands are used exclusively for multiple system configurations and they are:

CONNECT	Connect the user with the requested system
DISCONNECT	Disconnects the system
IDENT	Identifies the system

The Brain Board supports NSC's COP4 development system Personality board.

Features

- Single 5V operation
- Ability to interface to host computers
- Full communication control of other systems with host computer and a modem
- Three RS-232 ports
- Auto baud selection (110, 300, 600, 1200, 2400, 4800, 9600, 19200 baud)
- Self diagnostics

Features (Continued)

- Program EPROMS
 - MM2716, NMC27C16
 - MM2732, NMC27C32
 - NMC2764
 - NMC27256

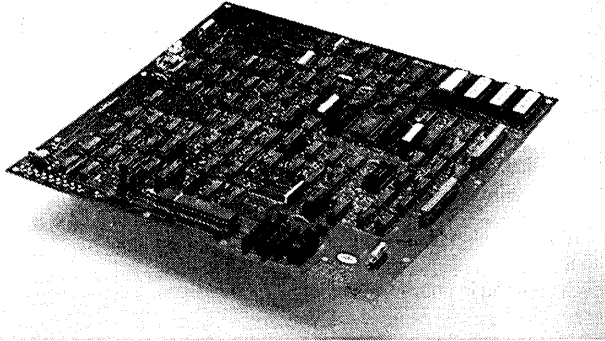
PHYSICAL SIZE
10" x 12"

POWER REQUIREMENTS

+5V DC @ 3.5A
+12.5/ +21V or +25V @ 50 mA
(Optional—required only for PROM programming)

COP400 Family Personality Board

COPS Personality Board



TL/DD/8830-6

General Description

The COPS Family Personality Board supports the emulation of COP400 family of Microcontrollers. The Personality Board allows the user to emulate the appropriate Microcontroller in the user's end system for fast development of application code and hardware. The Personality Board consists of: a Monitor, the hardware to control the operation of the Microcontroller in the emulation system, and an emulation cable to connect the emulator to the application system. The cable has the same pin configuration as the final masked part.

The Personality Board Monitor is contained in firmware ROM, contains an assembler and disassembler and is directly executable by the NSC800 on the Brain Board. The Monitor commands will allow the user to execute the application code, examine and modify internal registers and I/O, examine and alter object code in hex or mnemonic format, execute Time measurements, and set Trace and Breakpoints.

The Personality Board also contains 2k bytes of shared memory (RAM) for application code and the necessary hardware for Trace and Breakpoint operation.

Features

- Supports entire COPS CMOS and NMOS family
- Single 5V operation
- Firmware monitor
- Firmware diagnostics
- Firmware Line-by-Line Assembler and Unassembler
- 2k bytes of shared memory
- 256 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges, or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Features (Continued)**Common Monitor Commands**

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
Dlagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and development system Status
Time	Time program execution or external events
TRace	Specify triggers for capturing Trace data
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory
Chip	Specify COP device to emulate
Option	Specify COP chip options being emulated
Set	Set special emulation options

PHYSICAL SIZE

12" x 12"

POWER REQUIREMENTS

+ 5V @ 3.5A

COP400 Development System

HOST SYSTEM REQUIREMENTS

IBM PC-XT®/PC-AT® or compatibles, 640 kbytes memory with 5.25" double density floppy drive.

RS-232 Serial port

MS-DOS or PC-DOS operating system

Power Supply for emulator operation +5V DC

For EPROM Programming 12.5V or 21V DC

Development Tools Selection Table

Microcontroller	NSID	Description	Includes	Manual Number
COP400 Family	MOLE-BRAIN	Brain Board	Brain Board User's Manual, RS-232 Cable, Power Cable	420408188-001
	MOLE-COPS-PB1	Personality Board	COP400 Personality User's Manual COP400 User's Manual Emulator Cables 20 DIP, 24 DIP, 28 DIP	420408189-001
				424410284-001
MOLE-COPS-IBM	Assembler Software for IBM	COP400 Software User's Manual and Software PC-DOS Comm. User's Manual	424409479-001 420040416-001	

EMULATOR DEVICES

COP4 family provides Piggy-back devices for form, fit and function of the COP4XX products in 28-lead DIP packages.

NSID	Package	Description	Emulates
COP420P	28-Lead DIP	Piggy-back	COP420/421/422
COP444CP	28-Lead DIP	Piggy-back	COP410C/411C/413C, COP424C/425C/426C, COP444C/445C
COP444LP	28-Lead DIP	Piggy-back	COP410L/411L/413L/COP420L/421L/422L, COP444L/445L

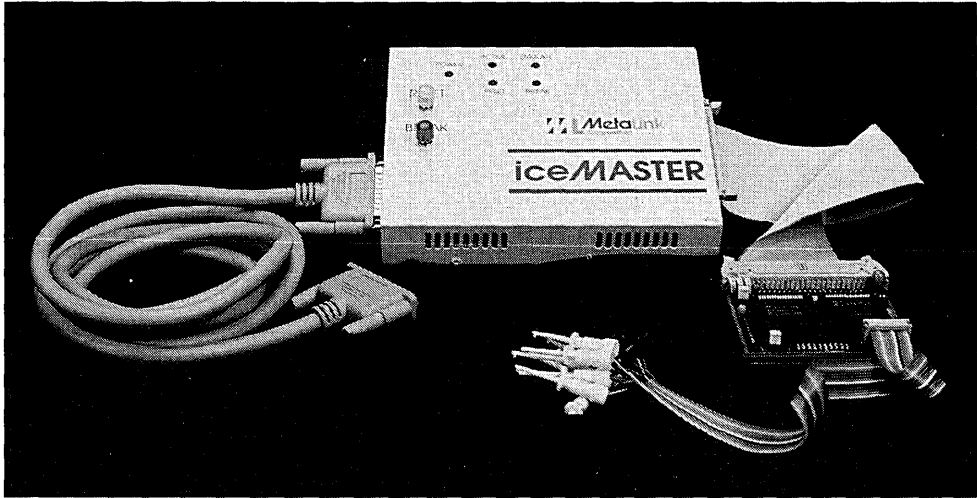
ACCESSORIES

NSID	Part Type	Description
MOLE-CBL-20DIP	20-Pin DIP Cable	Cable Used In-System Emulation of COP4
MOLE-CBL-24DIP	24-Pin DIP Cable	Cable Used In-System Emulation of COP4
MOLE-CBL-28DIP	28-Pin DIP Cable	Cable Used In-System Emulation of COP4

 National Semiconductor

COP800 Development System

iceMASTER™ COP8/400



TL/DD/11386-1

Product Overview

The iceMASTER COP8/400 in-circuit emulator manufactured by MetaLink Corporation and marketed by National Semiconductor provides complete real-time emulation support for all members of the COP8 family. This stand-alone system is designed to provide maximum flexibility to the user through the interchangeable probe cards to support the various configurations and packages of the COP8 family. The interchangeable probe card connects to a common base unit which is linked with an IBM® PC® host through the RS-232 serial communications channel. Full assembly-level symbolic debugging is supported.

MetaLink COP8 iceMASTER Feature List

- Flexible, easy-to-use windowed interface, with window size, position, contents and color being completely configurable.
- Fast serial download with 115.2 kBaud using a standard PC COMM port.
- Context-sensitive hypertext on-line help system.
- Commands can be accessed via pull-down menus and/or redefinable hot keys.
- Dynamically annotated code feature displays contents of all accessed (read and write) memory locations and registers, as well as flow-of-control direction change markers next to each instruction executed when single-stepping.
- 4k-frame trace buffer captures data in real-time. Trace information consists of address and data bus values and user-selectable probe clips (external event lines). Trace buffer data can be viewed as raw hex or disassembled instructions. The probe clip bit values can be displayed in binary, hex or digital waveform formats.
- Performance analyzer with a resolution better than 6 μ s. Up to 15 independent memory areas based on code address, line number or label ranges can be defined. Analysis results can be viewed in bar graph format or as actual frequency count.
- 32k of break and trace triggers. Triggers can be enabled, disabled, set or cleared. They can be simple triggers based on code or address ranges or complex triggers based on code address, direct address, opcode value, opcode class or immediate operand. Complex breakpoints can be ANDed and ORed together.
- Memory operations for program memory include single-line assembler, disassembler, view, change, and write to file.
- Memory operations for data memory include fill, move, change, compare, dump to file and examine, modify for registers and program variables.
- Complete status of debugger including breakpoints, trace triggers, etc. can be saved to file for later resumption of debugging process.

Specifications

EMULATOR SYSTEM REQUIREMENTS

Basic Emulator System Model 400
Interchangeable Probe Card
+5V, 1.5A Power Source

MODELS

400 Emulator with:
4k Trace Buffer
2 Performance Analyzers
Full WATCHDOG™ Timer Support

FILE FORMATS

Intel HEX and National Semiconductor

MACRO

Repetitive Routines
User-created and callable

MEMORY OPERATIONS

Program Memory:
Single Line Assembler
Disassemble
Disassemble to File
View/Change
Mapping

Data/Code Memory:

Dump
Dump to File
Fill
Move
Change
Compare

Registers:

Examine/Modify

Program Variables:

Examine/Modify

OPERATING CHARACTERISTICS

Electrically Transparent
Operationally Transparent

USER INTERFACE

Keyboard or Mouse Control
Pull-Down and Pop-Up Menus
Main Screen Windows:
Registers/SFRs/PSW Bits
Stack
Up to 5 Internal Data Memory
Up to 5 Code Memory
Source Program
Watch
System Status

User Window Controls:

Selectable (On/Off)
Movable
Resizable
Scrollable
Color Selection
Highlighting

Function/Hot Key Access:

User-Assignable

EMULATION CONTROLS

Reset from Emulator
Reset from Target
Reset Processor
Go
Go From
Go Until
Slow Motion
Step
Step Line
Step Over
Step To
Repetition Counter

PERFORMANCE ANALYZER

Real-Time Program Profiling
5.4 μ s Sampling Period
7 Year Duration
Display Options:
Bar Graph
Frequency Count
Display Modes:
Raw
Symbolic
Up to 15 Bin Capacity:
Multiple Ranges per Bin
User-Controlled Bin Setup:
By Address
By Symbol
Automatic

TRACE

Trace Triggers:
Start
Center
End
Variable
4k-Frame Trace Buffer

Specifications (Continued)

Trace Contents:

Address
Data
External Clips

Trace Display Modes:

Raw Hex
Symbolic
Binary (Clips)
Digital Waveform (Clips)

Trace Buffer Operations:

Write Buffer to File
Search Trace Buffer

HELP

On-Line
Context Sensitive
Hypertext/Hyperlinked

SOURCE/SYMBOL SUPPORT

Source-Level Debug

ELECTRICAL SPECIFICATIONS

Input Power (Maximum):
1.5A @ +5 V_{DC} ±5%

MECHANICAL SPECIFICATIONS

Emulator Dimensions:
1.0" x 7.0" x 5.5"
(2.5cm x 17.8cm x 14cm)
Probe Card Cable Length:
14.0" (35.6cm)
Emulator Weight:
2.0 lbs. (0.9 kg)

WARRANTY

One (1) year limited warranty, parts and labor, for registered users.

IceMASTER COP8	
Emulation Memory	
Program	32k
Real Time:	DC - 10 MHz
Breakpoints:	32k
Trace On:	32k
Trace Off:	32k
Pass Count	32K
Trigger Conditions:	
PC Address and Range	X
Opcode Value	X
Opcode Class	X
SFRs/Registers	X
Direct Byte Address and Range	X
Direct Bit Address and Range	X
Immediate Operand Value	X
Read/Write to Bit Address	X
Register Address Modes	X
Read/Write to Register Address	X
Logical AND/OR of	X
Any of the Above	
External Input	X
Operating Modes	
Single-Chip/ROM	X

HOST SYSTEM REQUIREMENTS

IBM PC-XT/PC-AT or compatibles, 640 kbytes of Memory with 5.25" Double Density Floppy Drive.
RS-232 Serial Port
MS-DOS or PC-DOS Operating System

Ordering Information

Emulator Ordering Information

Part Number	Description
IM-COP8/400	MetaLink base unit in-circuit emulator for all COP8 devices, symbolic debugger software and RS-232 serial interface cable
MHW-PS3	Power Supply: 110V/60 Hz
MHW-PS4	Power Supply: 220V/50 Hz

Probe Card Ordering Information

Device	Package	Voltage Range	Probe Card
COP880C, 8780C	44 PLCC	4.5V-5.5V	MHW-880C44D5PC
		2.5V-6.0V	MHW-880C44DWPC
COP880C, 8780C	40 DIP	4.5V-5.5V	MHW-880C40D5PC
		2.5V-6.0V	MHW-880C40DWPC
COP881C, 8781C, 840C, 820C	28 DIP	4.5V-5.5V	MHW-880C28D5PC
		2.5V-6.0V	MHW-880C28DWPC
COP842C, 822C, 8742C	20 DIP	4.5V-5.5V	MHW-880C20D5PC
		2.5V-6.0V	MHW-880C20DWPC
COP820CJ	28 DIP	4.5V-5.5V	MHW-820CJ28D5PC
		2.3V-6.0V	MHW-820CJ28DWPC
COP822CJ	20 DIP	4.5V-5.5V	MHW-820CJ20D5PC
		2.3V-6.0V	MHW-820CJ20DWPC
COP8640C, 8620C	28 DIP	4.5V-5.5V	MHW-8640C28D5PC
		2.5V-6.0V	MHW-8640C28DWPC
COP8642C, 8622C	20 DIP	4.5V-5.5V	MHW-8640C20D5PC
		2.5V-6.0V	MHW-8640C20DWPC
COP888CF	44 PLCC	4.5V-5.5V	MHW-888CF44D5PC
		2.5V-6.0V	MHW-888CF44DWPC
COP888CF	40 DIP	4.5V-5.5V	MHW-888CF40D5PC
		2.5V-6.0V	MHW-888CF40DWPC
COP884CF	28 DIP	4.5V-5.5V	MHW-884CF28D5PC
		2.5V-6.0V	MHW-884CF28DWPC
COP888CL	44 PLCC	4.5V-5.5V	MHW-888CL44D5PC
		2.5V-6.0V	MHW-888CL44DWPC
	40 DIP	4.5V-5.5V	MHW-888CL40D5PC
		2.5V-6.0V	MHW-888CL40DWPC

Ordering Information (Continued)**Probe Card Ordering Information** (Continued)

Device	Package	Voltage Range	Probe Card
COP884CL	28 DIP	4.5V-5.5V	MHW-884CL28D5PC
		2.5V-6.0V	MHW-884CL28DWPC
COP888CG, 888CS	44 PLCC	4.5V-5.5V	MHW-888CG44D5PC
		2.5V-6.0V	MHW-888CG44DWPC
	40 DIP	4.5V-5.5V	MHW-888CG40D5PC
		2.5V-6.0V	MHW-888CG40DWPC
COP884CG, 884CS	28 DIP	4.5V-5.5V	MHW-884CG28D5PC
		2.5V-6.0V	MHW-884CG28DWPC

LANGUAGE TOOLS

Product	NSID	Description	Includes	Number
COP800 Family	MOLE-COP8-IBM	Assembly Language Software for the COP800 Family	COP800 System Software User's Manual	424410527

Single-Chip Emulator**Form, Fit, Function Emulator Ordering Information**

Part Number	Emulator		Clock Option	Description	
	Part Number	Package			
COP880C	COP880CMHEL-X	44 LDCC	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable	
	COP8780CV	44 PLCC	Programmable	One-Time Programmable	
	COP8780CEL	44 LDCC		UV Erasable	
	COP880CMHD-X	40 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable	
	COP8780CN			Programmable	One-Time Programmable
	COP8780CJ				UV Erasable
COP881C, COP840C, COP820C	COP881CMHD-X	28 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable	
	COP8780CN			Programmable	One-Time Programmable
	COP8780CJ				UV Erasable

Single-Chip Emulator (Continued)

Form, Fit, Function Emulator Ordering Information (Continued)

Part Number	Emulator		Clock Option	Description
	Part Number	Package		
COP881C, COP840C, COP820C	COP881CMHEA-X	28 LCC (Shoebbox)	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
	COP8781CWN	28 SO	Programmable	One-Time Programmable
	COP8781CMC			UV Erasable
COP842C	COP842CMHD-X	20 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable
COP822C	COP822CMHD-X			
COP842C, COP822C	COP8742CN	20 DIP	Programmable	One-Time Programmable
	COP8742CJ			UV Erasable
	COP8742CWM	20 SO	Programmable	One-Time Programmable
	COP8742CMC			UV Erasable
COP8640C, COP8620C	COP8640CMHD-X	28 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable
	COP8640CMHEA-X	28 LCC (Shoebbox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
COP8642C, COP8622C	COP8642CMHD-X	20 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable
COP820CJ	COP820CJMHD-X	28 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable
	COP820CJMHEA-X	28 LCC (Shoebbox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
COP822CJ	COP822CJMHD-X	20 DIP	X = 1: Crystal X = 2: External X = 3: R/C	Multi-Chip Module, UV Erasable
COP888CL	COP888CLMHEL-X	44 LDCC	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP888CLMHD-X	40 DIP		
COP884CL	COP884CLMHD-X	28 DIP	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP884CLMHEA-X	28 LCC (Shoebbox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
COP888CF	COP888CFMHEL-X	44 LDCC	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP888CFMHD-X	40 DIP		
COP884CF	COP884CFMHD-X	28 DIP	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP884CFMHEA-X	28 LCC (Shoebbox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
COP888CG	COP888CGMHEL-X	44 LDCC	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP888CGMHD-X	40 DIP		

Single-Chip Emulator (Continued)**Form, Fit, Function Emulator Ordering Information** (Continued)

Part Number	Emulator		Clock Option	Description
	Part Number	Package		
COP884CG	COP884CGMHD-X	28 DIP	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP884CGMHEA-X	28 LCC (Shoebox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
COP888EG	COP888EGMHEL-X	44 LDCC	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP888EGMHD-X	40 DIP		
COP884EG	COP884EGMHD-X	28 DIP	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP884EGMHEA-X	28 LCC (Shoebox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable
COP888CS	COP888CSMHEL-X	44 LDCC	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP888CSMHD-X	40 DIP		
COP884CS	COP884CSMHD-X	28 DIP	X = 1: Crystal X = 3: R/C	Multi-Chip Module, UV Erasable
	COP884CSMHEA-X	28 LCC (Shoebox)		Multi-Chip Module, Same Footprint as 28 SO, UV Erasable

Programming Support

The main board and scrambler boards can be purchased separately or as a set. The table below lists the product identification numbers of the Duplicator Board products.

Product ID	Description
COP8-PRGM-28D	COP8 Duplicator Board for 28-pin DIP Multi-Chip Module (MCM) and for use with Scrambler Boards
COP8-SCRM-DIP	MCM-Scrambler Board for 20-pin DIP and 40-pin DIP
COP8-SCRM-PCC	MCM-Scrambler Board for 44-pin PLCC/LDCC
COP8-PRGM-DIP	COP8 Duplicator Board with DIP MCM Scrambler Board (PRGM-28D and SCRM-DIP)
COP8-PRGM-PCC	COP8 Duplicator Board with PLCC/LDCC MCM Scrambler Board (PRGM-28D and SCRM-PCC)
COP8-SCRM-87A	Scrambler Board for COP8780 devices, 28-pin DIP, 40-pin DIP, 28-pin SO

Product ID	Description
COP8-SCRM-87B	Scrambler Board for COP8780 devices, 20-pin DIP, 20-pin SO, 44-pin PLCC/LDCC
COP8-PRGM-87A	COP8 Duplicator Board with COP8-SCRM-87A Scrambler Board
COP8-PRGM-87B	COP8 Duplicator Board with COP8-SCRM-87B Scrambler Board
COP8-PRGM-SBX	COP8 Duplicator Board with COP8-SCRM-SBX Scrambler Board
COP8-SCRM-SBX	Scrambler Board for 28-pin LCC MCM Package (Shoebox)

Programming Support (Continued)

The COP device pin/package types, COP device numbers, and the Duplicator Board product identification number for each package type are listed in the table below.

Package Type	COP Devices	COP Duplicator Product ID #
20-Pin DIP	842CMH, 8642CMH, 822CJMH	COP8-PRGM-DIP
28-Pin DIP	884CLMH/CFMH/CGMH/EGMH/CSMH, 881CMH, 8640CMH, 820CJMH	COP8-PRGM-28D
28-Pin LCC (Shoebox)	881CMH, 820CJMH, 8640CMH, 884CFMH/CLMH/CGMH/EGMH/CSMH	COP8-PRGM-SBX
40-Pin DIP	888CLMH/CFMH/CGMH/EGMH/CSMH, 880CMH, 943CMH	COP8-PRGM-DIP
44-Pin PLCC/LDCC	888CLMH/CFMH/CGMH/EGMH/CSMH, 880CMH	COP8-PRGM-PCC
28-Pin DIP or SO, 40-Pin DIP	8780C, 8781C	COP8-PRGM-87A
20-Pin DIP or SO, 44-Pin PLCC/LDCC	8780C, 8742C	COP8-PRGM-87B

COP800 DESIGNER'S TOOL KIT



TL/DD/11386-2

General Description

The COP800 Designer's Tool Kit is available today to help you evaluate National's COP800 microcontroller family. The Kit contains programmer's manuals, device data sheets, application notes, and pocket reference guides for immediate in-circuit evaluation. The Designer Kit includes an assembler and simulator, which allow you to write, test and debug COP800 code before your target system is finalized.

The simulator can handle script files that simulate hardware inputs and interrupts to the device being simulated. Any simulator command and comments may be included in a script file. The simulator also supports an additional command called WAIT, used to simulate machine cycles to delay before continuing with the script file.

A capture file feature enables you to record current cycle count and changes to an output port which are caused by the program under test. When used in combination with script files, this feature provides powerful software testing and debug capability.

Features

- Software simulator
- Assembler
- Programmer's manuals
- Device data sheets
- Application notes
- Assembler manual
- Tool kit user's guide
- Pocket reference guides
- COP8 SIM user's guide

Features (Continued)**Simulator Commands**

@RAM [ramadd]	Causes a break in execution to occur when a write to the specified RAM location is attempted.	LISTON	Turns on screen listing during stepping.
ASM [add]	Assembles directly to ROM at specified address or starting at last address used by command.	LISTOFF	Turns off screen listing.
BR [add]	Set breakpoint at the indicated ROM address.	LOAD filename	Loads Intel hex format file into simulator.
CAPTURE fname	Saves all hardware outputs in the file specified.	PRINTON	Sends all debug output to printer.
CAPTUREOFF	Stops capture and closes capture file.	PRINTOFF	Stops sending debug output to printer.
CY n	Sets cycle counter.	RAM add [n]	Sets RAM location at indicated address to value specified.
DASM [add]	Disassembles memory to screen starting at specified address or last location disassembled.	REG	Shows register status in debug window.
EVAL n [op] [n]	Evaluates input in decimal, hex, and binary. Can do simple calculations where op may be +, -, /, or *.	RESET	Simulates a hardware reset.
GO [add] [add]	Sets breakpoint at second address. Go from first address.	RESTORE fname	Restores simulator state from a file created with the SAVE command.
GOTIL add	Go from the current PC until the PC = add.	ROM add [n]	Sets ROM location at indicated address to value specified.
		SAVE filename	Saves the simulator state in the specified file.
		SCRIPT fname	Executes a script file.
		STEP [n]	Single step execution of n instructions.
		STEPTIL add	Single step until the PC = add.
		QUIT, EXIT	Return to DOS.

Ordering Information

NSID	Description	Includes:
COP8-TOOL-KIT	COP800 Designer's Tool Kit	Software Simulator Assembler Programmer's Manual Assembler Manual Tool Kit User's Guide


National Semiconductor
PRELIMINARY

HPC™ Microcontroller Development System



TL/DD/11211-1

General Description

The HPC Microcontroller Development System provides an optimized environment for real time emulation and software debugging for High Performance Controller (HPC) based designs. The system features National's powerful C and Assembly Source/Symbolic debugger. Development software consisting of an ANSI compatible C compiler/Assembler/Linker is hosted on an IBM® PC-AT® class computer as well as Sun® SPARCstations, with automatic download capability provided by the Source debugger through serial link to user target system. Microsoft™ Window 3.0 is used as the user interface for IBM hosts. A line debugger is supported for Sun SPARCstation under Sunview. HPC-MDS gives you complete control over hardware/software development, integration and debug.

The Source/Symbolic debugger provides a pull down menu, multiple windows and on-line help to make the system easy to learn and use. The Source/Symbolic debugger supports real-time transparent emulation to 20 MHz, one wait state for the entire HPC microcontroller product family. The Source/Symbolic debugger

is also equipped with interactive tutorials which would allow faster learning of the HPC-MDS environment.

The Real time trace capability is fully supported by the Source/Symbolic debugger. The debugger will allow complete trace and breakpoint triggering on C or Assembly Source statements, labels, symbols or line numbers. The system offers 8 hardware breakpoints with capability to break on multiple addresses, address ranges or external events.

Features

- 20 MHz 1 wait state real time emulation
- 2k x 48-bit Trace Memory
- 8 hardware breakpoints with break on 8 multiple addresses, address ranges or external events
- 64k x 8-bit emulation memory in 4k bytes mapping
- Real time trace
- External input events captured in trace
- Single line assembly/disassembly
- Modular designed POD cable for optimum AC emulation and ease of upgrading
- Fully supports development of system using HPC in Extended Memory mode
- Complete Source/Symbolic debug
- Graphical user interface (MS-WINDOWS)

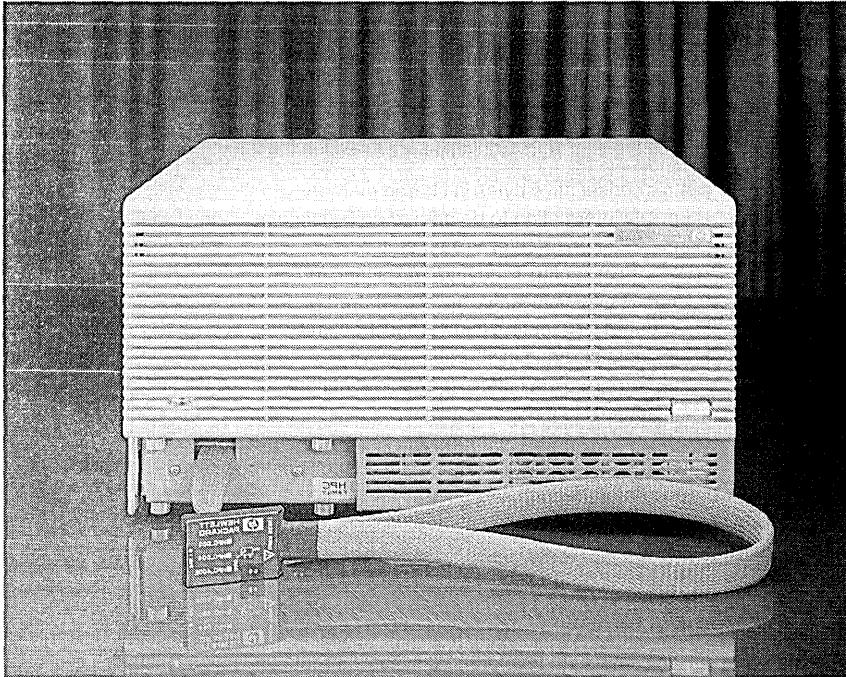
Physical Dimensions

Weight:	22 lbs	Height	15½"	Depth	17½"
Width	7⅞"				
Cable Length:		Emulator to POD		29¾"	
		POD to Target		3⅜"	

HPC-MDS Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
Dlagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and development system Status
Tlme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory
AlterWord	Alter consecutive words in shared memory
BAnk	Specify bank trigger information
CHip	Select chip and specify system memory map
DepositWord	Deposit word value in range of shared memory
End	Exit Monitor and return to Exec
ERror	Enable/disable HPC access error checking
EXclusion	Specify address ranges to exclude from Trace
FindWord	Find word values in shared memory
ListWord	List shared memory or memory range as words
MAp	Specify address range of memory on-board development system
XMove	Move data from one address range to another

HP 64700 Series Emulators/Analyzers for National Semiconductor HPC16003, 16083, 16004, 16064, 16400E



TL/DD/11211-3

Real-Time, Transparent Emulation and Analysis

Description

HP 64700 Series Emulators/Analyzers provide real-time, transparent emulation and analysis for National Semiconductor's HPC family of 16-bit microcontrollers.

Model 64775 emulator/analyzer is a self-contained emulation and analysis vehicle tuned for development of HPC16003, 16083, 16004, 16064, and 16400E based systems. HP 64775 emulator/analyzer focuses on powerful, nonintrusive analysis to trace complex program flow and characterize overall system performance. Choice of user interface plus high-speed program download maximize system integration efficiency. Reliable target system connection is made through a slim, flexible 1.5-foot cable ending in a PLCC probe, with an adapter for PGA.

Software development tools from National Semiconductor include a C compiler, assembler, and linker hosted on both the PC and the HP 9000 Series 300 workstation. A PC-hosted debugger is also available, which runs under Microsoft Window 3.0.

Powerful real-time, nonintrusive logic analysis functions use HP's "logic analyzer on-a-chip" technology.

A 48-channel state analyzer traces address, data, and status conditions.

In expanded memory applications, addresses can be treated as a 4-bit bank code plus a 16-bit offset in user-selectable bank-switch models. Eight hardware and 32 software breakpoints give the user enhanced control over the analysis of complex instruction expressions, ranges, and sequences without altering program code. An optional 16-channel external analyzer allows for a choice of synchronous or asynchronous signal analysis.

Emulation Features

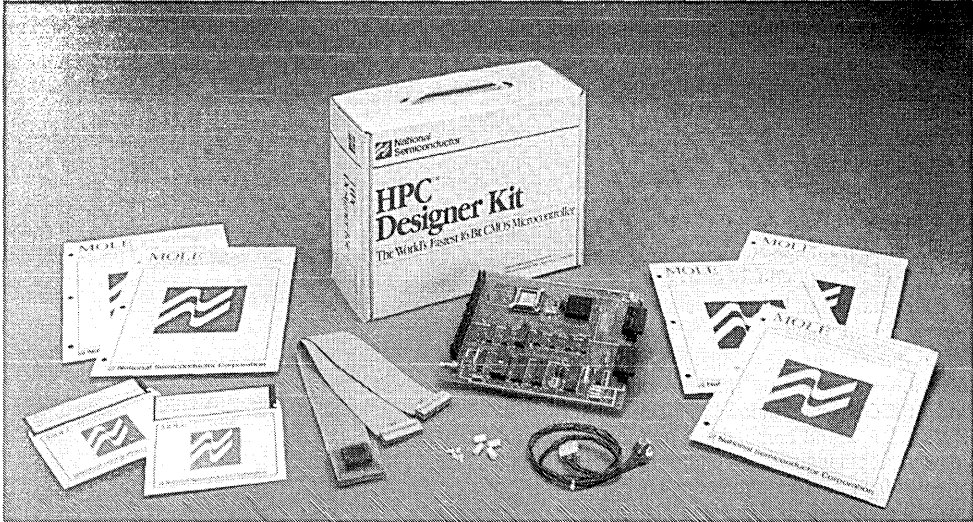
- Real-time, zero-wait-state operation to 20 MHz with 32k bytes of emulation memory for internal ROM support (not available for HPC16400E)
- 128k bytes of real-time emulation memory for external memory support to 30 MHz (20 MHz for HPC16400E)
- Mapping terms of RAM, ROM, or guarded memory accesses
- Target/emulation memory mappable in 256 byte blocks
- 8 real-time hardware breakpoints

Emulation Features (Continued)

- 32 software breakpoints
- Hybrid foreground/background monitor with shared NMI entry; both modifiable with user-supplied routines
- Full support for user-selectable bank-switching models—20-bit addressing range
- Emulation support for 8- or 16-bit memory and 16-bit on-chip memory
- 1.5 ft, slim, flexible cable ending in a PLCC emulation probe, with PGA adapter
- Full coordinated measurement bus for synchronizing or cross triggering up to 32 HP 647xx emulators
- 48 channels of state analysis with optional 16-channel external state/timing analysis
- Choice of user-interface software: firmware-based ASCII interface for true host-independent operation; MS-DOS multiwindow interface on IBM PC, HP Vectra PC, and compatibles enhances ease of use with popular low-cost platforms; or HP 9000 Series 300 computer offers the advantages of a powerful multiuser, multi-tasking platform
- For more Information call 1-800-447-3282

HPC Designer's Kits

HPC Designer's Kits



TL/DD/11211-2

General Description

The HPC Designer's Kit is a 16-bit microcontroller Development System for program development and real-time emulation. An on-board HPC microcontroller executes monitor firmware and also acts as the target processor.

When used as the target processor, all of the features of the HPC are available for use in the application. All operating modes of the HPC are supported, with up to 8k bytes of addressable memory available for application programs.

This kit contains all of the components, manuals, and software to design an HPC system. Just add an IBM or compatible PC, +5V DC 1.0A power supply and RS232 cables.

The development package has a complete Assembler/Linker/Librarian with no code limitations and an evaluation C-Compiler.

Features

- Supports HPC microcontroller family
- Single 5V operation
- Firmware monitor directly executed by the HPC
- Firmware diagnostics directly executed by the HPC
- Firmware Line-by-Line Assembler and Unassembler
- 8k bytes of user program memory
- Breakpoint on multiple addresses
- List and alter memory
- Print and modify internal registers
- Singlestep
- Real time emulation
- Evaluation module that allows up to 1000 lines of code to be developed for evaluation purposes

Emulation Features**HPC Development Board Monitor**

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
BAud	Set or display the host or terminal Baud rate
BYpass	Connect terminal port to host port
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint function
Deposit	Deposit byte value into range of shared memory
Dlagonstic	On-board test routine for system checkout
Go	Start program execution
Help	On-screen Help menu
List	List data in shared memory
ListUnassemble	List shared memory in mnemonic form
LOad	Load hex object file from terminal or host port
Modify	Modify on-chip RAM or Registers during Breakpt
ModifyByte	Modify on-chip RAM or registers as bytes
ModifyWord	Modify on-chip RAM or registers as words
Put	One-line assembler
Restart	Restart HPC chip
Singlestep	Execute one instruction, then Breakpoint
Type	Type on-chip data during Breakpoint
Unassemble	Disassembler for shared memory

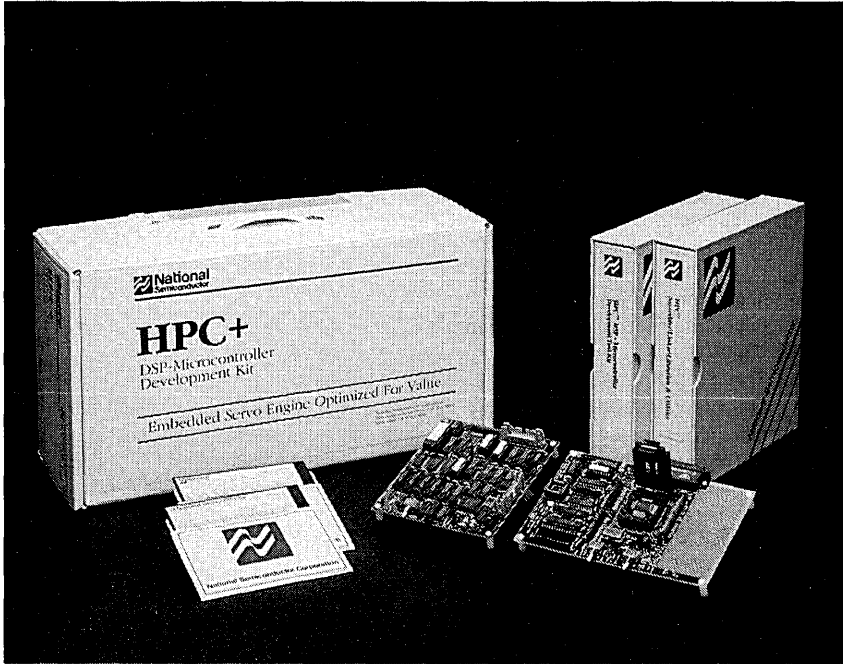
PHYSICAL SIZE

6¼" x 4½"

POWER REQUIREMENTS

+5V @ 1.0A

HPC Plus (HPC46100) Development Kit



TL/DD/11211-4

General Description

The HPC Plus Development kit is a low cost development tool package for HPC46100. The kit contains HPC software and hardware which enables the user to assemble, link and download the user programs and debug the code running on the application target with the help of the debug software hosted on IBM PC/AT compatible machines running DOS operating system.

The Development kit is a combination of PROM Emulator, Logic Analyzer Interface, with a PROM based Monitor and a Host software (Serial Hook) which communicates to the Development kit through the serial COM port on the PC. It also has an evaluation board (Design and Test Board) for the users who don't have their target board designed yet, but want to verify the program.

The Serial Hook and the Monitor Firmware together give the ability to debug the application run by the target HPC at full speed. The Logic Analyzer Interface helps the user to connect HP1650/16500 analyzer easily and trace the code execution. The kit also contains configuration and inverse assembly software for the HP analyzer with which one can analyze the code in the disassembled form.

The kit comes with all the necessary hardware components and user needs the IBM PC/AT running DOS 3.0 or above and a +5V DC 1.0A power supply.

The software package provided along with the kit contains the HPC Assembler/Linker/Librarian, Serial Hook Host interface software and the HP Logic Analyzer configuration/inverse assembly floppy.

Features

- Supports HPC46100 microcontroller
- Single supply +5V operation
- Compact Monitor Firmware executed by Target HPC
- Target Access through EPROM Socket
- 64k bytes of emulation memory
- Monitor reserved address space mappable
- Program control through Break Point, Single Step
- Memory and Register examine and Alter commands
- Real Time emulation
- Honor Interrupts while break pointed
- Unrestricted Assembler package

Accessories from Emulation Technology

HP PREPROCESSOR FOR HPC FAMILY

The preprocessor provided by Emulation Technology simplifies inter-connection between the HP-1650A/B, 16510A/B series of logic analyzers and the HPC family of microcontrollers when used in the ROMless mode. It will provide all clocking and status lines to capture and decode the operation of the HPC Microcontroller.

The preprocessor is available in 68-pin PLCC and 80-pin PQFP inserts. The inverse assembler software included in the package, configure logic analyzer and translate data into HPC specific mnemonics. The user can compare the resulting display to the original assembly language code to help debug the software.

HPC-Microcontroller Development System

HOST SYSTEM REQUIREMENTS

IBM PC/AT Hosts

Without Source/Symbolic Debugger

IBM PC/AT or 100% compatible with 512k bytes of available user memory PC-DOS or MS-DOS 3.0 or above

5.25" floppy drive capable of reading double sided high density format diskette. One RS-232 synchronous serial port for Communication.

With Source/Symbolic Debugger with Graphical User Interface

2 Mbytes of available user memory

MS-Windows 3.0

A Serial or Bus mouse

Sun Workstation Hosts

Sun SPARCstation 1 or 2

SunOS™ version 4.0.3 or later

Development Tools Selection Table

Product	Order Number	Description	Included	Manual Number
HPC16003/ 16083	HPC-DEV-ISE1	HPC In-System Emulator	HPC MDS User's Manual	420420184-001
	HPC-DEV-ISE1-E	HPC In-System Emulator for Europe and Southeast Asia	MDS Comm User's Manual	424420188-001
			HPC Emulator Programmer User's Manual	420421313-001
			HPC16083/16004/16064 Manual	424410897-001
HPC-DEV-IBMA	Assembler/Linker/Library Package for IBM PC/AT	HPC Assembler/Linker Librarian User's Manual	424410836-001	
HPC-DEV-IBMC	C Compiler/Assembler/Linker/Library Package for IBM PC/AT	HPC C Compiler User's Manual	424410883-001	
		HPC Assembler/Linker/Library User's Manual	424410836-001	
HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC/AT C Compiler/Assembler/Linker Library Package for IBM PC/AT	Source/Symbolic Debugger User's Manual	424420189-001	
		HPC C Compiler User's Manual	424410883-001	
		HPC Assembler/Linker Library User's Manual	424410836-001	
HPC-DEV-SUNC	C-Compiler/Assembler/Linker Library Package for Sun SPARCstation	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual		
HPC-DEV-SUNDB	Source/Symbolic Debugger for Sun SPARCstation C Compiler/Assembler/Linker Library Package	Source/Symbolic Debugger User's Manual		
		HPC C Compiler User's Manual		
		HPC Assembler/Linker Library User's Manual		

COMPLETE SYSTEM

HPC16003/ 16083	HPC-DEV-SYS1	HPC In-System Emulator with C Compiler/Assembler/Linker/Library and Source Symbolic Debugger	
	HPC-DEV-SYS1-E	Same for Europe and Southeast Asia	

Development Tools Selection Table (Continued)

Product	Order Number	Description	Included	Manual Number
HPC16004/ 16064	HPC-DEV-ISE3	HPC In-System Emulator	HPC MDS User's Manual	420420184-001
	HPC-DEV-ISE3-E	HPC In-System Emulator for Europe and South East Asia	MDS Comm User's Manual	424420188-001
			HPC Emulator Programmer User's Manual	420421313-001
			HPC16083/16004/16064 Manual	424410897-001
HPC-DEV-IBMA	Assembler/Linker/Library Package for IBM PC/AT	HPC Assembler/Linker Librarian User's Manual	424410836-001	
HPC-DEV-IBMC	C Compiler/Assembler/Linker/Library Package for IBM PC/AT	HPC C Compiler User's Manual	424410883-001	
		HPC Assembler/Linker/Library User's Manual	424410836-001	
HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC/AT C Compiler/Assembler/Linker Library Package for IBM PC/AT	Source/Symbolic Debugger User's Manual	424420189-001	
		HPC C Compiler User's Manual	424410883-001	
		HPC Assembler/Linker/Library User's Manual	424410836-001	
HPC-DEV-SUNC	C-Compiler/Assembler/Linker Library Package for Sun SPARCstation	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual		
HPC-DEV-SUNDB	Source/Symbolic Debugger for Sun SPARCstation C Compiler/Assembler/Linker Library Package	Source/Symbolic Debugger User's Manual		
		HPC C Compiler User's Manual		
		HPC Assembler/Linker Library User's Manual		
COMPLETE SYSTEM				
HPC16004/ 16064	HPC-DEV-SYS3	HPC In-System Emulator with C Compiler/Assembler/Linker/Library and Source Symbolic Debugger		
	HPC-DEV-SYS3-E	Same for Europe and South East Asia		

Development Tools Selection Table (Continued)

Product	Order Number	Description	Included	Manual Number
HPC16400E	HPC-DEV-ISE2	HPC In-System Emulator	HPC MDS User's Manual	420420184-001
	HPC-DEV-ISE2-E	HPC In-System Emulator for Europe and South East Asia	MDS Comm User's Manual	424420188-001
			HPC Emulator Programmer User's Manual	420421313-001
			HPC16400E User's Manual	420420213-001
HPC-DEV-IBMA	Assembler/Linker/Library Package for IBM PC/AT	HPC Assembler/Linker Librarian User's Manual	424410836-001	
HPC-DEV-IBMC	C Compiler/Assembler/Linker/Library Package for IBM PC/AT	HPC C Compiler User's Manual	424410883-001	
		HPC Assembler/Linker/Library User's Manual	424410836-001	
HPC-DEV-WDBC	Source Symbolic Debugger for IBM PC/AT	Source/Symbolic Debugger User's Manual	424420189-001	
		HPC C Compiler User's Manual	424410883-001	
		HPC Assembler/Linker Library User's Manual	424410836-001	
HPC-DEV-SUNC	C-Compiler/Assembler/Linker Library Package for Sun SPARCstation	HPC C Compiler User's Manual HPC Assembler/Linker/Library User's Manual		
HPC-DEV-SUNDB	Source/Symbolic Debugger for Sun SPARCstation	Source/Symbolic Debugger User's Manual		
		HPC C Compiler User's Manual		
		HPC Assembler/Linker Library User's Manual		

COMPLETE SYSTEM

HPC16400E	HPC-DEV-SYS2	HPC In-System Emulator with C Compiler/Assembler/Linker/Library and Source Symbolic Debugger		
	HPC-DEV-SYS2-E	Same for Europe and South East Asia		

EMULATOR DEVICES

NSID	Description	Emulates
HPC467064EL20	Single Chip EPROM Microcontroller with UV Erasability	HPC16083/ 16064/16164

PROGRAMMING SUPPORT

NSID	Description	Emulates
PRGM-7064-LDCC	Programming Adapter for HPC167064 Works with HPC-MDS	HPC Emulator Programming Board User's Manual

Data I/O provides HPC programming support on its Unisite Programmer. For information please contact the nearest Data I/O sales location.

ACCESSORIES

Debugging accessories are available from Emulation Technology. Please contact Emulation Technology for detailed description.

HPC Designer Kit

How to Order:

The self contained designer kit for HPC provides in-system emulation for 20 MHz HPC and comes with full Assembler/Linker/Library package and evaluation C-Compiler. Just add IBM-PC or compatible with 256k byte memory running DOS 2.0 or above, a +5V DC 1.0A power supply and RS-232 cables.

Product	Order Number	Description	Included	Manual Number
HPC	MOLE-HPC-DEVLO	HPC Designer's Kit	HPC DB1 Board Evaluation Compiler Full Assembler/ Linker DB1 User's Manual C Compiler Manual Assembler/Linker Manual	420410901-001 424410883-001 424410836-001

HPC + Development Kit

How to Order:

HPC46100 DSP-Microcontroller is supported by a development kit consisting of:

Logic Analyzer Interface Board

Design and Test Target Board

Inverse Assembler for HP1650/HP16500A Logic Analyzer

HPC Assembler/Linker/Library Package
 Debug Monitor "Serial Hook"

Add an IBM PC/AT with 512 kbyte memory running DOS 3.0 or above and ±5V DC power supply to make it operational. For Logic Analyzer Interface select Hewlett-Packard 1650/16500A/B logic analyzer.

Product	NSID	Description	Included	Manual Number
HPC16100	HPC1-DEV-KIT	ROM Emulator Logic Analysis	Logic Analyzer Interface User's Manual Inverse Assembler User's Manual HPC Assembler/Linker Library User's Manual Target Board User's Manual Serial Hook User's Manual RS-232 Cable	

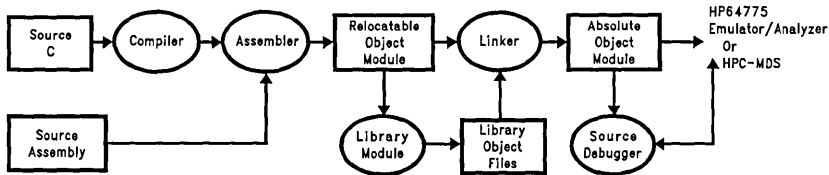
Third Party Vendors

How to order HPC microcontroller third party support tools and accessories:

- For HP64775 series ISE contact:
 Hewlett-Packard North American Sales office
 (800) 447-3282
- For HPC preprocessor and accessories contact:
 Emulation Technology, Inc.
 2344 Walsh Ave.
 Bldg. F
 Santa Clara, CA 95051
 Fax: (408) 982-0664
 Tel: (408) 962-6660

- For programming support of HPC Emulator devices contact:
 Data IO Corporation
 10525 Willows Road
 P.O. Box 97046
 Redmond, WA
 98073-9746
 Tel: (206) 881-6444
 (800) 247-5700

HPC™ Software Support Package



TL/DD/9727-1

■ **Choice of host systems**

- IBM® AT PC-DOS, MS-DOS
- SunOS™ SPARC Station

■ **CCHPC C Compiler**

- ANSI Standard C
- Additional storage class modifiers supported
- Additional statement types included
- Supports embedded assembly code
- Supports multiple source files

■ **ASM HPC Assembler**

- Macro and conditional assembly
- Instruction size optimization
- Symbol table and cross reference output
- Object files are linkable and relocatable

■ **LIBHPC Librarian**

- Supports user developed library modules

■ **LNHPC Linker**

- Links multiple relocatable object modules
- Selects required modules from library files

■ **DBHPC Source debugger**

- C Source, Assembly Symbolic debugger
- Microsoft Window 3.0 user interface for IBM PC/AT hosts
- Line debugger under Sunview for SunOS SPARC Station hosts

General Description

Software Development tools available from National consist of ANSI Compatible C Compiler with hardware specific extension to produce HPC optimized Code. HPC Assembler package has been designed to produce instruction size optimized relocatable object code for speed critical applications. HPC cross linker is used to link object modules produced by assembler or selected from the library file.

The most critical phase of development is when the designer has to integrate and debug hardware and software. National provides a C source and Assembly symbolic debugger. The user interface is MS-windows for PC/ATs and a line debugger under Sunview for Sun SparcStations. This graphical interface is user friendly and provides graphical menu driven debug environment. Source/Symbolic debugger fully supports all features of HPC-MDS and HP64775 emulator/analyzer for real time hardware emulation, breakpoints and trace. Multiple windows can be user defined and will display source file listing, symbols and debugger status. Top figure shows the interaction of software

packages and In-System Emulator. NSC and HP have worked together to ensure that all modes of HPC are properly emulated and development software works properly with HP Emulator/Analyzer.

All HPC development software is hosted on IBM PC/AT, operating in MS-DOS or PC-DOS environment and under UNIX environment supporting SUN operating system using SPARC Station as the host.

The assembler produces relocatable object modules from the HPC macro assembly language instructions. The object modules are then linked and located to absolute memory locations. The absolute object module may be downloaded to the HPC-MDS development system or HP64775 Emulator for debugging.

The C compiler generates assembly source. The C compiler may optionally pass symbolic information through the assembler and linker to the absolute object module. The source debugger then uses this information for C and Assembly language debugging on the host in conjunction with MDS or HP64775.

HPC C Compiler—CCHPC Introduction

The HPC C Compiler (CCHPC) is a full and complete implementation of ANSI Standard C for freestanding environment. Certain additions are included to take advantage of special features of the HPC (for the specific needs of microcontrollers). The enhancements include the support of two non-standard statement types (loop and switch), non-standard storage class modifiers and the ability to include assembly code inline. The compiler supports enumerated types of structures by value, functions returning structures, function prototyping and argument checking.

Symbol Names, both internal and external, are 32 characters. Numerics are 16-bit for **short** or **int**, 32-bit for **long**, and 8-bit for **char**, all as either **signed** or **unsigned**; floating point is offered as **float** or **double**, both using IEEE format.

All data types, storage classes and modifiers are supported. Additional storage class modifiers are provided:

BASEPAGE place **static** variable in faster and more efficient on-chip basepage memory.

NOLOCAL declare function without local variables, thus no stack frame.

INTERRUPTn declare function to execute in response to specific interrupt(s).

ACTIVE declare function to be accessed via faster and more efficient function call mechanism.

All statement types are supported, and two additions are provided:

loop (count) simpler, more efficient **for** looping command.

switchf (value) faster form of **switch** command without constraint checking.

CCHPC SPECIFICATIONS

Note: Enhancements are boldface.

Name length	32 letters, 2 cases
Numbers	
Integer, Signed and Unsigned	16–32 bits
Short and Long	16 bits and 32 bits
Floating, Single and Double	32 bits and 32 bits

Preprocessor

```
#include
#define #define() #undef
#if #ifndef #ifndef #if defined #else #elif #endif
```

Declarations

```
auto register const volatile BASEPAGE
static static global static function NOLOCAL INTERRUPTn ACTIVE
extern extern global extern function
char short int long signed unsigned float double void
struct union bit field enum
pointer to array of function returning
type cast typedef initialization
```

Statements

```
; { ... } expression; assignment; structure assignments;
while () ...; do ... while (); for(;;) ...; loop () ...;
if () ... else ...; switch () ...; case: ...; default: ...; switchf () ...;
return; break; continue; goto ...; ...;
```

Operators

```
primary: function() array[] struct_union . struct_pointer ->
unary: * & + - ! ~ ++ -- sizeof (typecast)
arithmetic: * / % + - << >>
relational: < > <= >= == !=
boolean: & ^ | && ||
assignment: = += -= *= /= %= >>= <<= &= ^= |=
misc.: ? ,
```

Functions

```
arguments: Numbers, Pointers, Structures
return values: Numbers, Pointers, Structures
forward reference (argument checking)
```

Library Definition **Limited-Freestanding environment**

Embedded Assembly Code

HPC C Compiler—CCHPC Introduction (Continued)

All operators are supported, and anachronisms have been eliminated (as per the standard). Structure assignment, structure arguments, and structure functions are also supported. Forward reference functions and argument type checking is supported.

Assembly code may be embedded within C programs between special delimiters.

COMPILER COMMAND FEATURES

The CCHPC runs under different host operating systems. Depending on the host system and the CCHPC command line options, ordering of the elements and their syntax may vary. In all cases, the command line consists of the command name, options or switches, and the filename to be compiled.

The compiler output, in the form of ASMHPC assembler source statements, is put in a file with the extension ".asm".

The following is a description of the CCHPC options or switches:

Include C code in assembler code output—Assembler output file contains the C source code lines as comments.

Invoke C preprocessor before compilation—Allows the C preprocessor invocation to be skipped.

Invoke an alternative C preprocessor before compilation—Allows an alternative preprocessor to be used.

Setting the stack size—This switch takes a numeric argument in the form of a C constant. If the module being compiled contains the function main, the compiler uses the number as the size of the program's execution stack, in words. The option is ignored if the module does not contain main.

Creating 8-bit wide code—This switch creates code that can be executed from 8-bit wide memory by avoiding the use of instructions that fetch 16-bit operands (such as JIDW). This option DOES NOT allow the use of 16-bit values or data in 8-bit memory.

Placing string literals in ROM—The ANSI draft language standard calls for string literals, and individual copies for each usage of the literal to be stored in RAM. This switch allows CCHPC to override this requirement for efficiency, saving startup time, RAM and ROM space. Turn off compiler warning messages.

Indicating directories for include files—This switch takes a string argument which is passed to the C preprocessor. The C preprocessor uses it as a directory to search for include files.

Defining symbol names—This switch passes the string argument to the C preprocessor. It instructs the preprocessor to perform the same function as the #define, allowing the symbol definitions to be moved to the invocation line.

Undefining symbol names—Similarly, this switch passes a string argument to the C preprocessor. It removes any previous definitions.

Permit old-fashioned constructs—Certain anachronisms from Kernighan and Ritchie C that are not permitted in ANSI C will be accepted by the compiler if this option is specified. This option is a convenience for users porting a C program to CCHPC from a Kernighan and Ritchie compiler.

Set chip revision level—This switch is used to generate code to work around bugs in specified chip revisions.

Generate symbolic debug information—This option causes the compiler to create symbolic debug information which is passed to the output assembly file.

BASIC DEFINITIONS

Names may be arbitrarily long, but only the first 32 characters are significant. Case distinctions are respected.

Constants may be of type decimal, octal, hex, character and string.

Escape sequences for new line, horizontal and vertical tab, backspace, carriage return, form feed, alert, backslash, single quote, double quote, octal and hexadecimal numbers are supported.

Comments imbedded in the source code begin with "/*" and end with "*/". Comments can not be nested.

CCHPC supports the following Data types:

Name	Size in Bits
char	8
short	16
int	16
enum	8 or 16
long	32
signed char	8
signed short	16
signed int	16
signed long	32
unsigned char	8
unsigned short	16
unsigned int	16
unsigned long	32
float	32
double	32
long double	32
struct	sum of component sizes
union	maximum of component sizes

The type "char" is treated as signed. Unsigned operations are treated the same as signed operation, except for multiplication, division, remainder, right shifts and comparisons. For signed integers, the compiler uses an arithmetic right shift. For unsigned integers, a logical shift is used when shifting right.

HPC C Compiler—CCHPC Introduction (Continued)

Keywords `const` and `volatile` can be applied to any data. `const` indicates that the symbol refers to a location which is read-only. If the symbol is in static or global storage, it will be assigned to ROM memory. `volatile` indicates that optimization must not change or reduce the accesses to the symbol.

Since the HPC supports 8-bit operations, CCHPC does not automatically promote "char" types to "int" when evaluating expressions. For a binary operation, the compiler promotes a "char" to an "int" only if the other operand is a 16-bit (or more) value or if the result of the operation is required to be a 16-bit (or more) value. The use of 8-bit operations yields efficient code without compromising the correctness of the result.

CCHPC uses the standard C preprocessor and any standard preprocessor functions, including "#define", "#include" and macros with arguments are supported.

A program is set of intermixed variable and function definitions. Variables must always be defined before use, functions may be defined in any order.

Variable initialization is performed according to the draft ANSI standard rules.

Standard C operators, and their hierarchy are as described in the ANSI standard draft.

CCHPC allows the programmer to imbed assembler code directly in the C source. All data between "/"\$ and "\$/" is copied directly to the assembler output file generated by CCHPC.

CCHPC IMPLEMENTATION DEPENDENT CONSIDERATIONS

Memory

CCHPC is designed to execute in a 16-bit environment. Special care must be taken when using CCHPC in an 8-bit HPC system.

Storage Classes

CCHPC supports the following storage classes:

- auto
- static
- register
- typedef
- extern

Due to HPC architectural features, the "register" storage class is limited. A variable can be assigned a "register" only if it is of type pointer and only if a register is available. The first "register" pointer variable encountered is assigned to the HPC B register, the second to the HPC X register and any subsequent ones are treated as "auto" (unless `NOLOCAL` is in effect, in which case it will be treated as "static").

The default storage class for global declarations is "static". The default storage class for declarations within functions is "auto".

Storage Class Modifiers

To make maximum efficient use of HPC architectural features CCHPC supports the notion of "storage class modifiers". A storage class modifier may appear with or in place of a storage class. Following is the set of storage class modifiers:

Keyword	Applicable to
BASEPAGE	variable
ACTIVE	function
NOLOCAL	function
INTERRUPT _n (where n = 1 to 7)	function

Storage class modifiers may be supplied with each variable or function declaration. The effect of each storage class modifier is described in the following:

BASEPAGE—The variable will be allocated in the BASE section. Accessing a basepage variable is more efficient than accessing any other type of variable but the amount of basepage storage is limited.

ACTIVE—The address of the function is placed in the 16 word JSRP table. Calls to the function will require 1 byte of code. The most frequently called functions should be considered for designation as ACTIVE functions for maximum code efficiency.

NOLOCAL—The functions local variables are not allocated on the run-time stack. Instead, they are allocated in static storage. Access to local variables in a NOLOCAL function will be more efficient since access can be direct rather than indexed from the frame pointer. If a function has no arguments or local variables, then entry and exit from the function will be much more efficient since there will be no need to adjust the frame pointer on entry and exit of the function.

INTERRUPT_n—These modifiers can be used to set interrupt vectors (one through seven) to point to a particular function. Any function which has an INTERRUPT storage class modifier has special entry and exit code generated. This code will push all HPC registers (A, B, K, X, PSW and word at RAM address 0) onto the stack before executing normal function entry code. Exit code restores all registers before returning from the interrupt.

C Stack Formation

The Stack Pointer (SP) is initialized to the start address assigned by the linker. The Stack Pointer always points to the next free location at the top of the stack.

Within a function, the compiler maintains a Frame Pointer which is used to access function arguments and local automatic variables. The Frame Pointer location is reserved by the compiler at location Oxbe.

HPC C Compiler—CCHPC Introduction (Continued)

To call a function, the compiler pushes arguments onto the stack in reverse order, performs a jump subroutine to the function. Since all stack pushes are 16-bits, any 8-bit arguments are automatically promoted to 16-bits. On function entry, the compiler creates new stack and frame pointers for the function. On exit, the stack and frame pointers are restored to the values they had on entry to the function.

Using In-Line Assembler Code

CCHPC allows in-line assembler code to be entered in the body of a C function. The assembler code can access any of the currently active variables or can get the address of a variable.

Efficiency Considerations

HPC code size and execution time can be optimized by making maximum use of BASEPAGE variables. When BASEPAGE is full, static variables are next most efficient. The least efficient variables are automatic since they require an indirect indexed access. Minimizing the use of longs and floats will improve efficiency. The HPC architecture strongly supports unsigned arithmetic, so the programmer should use unsigned variables except for cases that absolutely require signed arithmetic. The compiler does not attempt to identify common subexpressions for computation only once, so this must be done by the programmer.

Statements and Implementation

The following C statements are supported by CCHPC:

```
expression;
if
if ... else
while ...
do ... while
for ...
break
goto
continue
return
return ...
case ...
default
switch ...
switchf ...
loop ...
```

The switch statement will generate an efficient jump table for a set of cases if the cases are sufficiently close, or it will generate individual tests for each case. The switchf statement is the same as the switch statement except that when a jump table is generated for the switchf statement the compiler does not generate the code necessary to check the bounds of the value to be switched on. This creates a more efficient form of the switch statement but the programmer must insure that the value being switched on is in range.

The loop statement is an extension to the ANSI standard. Loop allows the programmer to create a code efficient loop by using the HPC DECSZ instruction. The loop statement may be nested. A break statement inside the loop will cause an immediate exit from the loop.

Run-Time Notes

During evaluation of complex expressions, the compiler uses the stack to store intermediate results.

All HPC C programs start with a call to the function "main" with no arguments. Before calling "main", run-time start-up code initializes RAM. The initial values of static or global variables with initialization are stored in ROM and copied to the appropriate variables in RAM. Static or global variables without initialization are cleared to zero. The function "main" must be defined. When "main" returns to the run-time start-up routine it executes the HALT macro provided which puts the chip in an infinite loop.

Since the run-time stack is of fixed size and there is no check for stack overflow, it is up to the programmer to insure that the stack area is large enough to prevent stack overflow.

Memory location zero is reserved by the compiler.

The HPC C Compiler User's Manual provides additional information on the features and functions of CCHPC.

HPC Assembler—ASMHPC

INTRODUCTION

The HPC assembler (ASMHPC) is a cross-assembler for the NSC HPC family of microcontrollers. ASMHPC translates symbolic input files into object modules and generates an output listing of the source statements, machine code, memory locations, error messages, and other information useful in debugging and verifying programs.

ASMHPC has the following useful features—

- Macro capability that allows common code sequences to be coded once.
- Conditional code assembly is supported.
- Translates symbolic assembly code modules into object code. Object modules are linkable and relocatable.
- Symbolic names may be defined for any HPC register, memory location or I/O port. Symbols may be defined as byte or word size.
- Symbol table and cross-reference output is provided.
- Full set of Assembler directives are provided for ease of generating vector tables for interrupts, short subroutine calls, jump indirects and other data generation within the object program.
- Data and code sections are user definable. Sections may be relocatable or absolute. Sections

HPC Assembler—ASMHPC (Continued)

may be assigned to 8-bit memory to support the HPC 8-bit mode. Data sections may be assigned to basepage RAM on the HPC to maximize efficient access to variables.

- Accepts assembly source code generated by the HPC C Compiler, CCHPC.
- Full set of Assembler controls for greater flexibility in debugging modules and programs created by ASMHPC.

ASSEMBLY LANGUAGE ELEMENTS**Assembly Language Statement**

Assembly language statements are comprised of four fields of information.

Label field—This is an optional field. It may contain a symbol used to identify a statement referenced by other statements. A symbol used in this manner is called a label.

Operation field—This field contains an identifier which indicates what type of statement is on the line. The identifier may be an instruction mnemonic or an assembler directive. The operation field is required on all assembler statement lines, except those lines which consist of only a label and/or comment.

Operand field—The operand field contains entries that identify data to be acted upon by the operation defined in the operation field. Operand examples are source or target addresses for data movement, immediate data for register initialization, etc.

Comment field—Comments are optional descriptive notes that are included in the program and listings for programmer reference and program documentation. Comments have no effect on the assembled object module file.

Character Set

Each assembly language statement is written using the following characters:

Letters—A through Z (a through z)

Numbers—0 through 9

Special Characters—!\$%'()* +, - . / : ; < = & # ? _ b^a

Note: Upper and lower case are distinct; b^a indicates a blank.

Location Counter

There is a separate location counter for each program section, and the counter is relative to the start of that section. The assembler uses the location counter in determining where the current statement goes in the current program section. If the program section is relocatable, the linker does the final job of assigning an absolute address to the instruction.

Symbols and Labels

Symbols and labels are used to provide a convenient name for values and statements. Symbols and labels have the same rules for construction, only their use distinguishes a symbol from a label.

Rules for symbol or label construction are:

1. The first character must be either a letter, a question mark (?), an underscore (_), a dollar sign (\$) or a period (.).
2. All other characters may be any alphanumeric character, dollar sign (\$), question mark (?) or underscore (_).
3. The maximum number of characters in a symbol or label may be selected by the user with the SIZE-SYMBOL control. The default is 64.
4. Symbols starting with dollar sign (\$) are local symbols and are defined only within a local region.
5. Labels and symbols are case sensitive.

Operand Expression Evaluation

The expression evaluator in the assembler evaluates an expression in the operand field of a source program. The expressions are composed of combinations of terms and operators. An expression may consist of a single term or may consist of two or more terms combined using operators. Terms are—numbers in decimal, hexadecimal, octal or binary, string constants, labels and symbols or the location counter symbol. Each term has four attributes: its' value, relocation type, memory type and size. The relocation type is either absolute or relocatable. The memory type indicates whether the term represents a BASE, RAM8, ROM8, RAM16, ROM16 or null (in the case of an absolute term). The size of a term is null, byte or word.

The operators allowed in ASMHPC are: arithmetic, logical, relational, upper and lower byte extraction and untype operators. Arithmetic operators are +, -, *, /, MOD, SHL, ROL and ROR. The logical operators are NOT, AND, OR and XOR. The relational operators are EQ, NE, GT, LT, GE and LE. Upper and lower extraction operators are HIGH and LOW. The untype operator is &.

Parentheses are permitted in expressions. Parentheses in expressions override the normal order of evaluation, with the expression(s) within parentheses being evaluated before the outer expressions.

Numbers are represented in ASMHPC in 16-bit 2's complement notation. Signed numbers in this representation have a range of -32768 (x'8000) to +32767 (x'7FFF). Unsigned numbers are in the range of 0 to 65535. String constants are internally represented in the 8-bit ASCII code. All expression evaluation is done treating terms as unsigned numbers, for example, -1 is treated as having the value x'FFFF. The magnitude of the expression must be compatible with the memory storage available for the expression. For example, if the expression is to be stored in an 8-bit memory location, then the value of the evaluated expression must not exceed x'FF.

HPC Assembler—ASMHPC (Continued)**ASSEMBLY PROCESS**

The ASMHPC assembler performs its functions by reading the assembly language statements sequentially from the beginning of a module or a program to the end, generating the object code and a program as it proceeds.

The ASMHPC assembler is a multi-pass assembler which allows it to resolve forward referenced symbols and labels efficiently. The number of passes can be selected using the PASS control. This allows the user to select the level of optimization of forward referenced instructions.

MACROS

Macros help make an assembly language program easier to create, read and maintain. A macro definition is an assembly statement or statements that are referred to by a macro name. The macro may have parameters that are operated upon by the assembly statements. ASMHPC will substitute the macro definition for the macro name with the appropriate parameters during the assembly process. Repetitive or similar code can be defined as macros and the programmer can use the macros to build a library of basic routines. Variables unique to particular applications can be defined in and passed to a particular macro when called by main programs.

Defining a Macro

Macros must be defined before they are used in a program. Macro definitions do not generate code. Code is generated only when the macros are called by the assembly program. Macro definitions have a Macro name by which the macro will be referred in the program, declaration of any parameters to be used in the macro, assembler statements that are contained in the macro body and directives that define the boundaries of the macro.

Following is the macro definition structure:

```
.MACRO mname [,parameters]
```

```
•
•
•
```

```
macro body
```

```
•
•
•
```

```
.ENDM
```

where:

- .MACRO is the assembler directive which initiates the macro definition.
- mname is the name of the macro. Multiple macros can have the same name. The last macro defined is the macro definition used. Macro definitions are retained in the macro definition table; if the current

macro is deleted by the .MDEL directive, the previous definition becomes active. If mname is the same as a valid instruction mnemonic, the macro name is used in place of the normal instruction.

- Parameters are the optional list of parameters used in the macro. Parameters are delimited from mname and additional parameters with commas.
- The macro body is a sequence of assembly language statements and may consist of simple text, text with parameters, and/or macro-time operators.
- .ENDM identifies the end of the macro and must be used to terminate the macro definition.

Calling a Macro

Once a macro has been defined, it may be called by a program to generate code. A macro is called by placing the macro name in the operation field of the assembly language statement, followed by the actual value of the parameters to be used (if any). The form of a macro call is:

```
mname [parameters]
```

where:

- mname is the previously assigned name in the macro definition and
- parameters are the optional list of input parameters. When a macro is defined without parameters, the parameter list is omitted from the call.

The macro call as well as the expanded macro assembly code will appear on the assembler listing if the appropriate controls are enabled.

Using Parameters

The power of a macro can be increased with the use of optional parameters. The parameters allow variable values to be declared when the macro is called.

When parameters are included in a macro call, the following rules apply to the parameter list:

1. One comma and zero or more blanks delimit parameters.
2. A semicolon terminates the parameter list and starts the comment field.
3. Single quotes (') may be included as part of a parameter except as the first character of a parameter.
4. A parameter may be enclosed in single quotes ('), in which case the quotes are removed and the string is used as the parameter. This function allows blanks, commas, or semicolon to be included in the parameter. To include a quote in a quoted parameter, include two quotes (").
5. Missing or null parameters are treated as strings of length zero.

The macro operator @ references the parameter list in macro call. Using the operator @ in an expression, the number of parameters can be used to control conditional macro expansion. The @ operator may also be

HPC Assembler—ASMHPC (Continued)

used with a constant or symbol to reference the individual parameters in the macro parameter list. These capabilities eliminate the need for naming each parameter in the macro definition, which is useful when there are long parameter lists. Using the @ parameter count operator it is possible to create macros which have a variable number of parameters.

The macro operator for concatenation is ^. In a macro expansion the ^ operator is removed and the strings on each side of the operator concatenated after parameter substitution. This operator provides the ability of creating variable labels through the use of macros.

Local Symbols

When a label is defined within a macro, a duplicate definition results with the second and each subsequent call of the macro. This problem can be avoided by using the .MLOC directive to declare labels local to the macro definition.

Conditional Expansion

The conditional assembly directives allow the user to generate different lines of code from the same macro simply by varying the parameter values used in the macro calls.

Nested Macro Calls

Nested macro calls are supported. A macro definition may call another macro. The number of allowable levels of nesting depends on the sizes of the parameter lists, but at least ten is typical.

A logical extension of the nested macro call is the recursive macro call, that is a macro that calls itself. This is allowed, but the programmer must insure that the call does not create an infinite loop.

Nested Macro Definitions

A macro definition may be nested within another macro. Such a macro is not defined until the outer macro is expanded and the nested macro is executed. This allows the creation of special purpose macros based on the outer macro parameters. Using the .MDEL directive and the nested macro capability a macro can be defined only within the range of the macro that uses it.

Macro Comments

All lines within a macro definition are stored with the macro, however, any text following “;” is removed before being stored. This text will appear on the listing of the macro definition but will not appear on the macro expansion.

ASSEMBLY LISTING

The listing generated by ASMHPC contains program assembly language statements, line numbers, page numbers, error messages and a list of the symbols used in the program. The listing of assembly language statements which generate machine code includes the hexadecimal address of memory locations used

for the statement and the contents of these locations. To the left of the instruction, an “R” indicates a relocatable argument in this instruction, “X” indicates an external argument, “C” indicates a complex argument and “+” indicates macro expansion.

The assembler listing optionally includes an alphabetical listing of all symbols used in the program together with their values, absolute or relocatable type, word or byte or null type, section memory type and public or external. Optionally a cross reference of all symbol usage by source line number is given; the defining line number is preceded by a “-”.

The total number of errors and warnings, if any, is printed with the listing. Errors and warnings associated with assembly language statements are flagged with descriptive messages on the appropriate statement lines.

Directives

Directive statements control the assembly process and may generate data in the object program. The directive name may be preceded by one or more labels, and may be followed by a comment. The directive's name occupies the operation field. Some directives require an operand field expression.

Assembler Controls

An assembler control is a command that may be used in the source program on a control line or on the invocation line as an option. A control line is indicated by a # in column 1 of the source line. Comments may be included on a control line by preceding the comment with a semicolon. Invocation line controls are masters and override the same controls in the program source. Examples of assembler control capabilities are: format control of the assembly listing, enable/disable listing of conditional code and conditional directives, listing of comment lines, macro expansion lines, macro object lines only. Cross references and symbol tables can be generated in the listing file, macro local symbols and constants can be put into the symbol table, number of assembler passes specified, assembler controls saved and restored . . .

ASSEMBLER INVOCATION

ASMHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for ASMHPC invocation are: the name of the assembly program(s) or module(s) to be assembled, list of assembler options and the name of a command file that contains additional invocation line source filenames and/or options. An assembler invocation line option is an assembler control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the ASMHPC HELP menu is displayed.

HPC Cross-Linker—LNHPC

INTRODUCTION

The HPC linker (LNHPC) links object files generated by ASMHPC. The result is an absolute load module in various formats, such as NSC ".lm" format, INTEL Hex or COFF formats. LNHPC combines a number of ASMHPC relocatable object modules into a single absolute object module with all the relocatable addresses assigned. All external symbol references between modules are resolved, and library object modules are linked as required.

LNHPC creates two outputs:

1. An absolute object module file that can be downloaded to the MDS development system for emulation and debugging. The output could also be used by the HPC Source Level Debugger for "C" Source or assembly symbolic debugging.
2. A load map that shows the result of the link with an optional cross reference listing.

LNHPC MEMORY ALLOCATION

The Linker places each section in memory based on the attributes of the section and the memory that is available. Available memory is specified by the RANGE command. Each section has the following attributes:

Memory type—BASE, ROM8, ROM16, RAM8, RAM16

Size—determined from the object modules

Absolute—section was specified as absolute in assembler

Fixed—starting address was specified by the SECT command

Ranged—memory range was specified by the SECT command.

Memory is allocated section by section. Sections are allocated in the following order:

1. Each absolute or fixed section is placed in memory at its specified address.
2. Each ranged section is placed in memory within the specified range, regardless of whether this memory has been allocated in the Range Definition. An error will occur if the section can not be located.
3. All remaining sections are allocated as follows: As each section is processed, the ranges for its memory type are examined to find enough free space to allocate the section. Each range is examined in order. The first space large enough to contain the section is used. At this point, the memory allocated is marked used. If not enough memory is available to allocate the section, an error message is displayed. For efficiency, sections which may contain

word aligned data (ROM16, RAM16, BASE which are word aligned) are allocated first. The user will benefit if the word aligned data is placed in these sections and byte data in other sections.

The load map shows the following:

- Range definitions showing the memory ranges specified by the /RANGE option or by the default.
- The Memory Order Map showing the starting and ending addresses of each contiguous range of memory used.
- The Memory Type Map showing how memory is allocated organized by the memory type.
- The Total Memory Map showing the allocation of all ROM and all RAM.
- The Section Table showing each section in the link, along with its starting and ending address. Section attributes are also displayed.

LINKER INVOCATION

LNHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for LNHPC invocation are—the name of the object file(s), module(s) or libraries to be linked, list of linker options and the name of a command file that contains additional invocation line source filenames and/or options. A linker invocation line option is a linker control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the LNHPC HELP menu is displayed.

HPC Librarian—LIBHPC

INTRODUCTION

The HPC librarian (LIBHPC) reads object modules produced by ASMHPC and combines them into one file called a library. The linker can then search the library for any undefined external symbols and link the object module associated with the external symbol. LNHPC will only link in those library object modules required to satisfy external references to maximize efficient use of memory space. LIBHPC is a librarian utility that is provided to allow the user to develop standard modules and place them in libraries. The user may add, delete and list modules in a library file. A library of typical C functions is supplied with the HPC C Compiler (CCHPC). This library is an example of the type of library that could be created for an HPC application program. It is intended to be used as a template for the user to create a custom library specific to the application for maximum code efficiency.

HPC Librarian—LIBHPC (Continued)**LIBRARIAN INVOCATION**

LIBHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for LIBHPC invocation are—the name of the library file to process, list of librarian options and the name of a command file that contains additional invo-

cation line source filenames and/or options. A librarian invocation line option is a librarian control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the LIBHPC HELP menu is displayed.

ISDN Basic Rate Interface Software for the HPC16400E High Performance Data Communications Microcontroller

General Description

The ISDN Basic Rate Interface Software Package implemented on the National Semiconductor HPC™ Microcontroller Family contains the software elements that are necessary to implement CCITT standards Q.921 and Q.931 as approved by ANSI committees for North America.

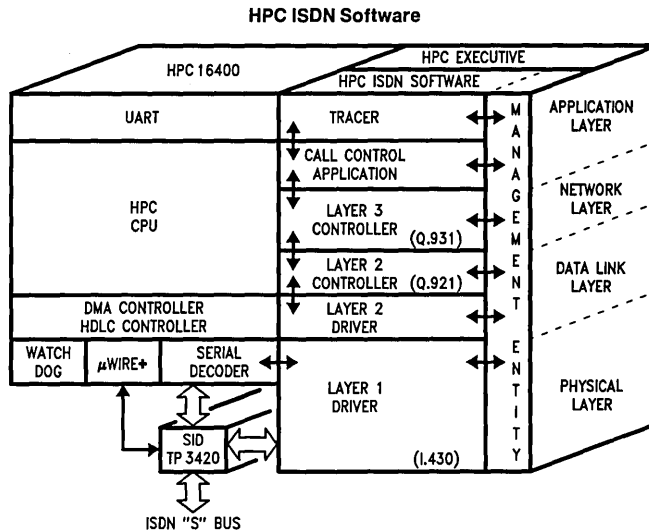
The software package is designed to be easily unbundled and used independently by a software developer. Each layer or function is written as a separate software task. This modular design and well defined task interface make it easy to interface application dependent software to the modules provided. The coding standards for software development have been designed to ensure development of consistent, structured code, which can be easily used and maintained over the life of the code.

This software is supplied as a disk set and is used in conjunction with HPC development tools and software.

Features

- Multi-tasking executive
- Preemptive scheduling
- Modular software design
- Multiple timer facility
- HPC physical layer I/O interface
- Layer 2 link access procedure for the D channel (LAPD)
- Layer 2 link access procedure for the B channel (LAPB)
- Layer 3 protocol control procedure for a terminal endpoint
- Layer management entity support
- Demonstration Call Control Task
- Task_View task exerciser and debugger
- Message trace capability
- Split frame message formatting
- Source code in C language

Block Diagram



TL/DD/10077-1

1.0 Architectural Description

1.1 INTRODUCTION

This description defines the software required to implement the ISDN Basic Rate Interface on the HPC family of micro-controllers, including the HPC16400 which has onboard hardware specifically designed for Data Communication and ISDN applications.

The software consists of the following main parts, shown in overview in *Figure 1.1*:

- HPC Executive, providing an operating environment and services for the ISDN software and for additional application software written by OEM users of the HPC.
- I/O Drivers, interfacing to the DMA/HDLC controllers on the HPC16400 and to the TP3420 "S" Interface Device.
- Data Link Layer Software, implementing the CCITT Q.921 and X.25 link access procedures (LAPD and LAPB).
- Network Layer Software, implementing the Protocol Control Procedures defined in the CCITT Q.931 standard.
- Demonstration Call Control Module, allowing a development engineer at a terminal to make and receive ISDN phone calls which exercise the above software.
- Tracer Module, allowing a development engineer at a terminal to monitor the operation of the above software.
- Management Entity Module

1.2 SOFTWARE ARCHITECTURAL PRINCIPLES

1.2.1 Modular Multitasking Environment

Each layer or function is written as a separate software task. Intertask communications and the interface between tasks

and I/O drivers is by means of mail messages and semaphores, which are managed by the multi-tasking HPC Executive.

This modular design and well defined intertask interface make it easy for users to interface application-dependent software to the modules provided. The services of the HPC Executive (mail, semaphores, timers, memory management) facilitate the writing of software tasks and I/O drivers. These services are available to all tasks and to interrupt-level drivers.

1.2.2 Event-Driven State-Machine Architecture

Telecommunication software typically involves many invocations of the same code (one per call, one per logical connection, etc.) and requires a particular software architecture: tasks must be structured as event-driven state machines. Each task has one or more mailboxes and operates by picking up mail, one message at a time, from the mail queue, and processing the message to completion before returning for the next mail message.

Each "entity" within a task (each call, each logical connection, etc.) has a state block, indicating its current state. After picking up a mail message, the task identifies the entity involved in the message, accesses the state block for that entity, processes the message based on the state of the entity, and finally sets the state block to the new state of the entity.

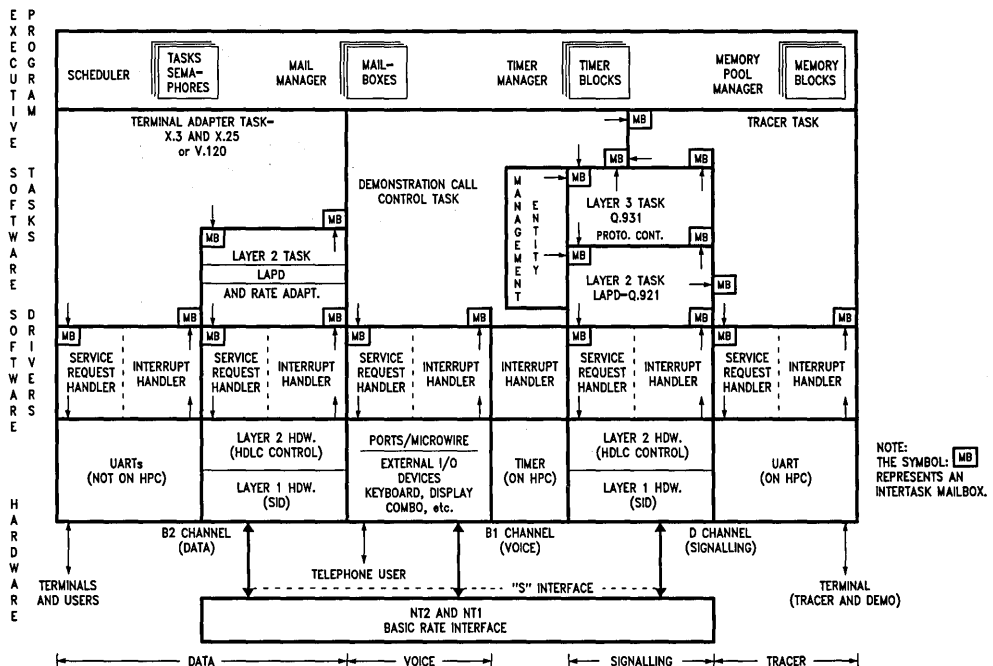


FIGURE 1.1 HPC16400 Software for ISDN

TL/DD/10077-2

1.0 Architectural Description (Continued)

1.2.3 Coding Standards

The coding standards for software development have been designed to ensure development of consistent, structured code, which can be easily used and easily maintained over the life of the code.

1.3 HPC EXECUTIVE

The HPC Executive provides an operating environment for the Layer 2 and Layer 3 tasks, the application tasks, the various support tasks, and the I/O drivers which interface to the hardware. It provides the following services to the tasks and I/O drivers:

- Scheduling of tasks that are ready to run, based on task priority. Preemptive scheduling and time slicing can be optionally enabled.
- Task-task and driver-task communication, by means of mail messages, which can be sent and picked up, and semaphores, which can be signaled and awaited.
- Timers, which are equivalent to mail messages with a specified delay and which allow tasks and drivers to time their activities and time out when an expected event does not occur.
- Memory management, to allocate and deallocate fixed-size buffers as needed by tasks and drivers.

Application tasks and I/O drivers developed by users of the HPC can easily be inserted in the HPC Executive environment and can take full advantage of its services.

1.4 ISDN TELECOMMUNICATIONS STANDARDS

1.4.1 CCITT Standards

The Layer 2 Task implements CCITT 1988 (Blue Book) specification Q.921 (LAPD) and Layer 2 (LA(B) of CCITT specification x.25.

The Layer 3 Task implements the Protocol Control Procedures of CCITT 1988 Specification Q.931.

The Layer 3 Task implements the circuit-switched procedures described in Section 5. The Layer 3 Task implements the Protocol Control procedures and some of the Resource Management. The Call Control Task implements a demonstration version of the Call Control Procedures and the balance of the Resource Management.

In terms of the specification and description language (SDL) diagrams in the Q.931 specification, the Layer 3 Task implements *Figure 38* (26 pages).

The establishment and release of logical links are fully covered in the Layer 2 specifications (Q.921), but the Layer 3 aspects of this are not handled in the version of Q.931 on which the Layer 3 Task is based. Therefore, additional Layer 3 states and SDL diagrams have been created and additional software has been written to handle this requirement.

1.5 ISDN TELECOMMUNICATIONS SOFTWARE

The software packages described below are designed to be easily "unbundled" and used independently by a software developer.

1.5.1 Layer 1 I/O Driver

The Layer 1 I/O Driver controls the HPC MICROWIRE/PLUSTM interface, and the onboard Serial Decoder. This driver is responsible for the hardware initialization, the control of the Serial Decoder, the activation and the deactivation of the Layer 1 I/O device. Use of the HPC Executive mail and semaphore services makes this driver simple to implement and easy to enhance by users that require additional Layer 1 hardware interfaces.

1.5.2 Layer 2 I/O Driver

The Layer 2 I/O Driver controls the HDLC/DMA controllers onboard the HPC16400, and interfaces this hardware to the Layer 2 Task. This driver is responsible for the hardware initialization, the reception of frames toward the HPC, the transmission of frames away from the HPC, and appropriate error handling. Use of the HPC Executive mail and semaphore services makes this driver simple to implement and easy to replace with alternative drivers that a user may wish to develop.

1.5.3 Layer 2 Task

The Layer 2 Task implements the full LAPD protocol defined in Q.921, providing error free in-sequence transmission, reception and multiplexing of the messages received by an HDLC controller connected to the D signaling channel. The event-driven state machine architecture, described above, enables a single software module to support simultaneous activity on multiple logical connections. The Layer 2 Task also supports X.25 LAPB processing for messages received by a second HDLC controller connected to a bearer B channel.

1.5.4 Layer 3 Task

The Layer 3 Task implements the user side of the Protocol Control Procedures of Q.931, which are used to setup, answer, suspend, resume, and disconnect a call. Specifically, it implements all of *Figure 2/Q.931* of Q.931. The event-driven state machine architecture, described above, enables a single software module to support simultaneous activity relating to calls on both bearer B channels.

1.5.5 Demonstration Call Control Task

The latest versions of Q.931 separate the Layer 3 procedures into Protocol Control Procedures and Call Control Procedures. Call Control Procedures are application dependent. These procedures handle bearer channel selection and actual establishment of the voice channel. As Q.931 notes, these procedures can also be considered to be part of the Applications Layer. The Call Control Task implements a minimal subset of the Call Control Procedures, for demonstration purposes. In an actual application, this task will be replaced by an application-specific task, tailored to the capabilities of the actual terminal equipment (number of terminals, handsets, etc.).

1.5.6 Management Entity Task

The Management Entity Task, which is only generically defined in Q.921 and Q.931, handles housekeeping functions

1.0 Architectural Description (Continued)

for all layers. These functions include TEI negotiation with the network management entity, and the handling of unrecoverable errors. This task implements as much of the management entity as is currently defined and in addition whatever is necessary for the operation of the other tasks.

1.5.7 Tracer Task

The Tracer Task serves two purposes; to demonstrate the lower ISDN layers via a menu-driven telephone emulation mode, and to trace system mail message traffic.

1.5.8 Task_View Task Exerciser and Debugger

Task_View is a special-purpose task that can be inserted into the multi-tasking Executive environment in place of the Tracer Task. It reads and interprets a user supplied ASCII scenario file. Under control of this scenario file, Task_View sends mail messages to a specified mailbox (or mailboxes), where they are read by the task under test. Mail messages sent by the task under test in response to this input are then displayed by Task_View. In this way the task may be exercised and debugged.

2.0 Functional Description

2.1 INTRODUCTION

This description defines the functional requirements of the ISDN Basic Rate Interface Software Package implemented on the National Semiconductor HPC Microcontroller Family. Specifically, the HPC16400 Software Package implements or supports the following high-level functions:

- Multi-Tasking Executive
- HPC Physical Layer I/O Interface
- Layer 2 Link Access Procedure for the D Channel (LAPD)
- Layer 2 Link Access Procedure for the B Channel (LAPB)
- Layer 3 Protocol Control Procedure for a Terminal Endpoint

- Management Entity Support
- Call Control Demonstration Task
- Message Trace Capability

The HPC ISDN Software Package has been divided into several functional software elements, as illustrated in the HPC ISDN Functional Block Diagram, *Figure 2.1*. These functional elements correspond to software modules. The purpose of this section is to introduce the various software elements, to define their interactions, and to relate their functionality to the appropriate ISDN standards, where applicable.

The HPC ISDN Software Package will require additional software drivers and application-specific tasks prior to serving as a useful ISDN Terminal Endpoint (TE) entity. The HPC ISDN software has been coded and documented to allow easy integration of additional application code.

The HPC ISDN Software elements illustrated in *Figure 2.1* have been divided into the following categories.

- HPC Executive (2.2)
- I/O Device Drivers (2.3)
- ISDN Layer Protocol Tasks (2.4)
- Application Tasks (2.5)
- System Utilities (2.6)

The HPC Executive contains software elements that are necessary for HPC ISDN Applications. These elements include a Multi-Tasking Scheduler, a Memory Manager, a Timer Manager and a Mail Manager. The HPC Executive software elements are tightly coupled, and streamlined for the National Semiconductor HPC family of controllers.

The I/O Device Drivers interface the HPC hardware elements to the HPC ISDN Software. The Layer 1 Driver implements the ISDN PHYSICAL Layer 1 requirements for the HPC ISDN system. The Layer 2 Driver interfaces the HPC DMA/HDLC controller channels to the Layer 2 Link Access

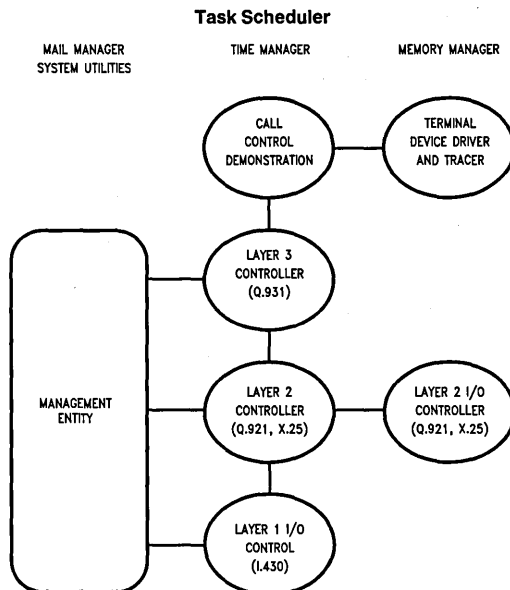


FIGURE 2.1 HPC ISDN Software Functional Block Diagram

TL/DD/10077-3

2.0 Functional Description (Continued)

Procedures. The Terminal Device Driver interfaces the HPC on-board UART to the ISDN Software. Device initialization sequences, service request tasks and accompanying interrupt service routines are all defined in the I/O Device Driver section of this document.

The Layer Protocol Tasks implement the ISDN DATA LINK Layer 2 and the NETWORK Layer 3 requirements for the HPC ISDN system. These tasks are designed to be hardware configuration and application independent. The Layer 2 Task provides both the "USER SIDE" and the "NETWORK SIDE" implementation of the CCITT Specification Q.921. The Layer 3 Task provides the "USER SIDE" implementation of CCITT Specification Q.931.

The Layer 2 Task has been designed to use many of the same routines to implement the link access procedures on either the signaling D channel or the bearer B channel (LAPD or LAPB). Design decisions have also been made to facilitate the implementation of V.120, the rate adaption protocol that processes LAPD frames on a bearer B channel.

The Management Entity Task and the Call Control Task are Application (Specific) Tasks that are closely coupled to the specific system hardware configuration and the Central Office Network Entity Software. These tasks are provided for demonstration purposes to drive the ISDN layer entities. Application users must either replace or extensively rewrite these tasks to match their particular ISDN Application environment.

The System Utilities include the power-up reset Main Task, the NMI handler, the Timer interrupt handler, and the Watchdog Task.

The Tracer utility provides the capability of on-line tracing of intertask mail messages and task states. Tracer is primarily a passive task; it displays messages that it receives from other tasks. Tracer also provides a user interface for Telephone Simulation.

The remainder of this document is devoted to defining each of the software elements at the functional level. Where applicable, specific ISDN standard documents such as CCITT Q.921, Q.931 and X.25 will be referenced, rather than duplicating the information here.

2.2 HPC EXECUTIVE

The HPC Executive provides a multitasking environment within which the ISDN and applications tasks can run and it provides various system services to those tasks. The services of the Executive are available to both tasks and interrupt service routines.

2.2.1 Tasks, Priorities, and the Ready Queue

A task is a subroutine which can be run (called) by the Executive. Tasks are managed by the Executive as Task Control Blocks (TCB's). A task's TCB contains all the parameters needed by the Executive to handle the task, in particular, the task's priority and its current starting address.

Tasks which are not blocked waiting for a semaphore or for mail are considered to be ready to run and their TCB's are queued on the Ready Queue, in the order of the tasks' priorities. The Task Scheduler runs the task at the head of the Ready Queue, i.e., the highest priority task that is ready to run. In this way the processor is always given to the highest priority task that is ready to run.

Once a task is started, it continues to run until it does a Semaphore Wait, ReadMail, or Return or, until a higher priority task is put on the Ready Queue, at which time the scheduler has the opportunity to once again choose the task at the head of prioritized Ready Queue and run that task.

A task may change the priority of any task, including itself. The priority change takes place immediately, to the extent that the target task's TCB is updated with the new priority and the queue in which the target task's TCB is waiting is resorted to reflect the new priority.

If the target task is in the Ready Queue and its new priority is higher than the priority of the running task, then the target task will run once all protected sections are exited. See Section 2.2.3, below.

2.2.2 Semaphores

A semaphore is a global variable, accessed through the Executive, which can be Signaled (incremented) by one task and Waited on by another task. A semaphore is typically used to manage the sharing between tasks of some resource, e.g., an I/O device, mail messages, etc. At any moment the value of a semaphore may be positive, negative, or zero. A positive value indicates the number of resources available, a negative value indicates the number of tasks waiting for resources and a zero value indicates that there are no resources available and no tasks waiting for them.

When a task Waits on a semaphore, if the semaphore has a nonzero positive value, the task will immediately go on the Ready Queue and the semaphore value will be decremented by one. On the other hand, if the semaphore has a zero or negative value, the task will be queued on the semaphore and the semaphore value will be decremented by one. When a task Signals a semaphore, the semaphore's value is incremented by one and the highest priority task waiting on the semaphore is put on the Ready Queue.

A common use for a semaphore is the management of a non-shareable resource, such as an I/O device. When the device is available, the associated semaphore has the value +1. When a task wishes to obtain exclusive use of the device, it Waits on the semaphore, which is then decremented to 0, with the task going immediately back on the Ready Queue. If another task then attempts to use the device, its Wait call will cause it to be placed on the Semaphore Queue and the value of the semaphore will be decremented to -1. Other tasks may also Wait on the semaphore, each decrementing its value by one. The negative value of the semaphore indicates the number of tasks Waiting for the device. The waiting tasks are ordered in the semaphore queue according to their priority. When the first task is done with the device, it Signals the semaphore, which moves the first waiting task to the Ready Queue and increments the semaphore or, if there are no waiting tasks, returns the semaphore to its original value of +1.

2.2.3 Preemptive Scheduling

Preemptive scheduling enables the executive to respond quickly to high priority events. If a task that is waiting on a Semaphore Queue moves to the Ready Queue and if that task is of higher priority than the currently running task, then, as soon as the currently running task emerges from all critical sections and non-preempt sections, the currently running task will stop running. The task that was moved to the Ready Queue will run. The preempted task will be placed on the Ready Queue in the normal manner.

2.0 Functional Description (Continued)

Executive functions allow preemption to be selectively turned on or off by task or for an entire application.

2.2.4 Time Slicing

Time slicing modifies the task scheduling algorithm as follows: at each "tick" of the timer clock (the clock which also controls the time-out timers), if the currently running task has the same priority as the task at the head of the Ready Queue, then, if the currently running task is not in a non-preempt section, it will stop running and the task at the head of the Ready Queue will run. The task that stops running is placed on the Ready Queue in the normal manner, i.e., after all tasks of equal priority. Time slicing enables the Executive to share the processor equally between tasks of equal priority.

2.2.5 Mailboxes and Mail Messages

The main form of intertask communication is the sending and receiving of mail. Mailboxes exist independently of tasks; any task may send mail to any mailbox and any task may read mail from any mailbox. However, in a typical system, each task has one mailbox from which it reads all its mail and to which other tasks send mail destined for that task.

Mail is prioritized. When a task calls upon the Executive to perform a SendMail, it specifies the priority of the message, which is inserted in the specified mailbox queue sorted by priority.

2.2.6 Timers

The Executive includes a timing facility specifically designed to handle the time-outs typical of telecom protocols and other real-time applications.

Timers are essentially a form of delayed mail. When a task sets a timer, the task provides a mailbox identifier, a mail message, and a time delay value. When the specified time delay is up, i.e. when the timer "expires", the mail message is mailed to the specified mailbox. When a task sets a timer, it receives a timer ID, which can be used to cancel the timer, if necessary, before it expires.

2.2.7 Memory Management

The memory manager is responsible for allocating and deallocating fixed-size memory blocks from fixed-size pools, which are completely defined at compile time. A memory pool may reside in extended memory.

2.2.8 System Module and Interface Module

The Multi-Tasking Executive Software is available either as source code or as object code. The interface module, which must be modified to insert application tasks, is always supplied as source code.

2.3 I/O DEVICE DRIVERS

I/O Device Drivers serve as interface routines between the HPC hardware machinery and the HPC Executive and Application Tasks. "Input" operations (data heading toward Application Tasks) are typically fielded by an Interrupt Service Routine (ISR). The ISR may SEND information to the appropriate task via the system mail facility, or it may signal the appropriate semaphore to schedule an I/O task. "Output" operations (data heading away from Application Tasks) are typically fielded by Service Request (SRQ) Tasks. SRQ Tasks communicate directly with the hardware control registers to initiate output operations. These tasks often work

closely with their accompanying ISR for output initiation and completion. Higher layer tasks send mail messages to the SRQ Tasks, using the system mail facility to queue messages pending output.

The HPC ISDN Software includes three I/O Device Drivers: the Layer 1 Driver, the Layer 2 Driver and the Terminal Driver. The functionality of these drivers is defined below. Details of particular Device Driver ISR and SRQ Task interactions are defined in the Software User's Manual.

2.3.1 Layer 1 I/O Device Driver

The Layer 1 I/O Device Driver provides implementation of the ISDN PHYSICAL Layer 1 for the HPC environment. This Device Driver controls the NSC MICROWIRE/PLUS Interface to the NSC TP3420 "S" Interface Device (SID), and the HPC16400 onboard, Serial Decoder. Control of a COMBO™ Codec, a display, and a keypad has been implemented later by either adding to this driver, or using it as a model for additional drivers.

The primary responsibility of this driver is to initialize and control the SID. The higher layer ISDN tasks mail activation and deactivation messages to the Layer 1 Service Request Task. This task sends the appropriate command to the SID via the MICROWIRE/PLUS Interface. The SID interrupts the HPC whenever it changes state. The Layer 1 Interrupt Service Routine fields responses when the SID changes state and mails the information to the Layer 2 Controller Task and to the Management Entity Task.

The Serial Decoder is initialized to MODE 4, with the ISDN D Channel terminated by DMA/HDLC Channel # 1, and Bearer Channel B2 terminated by DMA/HDLC Channel # 2. The SID can swap B1 and B2 internally to allow voice or data on either channel.

The Layer 1 I/O Device Driver can communicate with any other Task via the System Mail Utilities.

2.3.2 Layer 2 I/O Device Driver

The Layer 2 I/O Device Driver interfaces the two HPC16400 onboard DMA/HDLC channels; one to the 16 kbit per second "D" signaling channel, and one to the 64 kbit per second bearer (B2) channel. The Layer 2 Service Request Task receives Physical Layer (PH) Primitives from the Layer 2 Controller Task via the system mail utility. The Layer 2 Interrupt Service Routine handles block messages received from the DMA Controller and mails them as Physical Layer (PH) Data Primitives to the Layer 2 Controller Task. This generic mail message interface allows an Application User to easily introduce external DMA and HDLC Controllers, and accompanying device drivers, that either replace or complement the existing onboard controllers.

HDLC/DMA Channel # 1 is attached to the ISDN signaling D channel, and will be referred to as the LAPD Channel. HDLC/DMA Channel # 2 is attached to bearer channel B2, and will be referred to as the LAPB Channel. The two channels operate independently of each other as much as possible. Since they share the same interrupt hardware, the Layer 2 Interrupt Service Routine must poll the Message Pending Register and the Error Status Register to determine the source of each interrupt. Both HDLC/DMA channels use the HPC field separation feature for transmission and reception of data. This feature relieves some memory concerns, since it allows small memory buffers to be used for mes-

2.0 Functional Description (Continued)

sages that only have headers. In the transmit direction this feature allows large contiguous buffers to be broken up into smaller send buffers without having to copy them following a header. Issues specific to the HDLC/DMA Channels are defined below.

HDLC/DMA Channel #2, the LAPB Channel, requires frame sizes to be nominally 130 bytes, 2 bytes of header and 128 bytes of information. Provision can be made for messages with up to 1026 bytes, 2 bytes header and 1024 bytes of information.

The presentation of data between the Layer 2 Driver and Layer 2 Controller is identical regardless of which channel the frames are associated with.

2.3.3 Terminal Device Driver and Tracer

The Terminal Device Driver interfaces to the HPC onboard UART. The associated SRQ Driver Task, referred to as Tracer, serves primarily as a high-level demonstration vehicle. Tracer can field mail messages from any other task in the system, as well as keystroke mail messages from its accompanying ISR. Tracer's responsibilities include the following functions:

- Management of the Telephone Simulation User Interface,
- Display Management of the System Trace Mail Messages,
- Proper Display of Task-Related Information

The Telephone Simulation function of Tracer allows the user to enter "telephony-like" keystroke characters, that are passed to Tracer, then on to the ISDN layer tasks for processing. Menu responses are fielded by Tracer to select various levels of the Trace function, as well as to enter and exit the Telephone Simulation mode.

Depending on the level of trace that is selected, Tracer receives mail messages from the system tasks and properly formats them on the CRT display. Tracer offers various levels of trace capability. Trace can be turned off all together, in which case only the application layer Telephone Simulation inputs will be displayed. Trace can display all messages from every layer, or it can be set to "zoom" to display only the messages at a particular layer. Messages will generally have address fields and data fields.

The Terminal Driver's Interrupt Service Routine (ISR) handles keyboard characters from the UART and mails them to the Tracer SRQ Task for further processing. The ISR also handles transmission completion of a character that has been sent to the CRT.

The data structures and hardware interface requirements for the Terminal Device Driver, and capabilities of Tracer, are defined in the Software User's Manual.

2.4 ISDN LAYER PROTOCOL TASKS

The ISDN Layer Protocol Tasks provide implementation of the DATA LINK Layer 2 and the NETWORK Layer 3 in accordance with the protocol definitions of the CCITT Specifications. The two Layer Protocol Tasks (the Layer 2 Controller Task and the Layer 3 Controller Task) are designed to satisfy the ISDN Basic Rate Interface (BRI) Terminal Equipment requirements. They are independent of user applications and hardware environment. The PHYSICAL Layer 1 implementation is defined in the I/O Device Driver section of this document. Implementation of layers above the NETWORK Layer 3 are specific to user applications. Two such

layer tasks are provided, the Demonstration Call Control Task and the Management Entity Task. These tasks are defined in the Application Task section of this document.

The purpose of the Layer 2 Controller Task is to provide the NETWORK Layer 3 with an error free, sequenced data frame service. The Layer 2 Controller Task uses CCITT Specifications Q.921 and X.25 and the primary functional specifications. The Layer 2 Controller Task satisfies the Link Access Procedures for both the D Channel and the B Channel (LAPD and LAPB). Design considerations have also been included for the future implementation of V.120, the new CCITT rate adaption scheme.

The Layer 2 Controller Task's data frame delivery service allows the Layer 3 Controller Task to confidently setup and teardown user voice and data calls on the available facilities. The Layer 3 Controller Task uses CCITT Specification Q.931 as the primary functional specification. Note that the X.25 Layer 3 packet processor task is not included in the initial software package.

The Layer Protocol Tasks require a somewhat non-conventional task architecture in order to simultaneously manage a significant number of multiple logical connections. This event-driven state-machine architecture requires that a state memory block be created and maintained for each logical connection. When a Layer Protocol Task "wakes up" due to the arrival of mail, the message's address is interrogated to determine which logical connection is to receive the mail. The particular logical connection's state block is retrieved and the mail message is processed per the CCITT Specification requirements, depending on the state of the particular logical connection. Typically, processing the mail message results in sending a Primitive message to another task, and updating the logical connection's state block. The Layer Protocol Task then returns to its mail box to pick up any subsequent mail.

The interface between all of the ISDN Layer Tasks is deliberately achieved via the System Mail Utilities. This ensures a distinct, uniform layering mechanism in the event that application programmers wish to replace layers with their own implementations.

2.4.1 Layer 2 Controller Task

The primary job of the ISDN Data Link Layer 2 is to deliver error-free, sequenced data frames to the Network Layer 3. The Layer 2 Controller Task implements the following Layer 2 Link Access Procedures (LAP) for the HPC ISDN Software Package:

- LAPB per the X.25 CCITT Specification.
- LAPD per the Q.921 CCITT Specification.
- V.120 Terminal Adaption capability.

Since the Q.921 LAPD requirements were derived from the X.25 LAPB requirements, most of the same Layer 2 Controller Task routines can be used to implement both LAPB and LAPD. Design considerations have been made to allow future implementation of V.120.

The Layer 2 Controller Task communicates with the Layer 2 DMA/HDLC Controller Device Driver Task and the Management Entity Task, via the System Mail Utilities. These tasks interrogate the mail message headers to determine whether to process the frames using LAPB or LAPD procedures. The

2.0 Functional Description (Continued)

LAPD frames are mailed to the Q.931 Layer 3 Controller Task, while the LAPB frames are mailed to the X.25 Layer 3 Task.

The HPC16400 HDLC hardware handles the Layer 2 HDLC Procedures, which includes bit stuffing, address recognition, and Frame Check Sequence generation and detection. The Layer 2 Controller Task is responsible for the Layer 2 "Data Link Procedure", which includes the following functions:

- Data Transmission
- Protocol Exception Management
- LAPD-Specific Functions.

To accomplish these functions the Layer 2 Controller supports the full set of Layer 2 Peer-to-Peer messages defined in the CCITT Specification Q.921. These messages are listed below and defined further in the Software User's Manual.

UI	Unnumbered Information Frames
UA	Unnumbered Acknowledge
SABM(E)	Set Asynchronous Balanced Mode (Extended)
DISC	Disconnect Command
DM	Disconnect Mode
I	Acknowledged Information Frames
RR	Receiver Ready
RNR	Receiver Not Ready
REJ	Request Recrimination of Frames
FRMR	Unrecoverable Error, Frame Reject

The Layer 2 Controller Task also supports the primitives required to communicate with the other ISDN tasks.

2.4.1.1 Layer 2 Data Transmission

Layer 2 peer-to-peer Data Transmission is supported with two modes: Unacknowledged Data Mode and Multi-Frame Acknowledged Data Mode. The Unacknowledged Data Mode is used primarily for setting up logical connections and for peer-to-peer Management Entity communication. This mode uses the Unnumbered Information (UI) and the Unnumbered Acknowledge (UA) messages. The Multi-Frame Acknowledged Mode is established by the Set Asynchronous Balanced Mode (SABM) command. This mode provides the mechanism for acknowledgement of data frame transport in each direction. The Multi-Frame Acknowledged Mode is terminated with the Disconnect (DISC) command. The response to the DISC message can be either an Unnumbered Acknowledge (UA) message or a Disconnect Mode (DM) message. The actual Layer 2 data frames are transmitted in the Information (I) messages, while in the Multi-Framed Acknowledged Mode.

The Layer 2 Controller is responsible for avoiding message congestion and buffer overflow. A Layer 2 entity can issue the Receive Ready (RR) command to its peer to indicate that it is ready to continue data transmission. Likewise, the Layer 2 Controller can issue the Receiver Not Ready (RNR) command to its peer to indicate that it is not ready for data transmission.

2.4.1.2 Layer 2 Protocol Exception Management

The Layer 2 Controller Task is responsible for handling exceptions to the Data Link Protocol. These exceptions are of

two types: recoverable and unrecoverable. Recoverable exceptions in the receive direction are typically failed frames, which are handled by requesting the retransmission of the failed frame with the Reject (REJ) command. Recoverable exceptions in the transmit direction include the expiry of a Layer 2 Timer. Timer expiry requires the retransmission of the frame that was not acknowledged in time, and all subsequent frames. Timer expiry also prompts a message to the Management Entity. Unrecoverable exceptions result in the Frame Reject (FRMR) response. A message to the Management Entity Task is also sent in this case.

2.4.1.3 Layer 2 LAPD-Specific Functions

The following Layer 2 Controller Task functions are LAPD specific. These functions involve establishing and maintaining multiple logical data link connections. Note that a LAPB connection will be maintained as a special independent logical connection.

A two byte address is required for each logical data link. This address is referred to as the Data Link Connection Identifier (DLCI). The DLCI consists of a Service Access Point Identifier (SAPI) and a Terminal Endpoint Identifier (TEI). The Layer 2 Controller Task is responsible for supporting the TEI Assignment Procedure and the TEI Verification Procedure. These procedures are both initiated by the Management Entity. The Layer 2 Controller Task supports both the Automatic and Non-Automatic TEI Assignment Procedures.

Establishment of the LAPD multi-frame acknowledged data transmission mode requires an extended command (SABME) to prompt the peer entity that the frames are intended for a particular logical data connection identified by the accompanying DLCI. The Layer 2 Controller Task maintains each logical link's state and data frames independently, as explained earlier in this section.

The Layer 3 Controller Task addresses and maintains independent logical connections via an identifier called a Connection Endpoint Suffix (CES). Since the CES is different from the Layer 2 Terminal Endpoint Identifier (TEI), a mapping function is required. The Layer 2 Controller Task maintains a CES-TEI translation procedure to properly address Layer 3 logical entities.

2.4.2 Layer 3 Controller Task

The Layer 3 Controller Task implements the application independent portion of the ISDN NETWORK Layer 3 protocol, per the Q.931 CCITT Specification. The primary responsibility of the Layer 3 Controller Task is to establish a network access connection link between a terminal and its peer in the Central Office.

The Layer 3 Controller Task communicates with both the Layer 2 Controller Task and the Call Control Task by sending primitives via the System Mail Utilities. The Layer 3 Controller Task also communicates with the Management Entity Task. The HPC ISDN Layer 3 Controller Task is responsible for the following NETWORK functions:

- Call Establishment and Clearing
- Call Suspension and Resumption
- Call Status and Notification
- Protocol Exception Management.

2.0 Functional Description (Continued)

The Layer 3 Controller Task supports all the Network Layer Peer-to-Peer messages defined in the CCITT Specification Q.931, i.e.:

- Call Establishment and Clearing Messages:

ALERT	Alerting
CALL PROC	Call Proceeding
CONN	Connect
CONN ACK	Connect Acknowledge
INFO	Information
PROG	Progress
SETUP	Setup
SETUP ACK	Setup Acknowledge
DISC	Disconnect
REL	Release
REL COM	Release Complete

- Call Suspension and Resumption Messages

RESUME	Resume
RESUME ACK	Resume Acknowledge
RESUME REJ	Resume Reject
SUSPEND	Suspend
SUSPEND ACK	Suspend Acknowledge
SUSPEND REJ	Suspend Reject

- Miscellaneous Messages

NOTIFY	Notify
STATUS	Status
STATUS EN	Status Enquiry
USER INFO	User Information

2.4.2.1 Call Establishment And Clearing

The Layer 3 Controller Task's primary responsibility is to establish and clear user network connections on available bearer channel facilities. The Q.931 CCITT Specifications include Call Establishment and Clearing of both circuit-switched and packet-switched calls. Initially, the HPC ISDN Software Package only supports circuit-switched call procedures on Basic Rate Interface (BRI) Bearer Channels. The Layer 3 Controller Task is responsible for Call Reference assignment and maintenance. The Layer 3 Controller Task supports Call Establishment using both the Overlap and Non-Overlap (enbloc) addressing modes.

The procedure for establishing and clearing network connections is defined in CCITT Specification Q.931. It is important to note that the Layer 3 Controller Task maintains an associated state block for each network connection. Primitive mail messages arriving at the Layer 3 Controller Task will be interrogated to determine which network connection is to receive the mail. The mail message is processed depending on the state of the network connection. This processing typically includes the transmission of a Primitive to another Layer Task, and the appropriate update of the network connection state block.

2.4.2.2 Call Suspension And Resumption

Call Suspension (SUSPEND) requires that the Bearer Channel facility and the Call Reference for a call be temporarily relinquished. The network connection is left intact, but in the suspend state. The RESUME command reactivates the call by obtaining a Bearer Channel facility and establishing a new Call Reference. The Suspend function is somewhat analogous to the call HOLD feature.

2.4.2.3 Call Status And Notification

The Network can request the status of a network connection at any time via the USER INFO, NOTIFY and STATUS Commands. The information includes Service Validation and Channel Configuration.

2.4.2.4 Layer 3 Protocol Exception Management

The Layer 3 Controller is responsible for handling exceptions to the Network Control Protocol. The primary Layer 3 Controller Task protocol exception is the expiry of the Layer 3 timer. Such an exception requires the retransmission of the particular command and may prompt a message to the Management Entity Task.

2.4.2.5 Timer Support

The Layer 3 Controller supports the following system timers per CCITT Specification Q.931:

T303	SETUP ACK Timer
T305	DISCONNECT ACK Timer
T308	RELEASE ACK Timer
T313	CONNECT ACK Timer

2.4.2.6 SDL Updates

The Layer 3 Controller Task very closely follows the SDL procedures illustrated in CCITT Specification Q.931, with a few enhancements. These enhancements are listed here and fully defined in the Software User's Manual.

- Three new SDL States have been added to accommodate establishing the Data Link corresponding to a particular CES. The new states are:

- IDLESTATE
- RELEASEWAIT
- ESTABLISHWAIT

- The Q.931 NULLSTATE SDL now accepts a new command, CCBROADCASTRESP. This command is sent from the Call Control Task to allow transition from the NULLSTATE(0) to the CALLPRESENT State(6) during a Network Originated call via the Broadcast mechanism.

2.5 APPLICATION TASKS

The Application Tasks are very dependent on both the terminal equipment configuration and the far-end Network Entity software implementation. The HPC ISDN Software Package includes two sample Application Tasks: the Demonstration Call Control Task and the Management Entity Task. Both of these tasks can be replaced or updated when ported to a particular application. These tasks are included in the HPC ISDN Software Package primarily to verify the operation of the OSI Layer Protocol Tasks and the HPC Device Drivers.

2.5.1 Demonstration Call Control Task

The Demonstration Call Control Task is closely coupled to the specific facilities of an application. The interaction between the Layer 3 Controller Task and Call Control is defined in CCITT Specification Q.931. In the HPC ISDN Application, the Call Control Task communicates with the Layer 3 Controller Task and the Tracer Task. The availability of two circuit switched voice bearer channels is simulated in the Call Control Task. The Call Control Task sends standard Terminal Equipment prompts and messages to the Tracer Task where they are displayed on a UART driven CRT. The Call Control Task has the following responsibilities:

- B Channel Resource Management

2.0 Functional Description (Continued)

- Connection Endpoint Suffix (CES) Allocation
- Conversion between L3 Primitives and Terminal Action.

The Call Control Task and the Layer 3 Controller Task communicate via the NL_DATA_REQ and NL_DATA_IND Primitives. The messages that are supported between these tasks are listed below.

- Commands from Call Control to Layer 3

CC_SETUP_REQ	Setup Request
CC_SETUP_RESP	Setup Response
CC_SETUP_REJ_REQ	Setup Reject
CC_INFO_REQ	Information
CC_DISCONNECT_REQ	Disconnect
CC_RELEASE_REQ	Release
CC_ALERTING_REQ	Alerting
CC_BROADCAST_RESP	Broadcast Response
CC_CALLPROC_REQ	Call Proceeding
CC_PROGRESS_REQ	Progress
CC_NOTIFY_REQ	Notify
CC_RESUME_REQ	Resume
CC_RESUME_REJ	Resume Reject
CC_SUSPEND_REQ	Suspend Request
CC_SUSPEND_REJ	Suspend Reject
- Command from Layer 3 to Call Control

CC_SETUP_IND	Setup
CC_SETUP_CONF	Setup Confirm
CC_SETUP_COMP_IND	Setup Complete
CC_INFO_IND	Information Indication
CC_ALERTING_IND	Alerting
CC_PROGRESS_IND	Progress
CC_DISCONNECT_IND	Disconnect
CC_RELEASE_IND	Release
CC_CALLPROC_IND	Call Proceeding
CC_RELEASE_CONF	Release Confirm
CC_STATUS_IND	Status Indication
CC_ERROR_IND	Error Indication
CC_RESUME_CONF	Resume Confirm

The Call Control Task also communicates with the Tracer Task using single byte keystroke like commands. These commands are packaged mail messages containing two bytes: the first byte is the Sender Task's ID, the second byte is the keystroke command. The following messages are sent between Call Control and Tracer:

- Keystroke Commands from Tracer to Call Control Task

TR_ON_HOOK	ON Hook
TR_OFF_HOOK	Off Hook
TR_DIGIT_1	Digit 1
TR_DIGIT_2	Digit 2
TR_DIGIT_3	Digit 3
TR_DIGIT_4	Digit 4
TR_DIGIT_5	Digit 5
TR_DIGIT_6	Digit 6
TR_DIGIT_7	Digit 7
TR_DIGIT_8	Digit 8
TR_DIGIT_9	Digit 9

TR_DIGIT_0	Digit 0
TR_DIGIT_STAR	Digit *
TR_DIGIT_POUND	Digit #

- Commands from Call Control Task to Tracer

TR_IDLE	Idle, ON HOOK
TR_DIALTONE	Dial Tone
TR_DIALING	Dialing
TR_RINGING	Ringing
TR_BUSY	Busy
TR_CONVERSATION	Conversation
TR_RINGBACK	Ringback
TR_ERROR	Error

2.5.2 Management Entity Task

The Management Entity Task is closely coupled to the accompanying Network Management Entity design and to the terminal hardware configuration. Implementation design decisions have been made that make the Management Entity Task unique to a particular application, while still following the general requirements of the CCITT Specifications. Modifications will be required in the Management Entity Task prior to its successful operation in a particular application environment. The Management Entity Task that is included in the HPC ISDN Software Package presumes a particular hardware configuration and Central Office Software implementation.

The Management Entity Task communicates with the Layer 3 Controller Task, the Layer 2 Controller Task, and the Layer 1 Device Driver Task via the System Mail Utilities.

The Management Entity Task has the following responsibilities:

- Initialization of the Terminal Equipment
- Configuration of the Terminal Equipment
- TEI Assignment and Verification
- Multiple Error Notification
- Unrecoverable Error Notification
- Activation/Deactivation of the Terminal Equipment.

2.6 SYSTEM UTILITIES

The system utilities initializes the HPC system upon power-up, and provide support for various machine specific features of the HPC.

2.6.1 Power-Up Reset Main Task

This task is the entry point upon system power-up. The Main Task is responsible for:

- Initializing the general HPC Hardware.
- Initializing the HPC Executive Data Structures.
- Queuing up the Tasks on the Ready Queue.

The Main Task starts with the highest priority, 255. After running, the Main Task has served its purpose and is removed from the system by waiting on a semaphore which is typically never signaled.

2.6.2 Nonmaskable Interrupt (NMI) Handler

Since terminal power is generally a concern, the HPC can go into an idle, low-power mode when the Terminal Equipment is idle. In this mode the HPC is awakened via an NMI, prompted by a local off hook indication, or by a far-end line

2.0 Functional Description (Continued)

signal detection signal from the SID. Conditions for determining when to go in and out of idle mode are application dependent.

2.6.3 Timer Interrupt Handler

The Timer Interrupt Handler fields interrupts from two of the HPC onboard timers. Timer T0, the Watchdog Timer, overflows every 65536 clock counts. When this occurs the Timer Interrupt Handler mails a message to start the Watchdog Task. Timer T1, the ISDN Software Timer, overflows every 10 ms. The ISDN Software Clock is incremented every tenth Timer T1 overflow, resulting in an ISDN Clock with 100 ms resolution, which is used by the Executive Timer facility.

2.6.4 Watchdog Task

A special task is performed by the HPC's watchdog feature to verify system sanity. The Watchdog Task waits for a mail message that is sent by the Timer Interrupt Handler when Timer T0 overflows. This operation requires that the Watchdog Task be regularly scheduled by the HPC Executive. The Watchdog Task is assigned the highest task priority, 255.

3.0 Ordering Information

3.1 LICENSE AGREEMENT

A license agreement is required for the use and sale of the National Semiconductor ISDN Software. Contact your local National Semiconductor field sales office for more information or contact the factory direct at:

National Semiconductor
ISDN Software Support
M/S 16-174
2900 Semiconductor Drive
Santa Clara, CA 95051
(408) 721-5719

3.2 SOFTWARE ORDER INFORMATION

ISDN software is available in either Object or Source Code format. A Demonstration package is also available. Manuals are included with the Demonstration package and with the Executive and Basic Rate Interface Software packages.

Basic Rate Interface (BRI) software is available for several different central office switches. The generic BRI includes a generalized CCITT Switch Interface.

Each BRI Package contains the following modules:

- Layer 1 Driver (controls S device)
- Layer 2 Driver (controls DMA/HDLC)
- Layer 2 Controller (Q.921)
- Layer 3 Controller (Q.931 Protocol Control)
- Management Entity (Q.921 and Q.931)
- Call Control (Demonstration Application)
- Tracer (Demonstration and Development Tool)

The Multi-Tasking Executive contains two modules:

- Executive Core Module
- Executive Interface Module

The Executive Interface Module is always supplied as source code to allow modification to insert application tasks. A Multi-Tasking Executive is required to run the Basic Rate Interface.

Order Part Number Description

Multi-Tasking Executive

HPC-ISDN-EXEC-O Multi-Tasking Executive Object Code

Basic Rate Interface

HPC-ISDN-BRI-S Basic Rate Interface (Generic) Source Code

HPC-ISDN-BRID-S Basic Rate Interface (DMS-100) Source Code

HPC-ISDN-BRI5-S Basic Rate Interface (5ESS) Source Code

Demonstration Package

HPC-ISDN-PCDEMO ISDN Basic Rate Interface Demonstration (includes Multi-Tasking Executive and Basic Rate Interface Software Manuals)

4.0 Other Related Information

4.1 DEVICE INFORMATION

Additional technical information on devices referenced in this datasheet is available from National:

- HPC16400 High Performance microController
- HPC16083 High Performance microController
- TP3076 COMBO IITM
- TP3420 CCITT S/T Interface

4.2 DEVELOPMENT SUPPORT INFORMATION

Development tools are available for the HPC Family of Microcontrollers. These tools support the ISDN development environment. ISDN software must be ordered separately.

4.2.1 ISDN Demonstration Kit

A kit is available that demonstrates the software and hardware discussed in this datasheet. Included in this kit is a TP3500 development board featuring the HPC16400, TP3070 COMBO II, TP3420 "SID" and ISDN Basic Rate Interface software in ROM. A complete set of manuals are included. This demonstration kit may be ordered from National, part number.

ISDN-TP3500-Kit

4.2.2 Development Systems

Several different Microcontroller-On-Line-Emulator Development Systems are available for hardware and software development of the HPC Family of Microcontrollers. Complete information on Development Systems and Accessories may be found in the Microcontroller Development Support Datasheet.



Section 8
**Appendices/
Physical Dimensions**



Section 8 Contents

Surface Mount	8-3
PLCC Packaging	8-23
Physical Dimensions	8-27
Bookshelf	
Distributors	

Surface Mount

Cost pressures today are forcing many electronics manufacturers to automate their production lines. Surface mount technology plays a key role in this cost-savings trend because:

1. The mounting of devices on the PC board surface eliminates the expense of drilling holes;
2. The use of pick-and-place machines to assemble the PC boards greatly reduces labor costs;
3. The lighter and more compact assembled products resulting from the smaller dimensions of surface mount packages mean lower material costs.

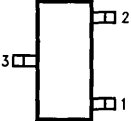
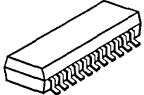
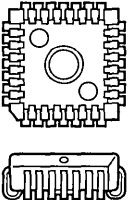
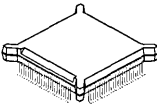
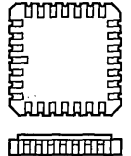
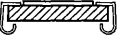
Production processes now permit both surface mount and insertion mount components to be assembled on the same PC board.

SURFACE MOUNT PACKAGING AT NATIONAL

To help our customers take advantage of this new technology, National has developed a line of surface mount packages. Ranging in lead counts from 3 to 360, the package offerings are summarized in Table I.

Lead center spacing keeps shrinking with each new generation of surface mount package. Traditional packages (e.g., DIPs) have a 100 mil lead center spacing. Surface mount packages currently in production (e.g., SOT, SOIC, PCC, LCC, LDCC) have a 50 mil lead center spacing. Surface mount packages in production release (e.g., PQFP) have a 25 mil lead center spacing. Surface mount packages in development (e.g., TAPEPAK®) will have a lead center spacing of only 12–20 mils.

TABLE I. Surface Mount Packages from National

Package Type	Small Outline Transistor (SOT)	Small Outline IC (SOIC)	Plastic Chip Carrier (PCC)	Plastic Quad Flat Pack (PQFP)	Leadless Chip Carrier (LCC) (LDCC)	Leaded Chip Carrier
						
Package Material	Plastic	Plastic	Plastic	Plastic	Ceramic	Ceramic
Lead Bend	Gull Wing	Gull Wing	J-Bend	Gull Wing	—	Gull Wing
Lead Center Spacing	50 Mils	50 Mils	50 Mils	25 Mils	50 Mils	50 Mils
Tape & Reel Option	Yes	Yes	Yes	tbd	No	No
Lead Counts	SOT-23 High Profile SOT-23 Low Profile	SO-8(*) SO-14(*) SO-14 Wide(*) SO-16(*) SO-16 Wide(*) SO-20(*) SO-24(*)	PCC-20(*) PCC-28(*) PCC-44(*) PCC-68 PCC-84 PCC-124	PQFP-84 PQFP-100 PQFP-132 PQFP-196(*) PQFP-244	LCC-18 LCC-20(*) LCC-28 LCC-32 LCC-44 (*) LCC-48 LCC-52 LCC-68 LCC-84 LCC-124	LDCC-44 LDCC-68 LDCC-84 LDCC-124

*In production (or planned) for linear products.

LINEAR PRODUCTS IN SURFACE MOUNT

Linear functions available in surface mount include:

- Op amps
- Comparators
- Regulators
- References
- Data conversion
- Industrial
- Consumer
- Automotive

A complete list of linear part numbers in surface mount is presented in Table III. Refer to the datasheet in the appropriate chapter of this databook for a complete description of the device. In addition, National is continually expanding the list of devices offered in surface mount. If the functions you need do not appear in Table III, contact the sales office or distributor branch nearest you for additional information.

Automated manufacturers can improve their cost savings by using Tape-and-Reel for surface mount devices. Simplified handling results because hundreds-to-thousands of semiconductors are carried on a single Tape-and-Reel pack (see ordering and shipping information—printed later in this section—for a comparison of devices/reel vs. devices/rail for those surface mount package types being used for linear products). With this higher device count per reel (when compared with less than a 100 devices per rail), pick-and-place machines have to be re-loaded less frequently and lower labor costs result.

With Tape-and-Reel, manufacturers save twice—once from using surface mount technology for automated PC board assembly and again from less device handling during shipment and machine set-up.

BOARD CONVERSION

Besides new designs, many manufacturers are converting existing printed circuit board designs to surface mount. The resulting PCB will be smaller, lighter and less expensive to manufacture; but there is one caveat—be careful about the thermal dissipation capability of the surface mount package.

Because the surface mount package is smaller than the traditional dual-in-line package, the surface mount package is not capable of conducting as much heat away as the DIP (i.e., the surface mount package has a higher thermal resistance—see Table II).

The silicon for most National devices can operate up to a 150°C junction temperature (check the datasheet for the rare exception). Like the DIP, the surface mount package can actually withstand an ambient temperature of up to 125°C (although a commercial temperature range device will only be specified for a max ambient temperature of 70°C and an industrial temperature range device will only be specified for a max ambient temperature of 85°C). See AN-336, "Understanding Integrated Circuit Package Power Capabilities", (reprinted in the appendix of each linear databook volume) for more information.

**TABLE II: Surface Mount Package
Thermal Resistance Range***

Package	Thermal Resistance** (θ_{JA} , °C/W)
SO-8	120–175
SO-14	100–140
SO-14 Wide	70–110
SO-16	90–130
SO-16 Wide	70–100
SO-20	60–90
SO-24	55–85
PCC-20	70–100
PCC-28	60–90
PCC-44	40–60

*Actual thermal resistance for a particular device depends on die size. Refer to the datasheet for the actual θ_{JA} value.

**Test conditions: PCB mount (FR4 material), still air (room temperature), copper traces (150 × 20 × 10 mils).

Given a max junction temperature of 150°C and a maximum allowed ambient temperature, the surface mount device will be able to dissipate less power than the DIP device. This factor must be taken into account for new designs.

For board conversion, the DIP and surface mount devices would have to dissipate the same power. This means the surface mount circuit would have a lower maximum allowable ambient temperature than the DIP circuit. For DIP circuits where the maximum ambient temperature required is substantially lower than the maximum ambient temperature allowed, there may be enough margin for safe operation of the surface mount circuit with its lower maximum allowable ambient temperature. But where the maximum ambient temperature required of the DIP current is close to the maximum allowable ambient temperature, the lower maximum ambient temperature allowed for the surface mount circuit may fall below the maximum ambient temperature required. The circuit designer must be aware of this potential pitfall so that an appropriate work-around can be found to keep the surface mount package from being thermally overstressed in the application.

SURFACE MOUNT LITERATURE

National has published extensive literature on the subject of surface mount packaging. Engineers from packaging, quality, reliability, and surface mount applications have pooled their experience to provide you with practical hands-on knowledge about the construction and use of surface mount packages.

The applications note AN-450 "Surface Mounting Methods and their Effect on Product Reliability" is referenced on each SMD datasheet. In addition, "Wave Soldering of Surface Mount Components" is reprinted in this section for your information.

TABLE III. Linear Surface Mount Current Device Listing

Amplifiers and Comparators

Part Number	Part Number
LF347WM	LM392M
LF351M	LM393M
LF451CM	LM741CM
LF353M	LM1458M
	LM2901M
LF355M	LM2902M
LF356M	LM2903M
LF357M	LM2904M
LF444CWM	LM2924M
	LM3403M
LM10CWM	LM4250M
LM10CLWM	LM324M
LM308M	LM339M
LM308AM	LM365WM
LM310M	LM607CM
LM311M	LMC669BCWM
LM318M	LMC669CCWM
LM319M	
LM324M	LF441CM
LM339M	
LM346M	
LM348M	
LM358M	
LM359M	

Regulators and References

Part Number	Part Number
LM317LM	LM2931M-5.0
LF3334M	LM3524M
	LM78L05ACM
LM336M-2.5	LM78L12ACM
LF336BM-2.5	LM78L15ACM
LM336M-5.0	
LM336BM-5.0	LM79L05ACM
LM337LM	LM79L12ACM
	LM79L15ACM
LM385M	LP2951ACM
LM385M-1.2	LP2951CM
LM385BM-1.2	
LM385M-2.5	
LM385BM-2.5	
LM723CM	
LM2931CM	

Data Acquisition Circuits

Part Number	Part Number
ADC0802LCV	ADC1025BCV
ADC0802LCWM	ADC1025CCV
ADC0804LCV	DAC0800LCM
ADC0804LCWM	DAC0801LCM
ADC0808CCV	DAC0802LCM
ADC0809CCV	DAC0806LCM
	DAC0807LCM
ADC0811BCV	DAC0808LCM
ADC0811CCV	DAC0830LCWM
ADC0819BCV	DAC0830LCV
ADC0819CCV	DAC0832LCWM
ADC0820BCV	DAC0832LCV
ADC0820CCV	
ADC0838BCV	
ADC0838CCV	
ADC0841BCV	
ADC0841CCV	
ADC0848BCV	
ADC0848CCV	
ADC1005BCV	
ADC1005CCV	

Industrial Functions

Part Number	Part Number
AH5012CM	LM13600M
LF13331M	LM13700M
LF13509M	LMC555CM
LF13333M	LM567CM
LM555CM	MF4CWM-50
LM556CM	MF4CWM-100
LM567CM	MF6CWM-50
LM1496M	MF10CCWM
LM2917M	MF6CWM-100
	MF5CWM
LM3046M	
LM3086M	
LM3146M	

Commercial and Automotive

Part Number	Part Number
LM386M-1	LM1837M
LM592M	LM1851M
LM831M	LM1863M
LM832M	LM1865M
LM833M	LM1870M
LM837M	LM1894M
LM838M	LM1964V
LM1131CM	LM2893M
	LM3361AM
	LM1881M

Hybrids

Part Number	Part Number
LH0002E	LH0032E
LH4002E	LH0033E

A FINAL WORD

National is a world leader in the design and manufacture of surface mount components.

Because of design innovations such as perforated copper leadframes, our small outline package is as reliable as our DIP—the laws of physics would have meant that a straight “junior copy” of the DIP would have resulted in an “S.O.” package of lower reliability. You benefit from this equivalence of reliability. In addition, our ongoing vigilance at each step of the production process assures that the reliability we designed in stays in so that only devices of the highest quality and reliability are shipped to your factory.

Our surface mount applications lab at our headquarters site in Santa Clara, California continues to research (and publish) methods to make it even easier for you to use surface mount technology. Your problems are our problems.

When you think “Surface Mount”—think “National”!

Ordering and Shipping Information

When you order a surface mount semiconductor, it will be in one of the several available surface mount package types. Specifying the Tape-and-Reel method of shipment means that you will receive your devices in the following quantities per Tape-and-Reel pack: SMD devices can also be supplied in conventional conductive rails.

Package	Package Designator	Max/Rail	Per Reel*
SO-8	M	100	2500
SO-14	M	50	2500
SO-14 Wide	WM	50	1000
SO-16	M	50	2500
SO-16 Wide	WM	50	1000
SO-20	M	40	1000
SO-24	M	30	1000
PCL-20	V	50	1000
PCL-28	V	40	1000
PCL-44	V	25	500
PQFP-196	VF	TBD	—
TP-40	TP	100	TBD
LCC-20	E	50	—
LCC-44	E	25	—

*Incremental ordering quantities. (National Semiconductor reserves the right to provide a smaller quantity of devices per Tape-and-Reel pack to preserve lot or date code integrity. See example below.)

Example: You order 5,000 LM324M ICs shipped in Tape-and-Reel.

- Case 1: All 5,000 devices have the same date code
 - You receive 2 SO-14 (Narrow) Tape-and-Reel packs, each having 2500 LM324M ICs
- Case 2: 3,000 devices have date code A and 2,000 devices have date code B
 - You receive 3 SO-14 (Narrow) Tape-and-Reel packs as follows:
 - Pack #1 has 2,500 LM324M ICs with date code A
 - Pack #2 has 500 LM324M ICs with date code A
 - Pack #3 has 2,000 LM324M ICs with date code B

Short-Form Procurement Specification

TAPE FORMAT

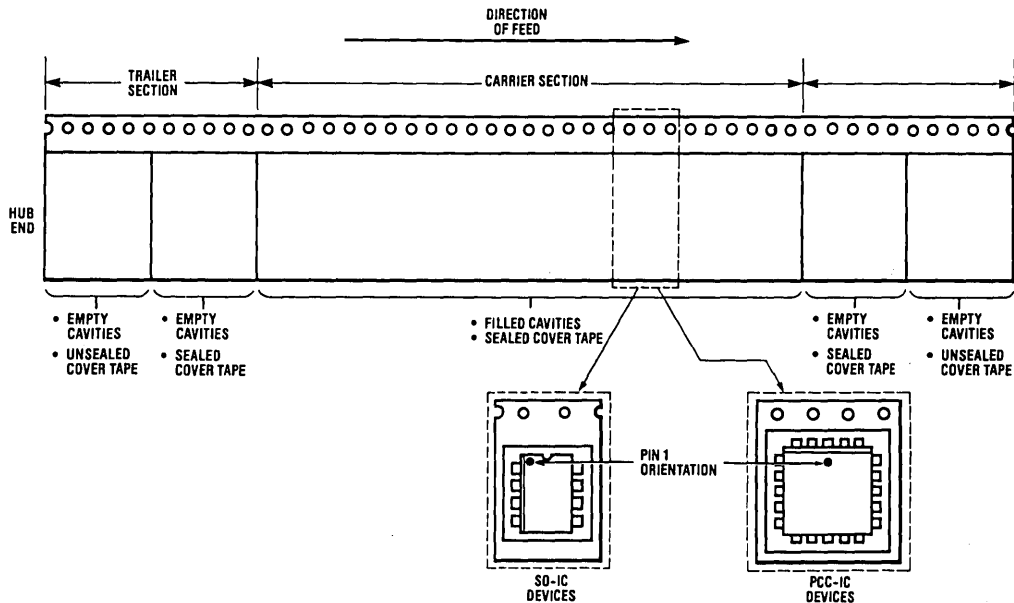
→ Direction of Feed

	Trailer (Hub End)*		Carrier*	Leader (Start End)*	
	Empty Cavities, min (Unsealed Cover Tape)	Empty Cavities, min (Sealed Cover Tape)	Filled Cavities (Sealed Cover Tape)	Empty Cavities, min (Sealed Cover Tape)	Empty Cavities, min (Unsealed Cover Tape)
Small Outline IC					
SO-8 (Narrow)	2	2	2500	5	5
SO-14 (Narrow)	2	2	2500	5	5
SO-14 (Wide)	2	2	1000	5	5
SO-16 (Narrow)	2	2	2500	5	5
SO-16 (Wide)	2	2	1000	5	5
SO-20 (Wide)	2	2	1000	5	5
SO-24 (Wide)	2	2	1000	5	5
Plastic Chlp Carrier IC					
PCC-20	2	2	1000	5	5
PCC-28	2	2	750	5	5
PCC-44	2	2	500	5	5

*The following diagram identifies these sections of the tape and Pin #1 device orientation.

Short-Form Procurement Specification (Continued)

DEVICE ORIENTATION



TL/DD/11325-7

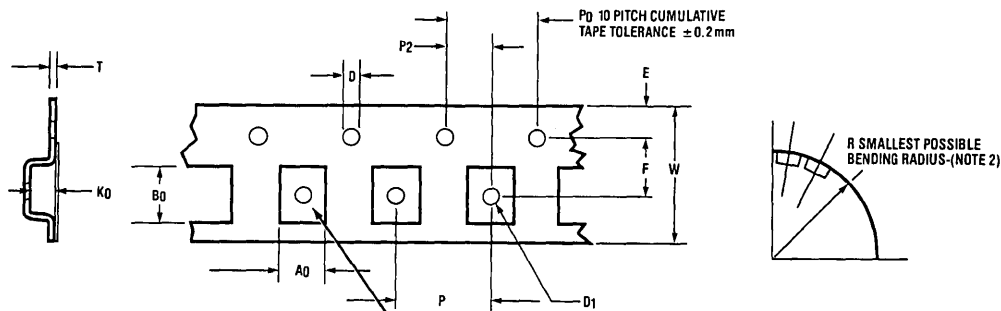
MATERIALS

- Cavity Tape: Conductive PVC (less than 10^5 Ohms/Sq)
- Cover Tape: Polyester
 - (1) Conductive cover available

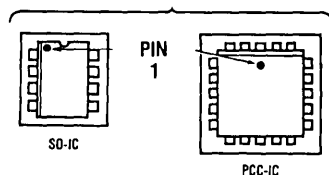
• Reel:

- (1) Solid 80 pt fibreboard (standard)
- (2) Conductive fibreboard available
- (3) Conductive plastic (PVC) available

TAPE DIMENSIONS (24 Millimeter Tape or Less)



DEVICE ORIENTATION



TL/DD/11325-8

Short-Form Procurement Specification (Continued)

	W	P	F	E	P ₂	P ₀	D	T	A ₀	B ₀	K ₀	D ₁	R
Small Outline IC													
SO-8 (Narrow)	12 ± .30	8.0 ± .10	5.5 ± .05	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	6.4 ± .10	5.2 ± .10	2.1 ± .10	1.55 ± .05	30
SO-14 (Narrow)	16 ± .30	8.0 ± .10	7.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	6.5 ± .10	9.0 ± .10	2.1 ± .10	1.55 ± .05	40
SO-14 (Wide)	16 ± .30	12.0 ± .10	7.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	10.9 ± .10	9.5 ± .10	3.0 ± .10	1.55 ± .05	40
SO-16 (Narrow)	16 ± .30	8.0 ± .10	7.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	6.5 ± .10	10.3 ± .10	2.1 ± .10	1.55 ± .05	40
SO-16 (Wide)	16 ± .30	12.0 ± .10	7.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	10.9 ± .10	10.76 ± .10	3.0 ± .10	1.55 ± .05	40
SO-20 (Wide)	24 ± .30	12.0 ± .10	11.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	10.9 ± .10	13.3 ± .10	3.0 ± .10	2.05 ± .05	50
SO-24 (Wide)	24 ± .30	12.0 ± .10	11.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	10.9 ± .10	15.85 ± .10	3.0 ± .10	2.05 ± .05	50
Plastic Chip Carrier IC													
PCC-20	16 ± .30	12.0 ± .10	7.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	9.3 ± .10	9.3 ± .10	4.9 ± .10	1.55 ± .05	40
PCC-28	24 ± .30	16.0 ± .10	11.5 ± .10	1.75 ± .10	2.0 ± .05	4.0 ± .10	1.55 ± .05	.30 ± .10	13.0 ± .10	13.0 ± .10	4.9 ± .10	2.05 ± .05	50

Note 1: A₀, B₀ and K₀ dimensions are measured 0.3 mm above the inside wall of the cavity bottom.

Note 2: Tape with components shall pass around a mandril radius R without damage.

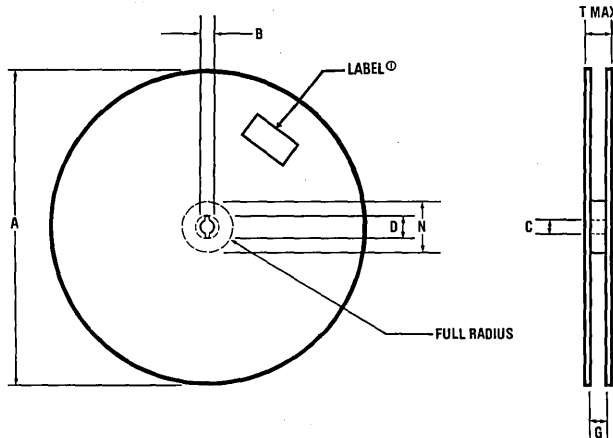
Note 3: Cavity tape material shall be PVC conductive (less than 10⁵ Ohms/Sq).

Note 4: Cover tape material shall be polyester (30-65 grams peel-back force).

Note 5: D₁ Dimension is centered within cavity.

Note 6: All dimensions are in millimeters.

REEL DIMENSIONS



STAR™ Surface Mount Tape and Reel

TL/DD/11325-9

Short-Form Procurement Specifications (Continued)

		A (Max)	B (Min)	C	D (Min)	N (Min)	G	T (Max)
12 mm Tape	SO-8 (Narrow)	$\frac{(13.00)}{(330)}$	$\frac{.059}{1.5}$	$\frac{.512 \pm .002}{13 \pm 0.05}$	$\frac{.795}{20.2}$	$\frac{1.969}{50}$	$\frac{0.488^{+.078}_{-.000}}{12.4^{+2}_{-0}}$	$\frac{.724}{18.4}$
16 mm Tape	SO-14 (Narrow) SO-14 (Wide) SO-16 (Narrow) SO-16 (Wide) PCC-20	$\frac{(13.00)}{(330)}$	$\frac{.059}{1.5}$	$\frac{.512 \pm .002}{13 \pm 0.05}$	$\frac{.795}{20.2}$	$\frac{1.969}{50}$	$\frac{0.646^{+.078}_{-.000}}{16.4^{+2}_{-0}}$	$\frac{.882}{22.4}$
24 mm Tape	SO-20 (Wide) SO-24 (Wide) PCC-28	$\frac{(13.00)}{(330)}$	$\frac{.059}{1.5}$	$\frac{.512 \pm .002}{13 \pm 0.05}$	$\frac{.795}{20.2}$	$\frac{1.969}{50}$	$\frac{0.960^{+.078}_{-.000}}{24.4^{+2}_{-0}}$	$\frac{1.197}{30.4}$
32 mm Tape	PCC-44	$\frac{(13.00)}{(330)}$	$\frac{.059}{1.5}$	$\frac{.512 \pm .002}{13 \pm 0.05}$	$\frac{.795}{20.2}$	$\frac{1.969}{50}$	$\frac{1.276^{+.078}_{-.000}}{32.4^{+2}_{-0}}$	$\frac{1.512}{38.4}$

Units: $\frac{\text{Inches}}{\text{Millimeters}}$

Material: Paperboard (Non-Flaking)

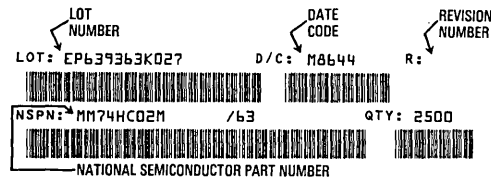
LABEL

Human and Machine Readable Label is provided on reel. A variable (C.P.I) density code 39 is available. NSC STD label (7.6 C.P.I.)

FIELD

Lot Number
Date Code
Revision Level
National Part No. I.D.
Qty.

EXAMPLE



TL/DD/11325-10

Fields are separated by at least one blank space.

Future Tape-and-Reel packs will also include a smaller-size bar code label (high-density code 39) at the beginning of the tape. (This tape label is not available on current production.) National Semiconductor will also offer additional labels containing information per your specific specification.

Wave Soldering of Surface Mount Components

ABSTRACT

In facing the upcoming surge of "surface mount technology", many manufacturers of printed circuit boards have taken steps to convert some portions of their boards to this new process. However, as the availability of surface mount components is still limited, many have taken to mixing the lead-inserted standard dual-in-line packages (DIPs) with the surface mounted devices (SMDs). Furthermore, to take advantage of using both sides of the board, surface-mounted components are generally adhered to the bottom side of the board while the top side is reserved for the conventional lead-inserted packages. If processed through a wave solder machine, the semiconductor components are now subjected to extra thermal stresses (now that the components are totally immersed into the molten solder).

A discussion of the effect of wave soldering on the reliability of plastic semiconductor packages follows. This is intended to highlight the limitations which should be understood in the use of wave soldering of surface mounted components.

ROLE OF WAVE-SOLDERING IN APPLICATION OF SMDs

The generally acceptable methods of soldering SMDs are vapor phase reflow soldering and IR reflow soldering, both requiring application of solder paste on PW boards prior to placement of the components. However, sentiment still exists for retaining the use of the old wave-soldering machine.

Wave Soldering of Surface Mount Components (Continued)

The reasons being:

- 1) Most PC Board Assembly houses already possess wave soldering equipment. Switching to another technology such as vapor phase soldering requires substantial investment in equipment and people.
- 2) Due to the limited number of devices that are surface mount components, it is necessary to mix both lead inserted components and surface mount components on the same board.
- 3) Some components such as relays and switches are made of materials which would not be able to survive the temperature exposure in a vapor phase or IR furnace.

PW BOARD ASSEMBLY PROCEDURES

There are two considerations in which through-hole ICs may be combined with surface mount components on the PW Board:

- a) Whether to mount ICs on one or both sides of the board.
- b) The sequence of soldering using Vapor Phase, IR or Wave Soldering singly or combination of two or more methods.

The various processes that may be employed are:

A) Wave Solder before Vapor/IR reflow solder.

1. Components on the same side of PW Board.
 - Lead insert standard DIPs onto PW Board Wave solder (conventional)
 - Wash and lead trim
 - Dispense solder paste on SMD pads
 - Pick and place SMDs onto PW Board
 - Bake
 - Vapor phase/IR reflow
 - Clean
2. Components on opposite side of PW Board.
 - Lead insert standard DIPs onto PW Board Wave Solder (conventional)
 - Clean and lead trim
 - Invert PW Board
 - Dispense solder paste on SMD pads
 - Dispense drop of adhesive on SMD sites (optional for smaller components)
 - Pick and place SMDs onto board
 - Bake/Cure
 - Invert board to rest on raised fixture
 - Vapor/IR reflow soldering
 - Clean

B) Vapor/IR reflow solder then Wave Solder.

1. Components on the same side of PW Board.
 - Solder paste screened on SMD side of Printed Wire Board
 - Pick and place SMDs
 - Bake
 - Vapor/IR reflow
 - Lead insert on same side as SMDs
 - Wave solder
 - Clean and trim underside of PCB

C) Vapor/IR reflow only.

1. Components on the same side of PW Board.
 - Trim and form standard DIPs in "gull wing" configuration
 - Solder paste screened on PW Board
 - Pick and place SMDs and DIPs
 - Bake
 - Vapor/IR reflow
 - Clean
2. Components on opposite sides of PW Board.
 - Solder paste screened on SMD-side of Printed Wire Board
 - Adhesive dispensed at central location of each component
 - Pick and place SMDs
 - Bake
 - Solder paste screened on all pads on DIP-side or alternatively apply solder rings (performs) on leads
 - Lead insert DIPs
 - Vapor/IR reflow
 - Clean and lead trim

D) Wave Soldering Only

1. Components on opposite sides of PW Board.
 - Adhesive dispense on SMD side of PW Board
 - Pick and place SMDs
 - Cure adhesive
 - Lead insert top side with DIPs
 - Wave solder with SMDs down and into solder bath
 - Clean and lead trim

All of the above assembly procedures can be divided into three categories for I.C. Reliability considerations:

- 1) Components are subjected to both a vapor phase/IR heat cycle then followed by a wave-solder heat cycle or vice versa.
- 2) Components are subjected to only a vapor phase/IR heat cycle.
- 3) Components are subjected to wave-soldering only and SMDs are subjected to heat by immersion into a solder pot.

Of these three categories, the last is the most severe regarding heat treatment to a semiconductor device. However, note that semiconductor molded packages generally possess a coating of solder on their leads as a final finish for solderability and protection of base leadframe material. Most semiconductor manufacturers solder-plate the component leads, while others perform hot solder dip. In the latter case the packages may be subjected to total immersion into a hot solder bath under controlled conditions (manual operation) or be partially immersed while in a 'pallet' where automatic wave or DIP soldering processes are used. It is, therefore, possible to subject SMDs to solder heat under certain conditions and not cause catastrophic failures.

Wave Soldering of Surface Mount Components (Continued)

THERMAL CHARACTERISTICS OF MOLDED INTEGRATED CIRCUITS

Since Plastic DIPs and SMDs are encapsulated with a thermoset epoxy, the thermal characteristics of the material generally correspond to a TMA (Thermo-Mechanical Analysis) graph. The critical parameters are (a) its Linear thermal expansion characteristics and (b) its glass transition temperature after the epoxy has been fully cured. A typical TMA graph is illustrated in *Figure 1*. Note that the epoxy changes to a higher thermal expansion once it is subjected to temperatures exceeding its glass transition temperature. Metals (as used on lead frames, for example) do not have this characteristic and generally will have a consistent Linear thermal expansion over the same temperature range.

In any good reliable plastic package, the choice of lead frame material should be such to match its thermal expansion properties to that of the encapsulating epoxy. In the event that there is a mismatch between the two, stresses can build up at the interface of the epoxy and metal. There now exists a tendency for the epoxy to separate from the metal lead frame in a manner similar to that observed on bi-metallic thermal range.

In most cases when the packages are kept at temperatures below their glass transition, there is a small possibility of separation at the epoxy-metal interface. However, if the package is subjected to temperature above its glass-transition temperature, the epoxy will begin to expand much faster than the metal and the probability of separation is greatly increased.

CONVENTIONAL WAVE-SOLDERING

Most wave-soldering operations occur at temperatures between 240–260°C. Conventional epoxies for encapsulation have glass-transition temperature between 140–170°C. An I.C. directly exposed to these temperatures risks its long term functionality due to epoxy/metal separation.

Fortunately, there are factors that can reduce that element of risk:

- 1) The PW board has a certain amount of heat-sink effect and tends to shield the components from the temperature of the solder (if they were placed on the top side of the board). In actual measurements, DIPs achieve a temperature between 120–150°C in a 5-second pass over the solder. This accounts for the fact that DIPs mounted in the conventional manner are reliable.
- 2) In conventional soldering, only the tip of each lead in a DIP would experience the solder temperature because the epoxy and die are standing above the PW board and out of the solder bath.

EFFECT ON PACKAGE PERFORMANCE BY EPOXY-METAL SEPARATION

In wave soldering, it is necessary to use fluxes to assist the solderability of the components and PW boards. Some facilities may even process the boards and components through some form of acid cleaning prior to the soldering temperature. If separation occurs, the flux residues and acid residues (which may be present owing to inadequate cleaning) will be forced into the package mainly by capillary action as the residues move away from the solder heat source. Once the package is cooled, these contaminants are now trapped within the package and are available to diffuse with moisture from the epoxy over time. It should be noted that electrical tests performed immediately after soldering generally will give no indication of this potential problem. In any case, the end result will be corrosion of the chip metallization over time and premature failure of the device in the field.

VAPOR PHASE/IR REFLOW SOLDERING

In both vapor phase and IR reflow soldering, the risk of separation between epoxy/metal can also be high. Operating temperatures are 215°C (vapor phase) or 240°C (IR) and duration may also be longer (30 sec–60 sec). On the same theoretical basis, there should also be separation. However, in both these methods, solder paste is applied to the pads of the boards; no fluxes are used. Also, the devices are not immersed into the hot solder. This reduces the possibility of solder forcing itself into the epoxy-lead frame interface. Furthermore, in the vapor phase system, the soldering environment is "oxygen-free" and considered "contaminant free". Being so, it could be visualized that as far as reliability with respect to corrosion, both of these methods are advantageous over wave soldering.

BIAS MOISTURE TEST

A bias moisture test was designed to determine the effect on package performance. In this test, the packages are pressured in a stream chamber to accelerate penetration of moisture into the package. An electrical bias is applied on the device. Should there be any contaminants trapped within the package, the moisture will quickly form an electrolyte and cause the electrodes (which are the lead fingers), the gold wire and the aluminum bond-pads of the silicon device to corrode. The aluminum bond-pads, being the weakest link of the system, will generally be the first to fail.

This proprietary accelerated bias/moisture pressure-test is significant in relation to the life test condition at 85°C and

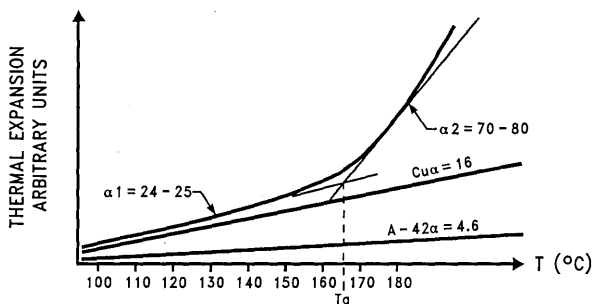


FIGURE 1. Thermal Expansion and Glass Transition Temperature

TL/DD/11325-11

Wave Soldering of Surface Mount Components (Continued)

85% relative humidity. Once cycle of approximately 100 hours has been shown to be equivalent to 2000 hours in the 85/85 condition. Should the packages start to fail within the first cycle in the test, it is anticipated that the boards with these components in the harsh operating environment (85°C/85% RH) will experience corrosion and eventual electrical failures within its first 2000 hours of operation.

Whether this is significant to a circuit board manufacturer will obviously be dependent on the products being manufactured and the workmanship or reliability standards. Generally in systems with a long warranty and containing many components, it is advisable both on a reputation and cost basis to have the most reliable parts available.

TEST RESULTS

The comparison of vapor phase and wave-soldering upon the reliability of molded Small-Outline packages was performed using the bias moisture test (see Table IV). It is clearly seen that vapor phase reflow soldering gave more consistent results. Wave-soldering results were based on manual operation giving variations in soldering parameters such as temperature and duration.

TABLE IV. Vapor Phase vs. Wave Solder

1. Vapor phase (60 sec. exposure @ 215°C)
= 9 failures/1723 samples
= 0.5% (average over 32 sample lots)
2. Wave solder (2 sec total immersion @ 260°C)
= 16 failures/1201 samples
= 1.3% (average over 27 sample lots)
Package: SO-14 lead
Test: Bias moisture test 85% R.H.,
85°C for 2000 hours
Device: LM324M

In Table V we examine the tolerance of the Small-Outlined (SOIC) package to varying immersion time in a hot solder pot. SO-14 lead molded packages were subjected to the bias moisture test after being treated to the various soldering conditions and repeated four (4) times. End point was an electrical test after an equivalent of 4000 hours 85/85 test. Results were compared for packages by itself against packages which were surface-mounted onto a FR-4 printed wire board.

**TABLE V. Summary of Wave Solder Results
(85% R.H./85°C Bias Moisture Test, 2000 hours)
(# Failures/Total Tested)**

	Unmounted	Mounted
Control/Vapor Phase 15 sec @ 215°C	0/114	0/84
Solder Dip 2 sec @ 260°C	2/144 (1.4%)	0/85
Solder Dip 4 sec @ 260°C	—	0/83
Solder Dip 6 sec @ 260°C	13/248 (5.2%)	1/76 (1.3%)
Solder Dip 10 sec @ 260°C	14/127 (11.0%)	3/79 (3.8%)
Package: SO-14 lead		
Device: LM324M		

Since the package is of very small mass and experiences a rather sharp thermal shock followed by stresses created by the mismatch in expansion, the results show the package being susceptible to failures after being immersed in excess of 6 seconds in a solder pot. In the second case where the packages were mounted, the effect of severe temperature excursion was reduced. In the second case where the packages were mounted, the effect of severe temperature excursion was reduced. In any case, because of the repeated treatment, the package had failures when subjected in excess of 6 seconds immersion in hot solder. The safety margin is therefore recommended as maximum 4 seconds immersion. If packages were immersed longer than 4 seconds, there is a probable chance of finding some long term reliability failures even though the immediate electrical test data could be acceptable.

Finally, Table VI examines the bias moisture test performed on surface mount (SOIC) components manufactured by various semiconductor houses. End point was an electrical test after an equivalent of 6000 hours in a 85/85 test. Failures were analyzed and corrosion was checked for in each case to detect flaws in package integrity.

**TABLE VI. U.S. Manufacturers Integrated Circuits
Reliability in Various Solder Environments
(# Failure/Total Tested)**

Package SO-8	Vapor Phase 30 sec	Wave Solder 2 sec	Wave Solder 4 sec	Wave Solder 6 sec	Wave Solder 10 sec
Manuf A	8/30*	1/30*	0.30	12/30*	16/30*
Manuf B	2/30*	8/30*	2/30*	22/30*	20/30*
Manuf C	0/30	0/29	0/29	0/30	0/30
Manuf D	1/30*	0/30	12/30*	14/30*	2/30*
Manuf E	1/30**	0/30	0/30	0/30	0/30
Manuf F	0/30	0/30	0/30	0/30	0/30
Manuf G	0/30	0/30	0/30	0/30	0/30

*Corrosion-failures

**No Visual Defects—Non-corrosion failures

Test: Accelerated Bias Moisture Test; 85% R.H./85°C, 6000 equivalent hours.

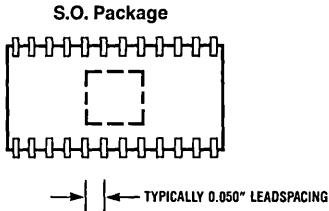
SUMMARY

Based on the results presented, it is noted that surface-mounted components are as reliable as standard molded DIP packages. Whereas DIPs were never processed by being totally immersed in a hot solder wave during printed circuit board soldering, surface mounted components such as SOICs (Small Outline) are expected to survive a total immersion in the hot solder in order to capitalize on maximum population on boards. Being constructed from a thermoset plastic of relatively low Tg compared to the soldering temperature, the ability of the package to survive is dependent on the time of immersion and also the cleanliness of material. The results indicate that one should limit the immersion time of package in the solder wave to a maximum of 4 seconds in order to truly duplicate the reliability of a DIP. As the package size is reduced, as in a SO-8 lead, the requirement becomes even more critical. This is shown by the various manufacturers' performance. Results indicate there is room for improvement since not all survived the hot solder immersion without compromise to lower reliability.

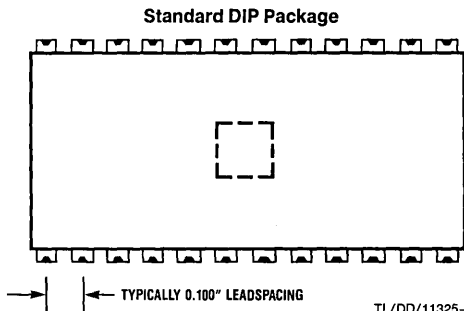
Small Outline (SO) Package Surface Mounting Methods—Parameters and Their Effect on Product Reliability

The SO (small outline) package has been developed to meet customer demand for ever-increasing miniaturization and component density.

COMPONENT SIZE COMPARISON



TL/DD/11325-12



TL/DD/11325-13

Because of its small size, reliability of the product assembled in SO packages needs to be carefully evaluated.

SO packages at National were internally qualified for production under the condition that they be of comparable reliability performance to a standard dual in line package under all accelerated environmental tests. *Figure A* is a summary of accelerated bias moisture test performance on 30V bipolar and 15V CMOS product assembled in SO and DIP (control) packages.

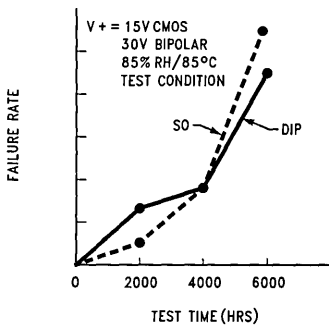


FIGURE A

TL/DD/11325-14

In order to achieve reliability performance comparable to DIPs—SO packages are designed and built with materials and processes that effectively compensate for their small size.

All SO packages tested on 85%RA, 85°C were assembled on PC conversion boards using vapor-phase reflow soldering. With this approach we are able to measure the effect of surface mounting methods on reliability of the process. As illustrated in *Figure A* no significant difference was detected between the long term reliability performance of surface mounted S.O. packages and the DIP control product for up to 6000 hours of accelerated 85%/85°C testing.

SURFACE-MOUNT PROCESS FLOW

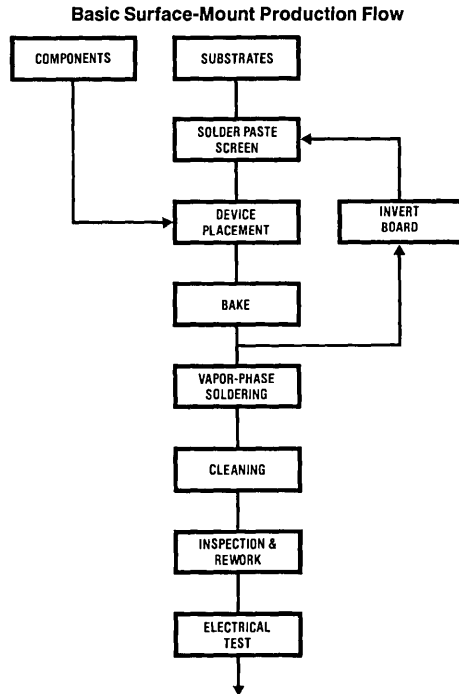
The standard process flowcharts for basic surface-mount operation and mixed-lead insertion/surface-mount operations, are illustrated on the following pages.

Usual variations encountered by users of SO packages are:

- Single-sided boards, surface-mounted components only.
- Single-sided boards, mixed-lead inserted and surface-mounted components.
- Double-sided boards, surface-mounted components only.
- Double-sided boards, mixed-lead inserted and surface-mounted components.

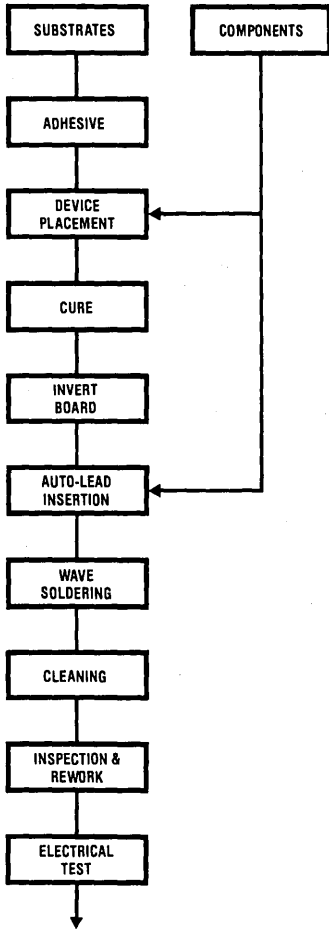
In consideration of these variations, it became necessary for users to utilize techniques involving wave soldering and adhesive applications, along with the commonly-used vapor-phase solder reflow soldering technique.

PRODUCTION FLOW



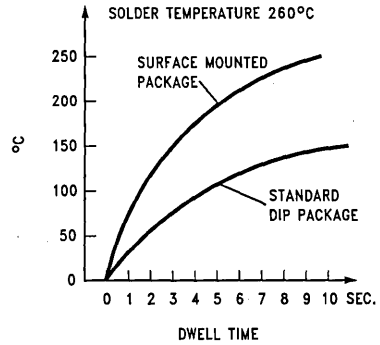
TL/DD/11325-15

Mixed Surface-Mount and Axial-Leaded Insertion Components Production Flow



TL/DD/11325-16

Thermal stress of the packages during surface-mounting processing is more severe than during standard DIP PC board mounting processes. *Figure B* illustrates package temperature versus wave soldering dwell time for surface mounted packages (components are immersed into the molten solder) and the standard DIP wave soldering process. (Only leads of the package are immersed into the molten solder).

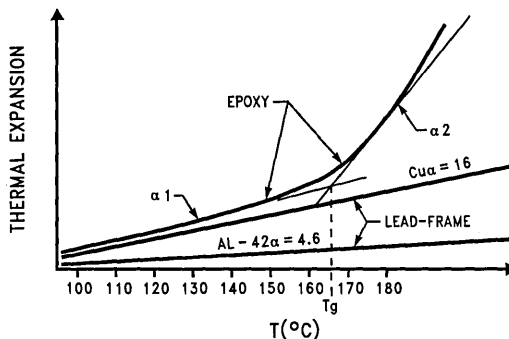


TL/DD/11325-17

FIGURE B

For an ideal package, the thermal expansion rate of the encapsulant should match that of the leadframe material in order for the package to maintain mechanical integrity during the soldering process. Unfortunately, a perfect matchup of thermal expansion rates with most presently used packaging materials is scarce. The problem lies primarily with the epoxy compound.

Normally, thermal expansion rates for epoxy encapsulant and metal lead frame materials are linear and remain fairly close at temperatures approaching 160°C, *Figure C*. At lower temperatures the difference in expansion rate of the two materials is not great enough to cause interface separation. However, when the package reaches the glass-transition temperature (T_g) of epoxy (typically 160–165°C), the thermal expansion rate of the encapsulant increases sharply, and the material undergoes a transition into a plastic state. The epoxy begins to expand at a rate three times or more greater than the metal leadframe, causing a separation at the interface.



TL/DD/11325-18

FIGURE C

When this happens during a conventional wave soldering process using flux and acid cleaners, process residues and even solder can enter the cavity created by the separation and become entrapped when the material cools. These contaminants can eventually diffuse into the interior of the package, especially in the presence of moisture. The result is die contamination, excessive leakage, and even catastrophic failure. Unfortunately, electrical tests performed immediately following soldering may not detect potential flaws.

Most soldering processes involve temperatures ranging up to 260°C, which far exceeds the glass-transition temperature of epoxy. Clearly, circuit boards containing SMD packages require tighter process controls than those used for boards populated solely by DIPs.

Figure D is a summary of accelerated bias moisture test performance on the 30V bipolar process.

Group 1 — Standard DIP package

Group 2 — SO packages vapor-phase reflow soldered on PC boards

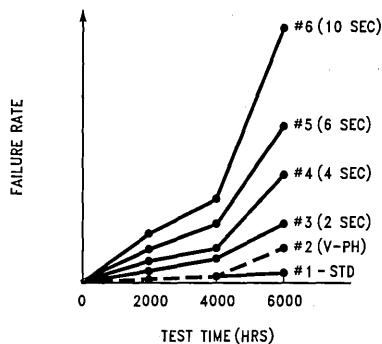
Group 3-6 SO packages wave soldered on PC boards

Group 3 — dwell time 2 seconds

4 — dwell time 4 seconds

5 — dwell time 6 seconds

6 — dwell time 10 seconds



TL/DD/11325-19

FIGURE D

It is clear based on the data presented that SO packages soldered onto PC boards with the vapor phase reflow process have the best long term bias moisture performance and this is comparable to the performance of standard DIP packages. The key advantage of reflow soldering methods is the clean environment that minimized the potential for contamination of surface mounted packages, and is preferred for the surface-mount process.

When wave soldering is used to surface mount components on the board, the dwell time of the component under molten solder should be no more than 4 seconds, preferably under 2 seconds in order to prevent damage to the component. Non-Halide, or (organic acid) fluxes are highly recommended.

PICK AND PLACE

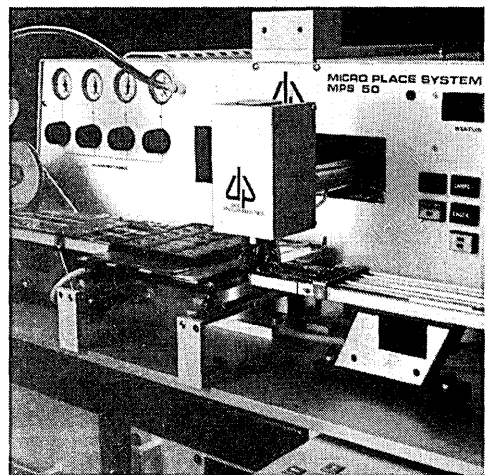
The choice of automatic (all generally programmable) pick-and-place machines to handle surface mounting has grown considerably, and their selection is based on individual needs and degree of sophistication.

The basic component-placement systems available are classified as:

- (a) In-line placement
 - Fixed placement stations
 - Boards indexed under head and respective components placed
- (b) Sequential placement
 - Either a X-Y moving table system or a θ , X-Y moving pickup system used
 - Individual components picked and placed onto boards
- (c) Simultaneous placement
 - Multiple pickup heads
 - Whole array of components placed onto the PCB at the same time
- (d) Sequential/simultaneous placement
 - X-Y moving table, multiple pickup heads system
 - Components placed on PCB by successive or simultaneous actuation of pickup heads

The SO package is treated almost the same as surface-mount, passive components requiring correct orientation in placement on the board.

Pick and Place Action



TL/DD/11325-20

BAKE

This is recommended, despite claims made by some solder paste suppliers that this step be omitted.

The functions of this step are:

- Holds down the solder globules during subsequent reflow soldering process and prevents expulsion of small solder balls.
- Acts as an adhesive to hold the components in place during handling between placement to reflow soldering.
- Holds components in position when a double-sided surface-mounted board is held upside down going into a vapor-phase reflow soldering operation.
- Removes solvents which might otherwise contaminate other equipment.
- Initiates activator cleaning of surfaces to be soldered.
- Prevents moisture absorption.

The process is moreover very simple. The usual schedule is about 20 minutes in a 65°C–95°C (dependent on solvent system of solder paste) oven with adequate venting. Longer bake time is not recommended due to the following reasons:

- The flux will degrade and affect the characteristics of the paste.
- Solder globules will begin to oxidize and cause solderability problems.
- The paste will creep and after reflow, may leave behind residues between traces which are difficult to remove and vulnerable to electro-migration problems.

REFLOW SOLDERING

There are various methods for reflowing the solder paste, namely:

- Hot air reflow
- Infrared heating (furnaces)
- Convectional oven heating
- Vapor-phase reflow soldering
- Laser soldering

For SO applications, hot air reflow/infrared furnace may be used for low-volume production or prototype work, but vapor-phase soldering reflow is more efficient for consistency and speed. Oven heating is not recommended because of "hot spots" in the oven and uneven melting may result. Laser soldering is more for specialized applications and requires a great amount of investment.

HOT GAS REFLOW/INFRARED HEATING

A hand-held or table-mount air blower (with appropriate orifice mask) can be used.

The boards are preheated to about 100°C and then subjected to an air jet at about 260°C. This is a slow process and results may be inconsistent due to various heat-sink properties of passive components.

Use of an infrared furnace is the next step to automating the concept, except that the heating is promoted by use of IR lamps or panels. The main objection to this method is that certain materials may heat up at different rates under IR radiation and may result in damage to these components (usually sockets and connectors). This could be minimized by using far-infrared (non-focused) system.

VAPOR-PHASE REFLOW SOLDERING

Currently the most popular and consistent method, vapor-phase soldering utilizes a fluoroinert fluid with excellent heat-transfer properties to heat up components until the solder paste reflows. The maximum temperature is limited by the vapor temperature of the fluid.

The commonly used fluids (supplied by 3M Corp) are:

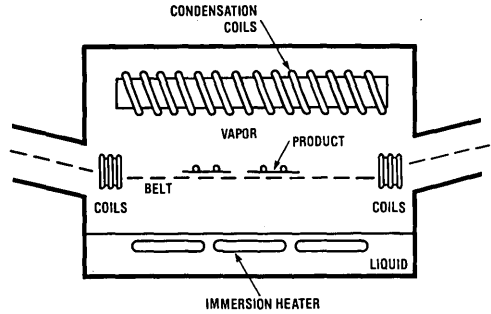
- FC-70, 215°C vapor (most applications) or FX-38
- FC-71, 253°C vapor (low-lead or tin-plate)

HTC, Concord, CA, manufactures equipment that utilizes this technique, with two options:

- Batch systems, where boards are lowered in a basket and subjected to the vapor from a tank of boiling fluid.
- In-line conveyerized systems, where boards are placed onto a continuous belt which transports them into a concealed tank where they are subjected to an environment of hot vapor.

Dwell time in the vapor is generally on the order of 15–30 seconds (depending on the mass of the boards and the loading density of boards on the belt).

In-Line Conveyerized Vapor-Phase Soldering



TL/DD/11325-21

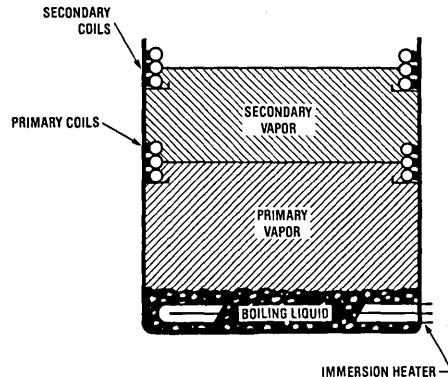
The question of thermal shock is asked frequently because of the relatively sharp increase in component temperature from room temperature to 215°C. SO packages mounted on representative boards have been tested and have shown little effect on the integrity of the packages. Various packages, such as cerdips, metal cans and TO-5 cans with glass seals, have also been tested.

Vapor-Phase Furnace



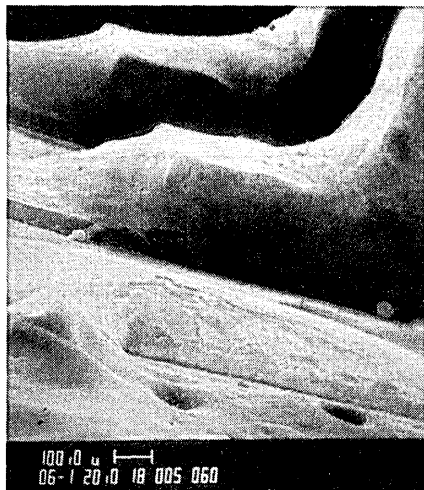
TL/DD/11325-22

Batch-Fed Production Vapor-Phase Soldering Unit



TL/DD/11325-23

Solder Joints on a SO-14 Package on PCB



TL/DD/11325-24

Solder Joints on a SO-14 Package on PCB



TL/DD/11325-25

PRINTED CIRCUIT BOARD

The SO package is molded out of clean, thermoset plastic compound and has no particular compatibility problems with most printed circuit board substrates.

The package can be reliably mounted onto substrates such as:

- G10 or FR4 glass/resin
- FR5 glass/resin systems for high-temperature applications
- Polyimide boards, also high-temperature applications
- Ceramic substrates

General requirements for printed circuit boards are:

- Mounting pads should be solder-plated whenever applicable.
- Solder masks are commonly used to prevent solder bridging of fine lines during soldering.

The mask also protects circuits from processing chemical contamination and corrosion.

If coated over pre-tinned traces, residues may accumulate at the mask/trace interface during subsequent reflow, leading to possible reliability failures.

Recommended application of solder resist on bare, clean traces prior to coating exposed areas with solder.

General requirements for solder mask:

- Good pattern resolution.
- Complete coverage of circuit lines and resistance to flaking during soldering.
- Adhesion should be excellent on substrate material to keep off moisture and chemicals.
- Compatible with soldering and cleaning requirements.

SOLDER PASTE SCREEN PRINTING

With the initial choice of printed circuit lithographic design and substrate material, the first step in surface mounting is the application of solder paste.

The typical lithographic "footprints" for SO packages are illustrated below. Note that the 0.050" lead center-center spacing is not easily managed by commercially-available air pressure, hand-held dispensers.

Using a stainless-steel, wire-mesh screen stencilled with an emulsion image of the substrate pads is by far the most common and well-tried method. The paste is forced through the screen by a V-shaped plastic squeegee in a sweeping manner onto the board placed beneath the screen.

The setup for SO packages has no special requirement from that required by other surface-mounted, passive components. Recommended working specifications are:

- Use stainless-steel, wire-mesh screens, #80 or #120, wire diameter 2.6 mils. Rule of thumb: mesh opening should be approximately 2.5–5 times larger than the average particle size of paste material.
- Use squeegee of Durometer 70.
- Experimentation with squeegee travel speed is recommended, if available on machine used.
- Use solder paste of mesh 200–325.
- Emulsion thickness of 0.005" usually used to achieve a solder paste thickness (wet) of about 0.008" typical.
- Mesh pattern should be 90 degrees, square grid.
- Snap-off height of screen should not exceed $\frac{1}{8}$ ", to avoid damage to screens and minimize distortion.

SOLDER PASTE

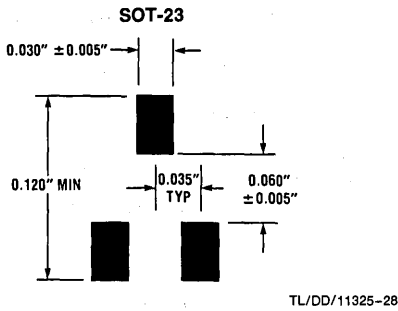
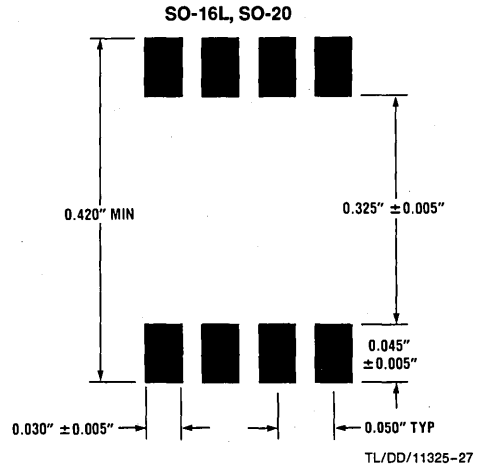
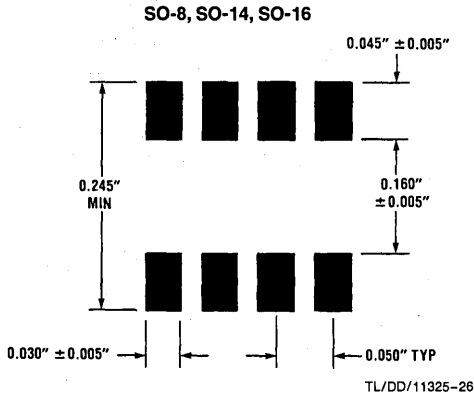
Selection of solder paste tends to be confusing, due to numerous formulations available from various manufacturers. In general, the following guidelines are sufficient to qualify a particular paste for production:

- Particle sizes (see photographs below). Mesh 325 (approximately 45 microns) should be used for general purposes, while larger (solder globules) particles are preferred for leadless components (LCC). The larger particles can easily be used for SO packages.

- Uniform particle distribution. Solder globules should be spherical in shape with uniform diameters and minimum amount of elongation (visual under 100/200 × magnification). Uneven distribution causes uneven melting and subsequent expulsion of smaller solder balls away from their proper sites.

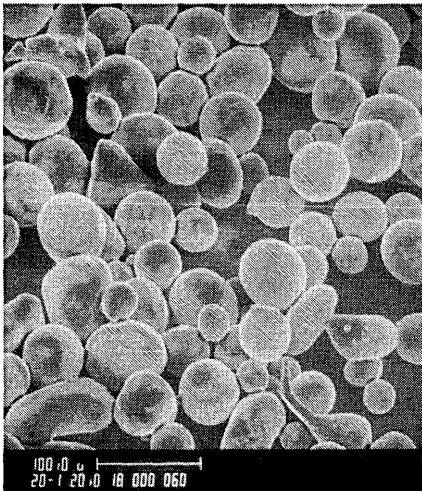
- Composition, generally 60/40 or 63/37 Sn/Pb. Use 62/36 Sn/Pb with 2% Ag in the presence of Au on the soldering area. This formulation reduces problems of metal leaching from soldering pads.
- RMA flux system usually used.
- Use paste with approximately 88–90% solids.

RECOMMENDED SOLDER PADS FOR SO PACKAGES

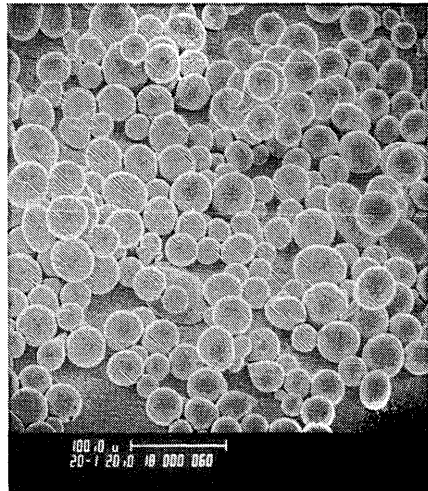


Comparison of Particle Size/Shape of Various Solder Pastes

200 × Alpha (62/36/2)



200 × Kester (63/37)

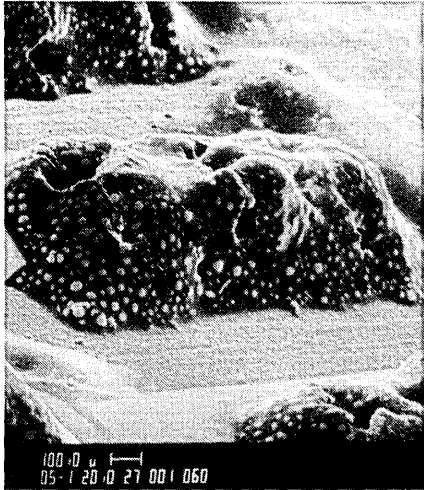


TL/DD/11325-29

TL/DD/11325-30

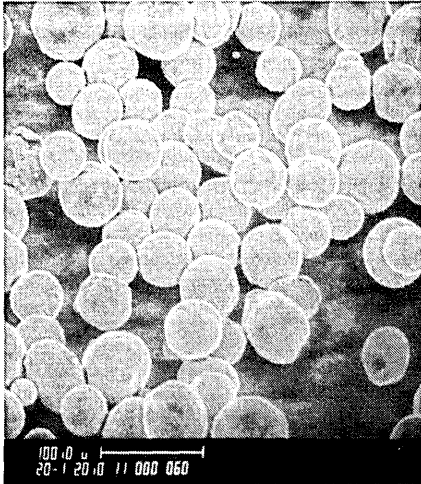
Comparison of Particle Size/Shape of Various Solder Pastes (Continued)

Solder Paste Screen on Pads



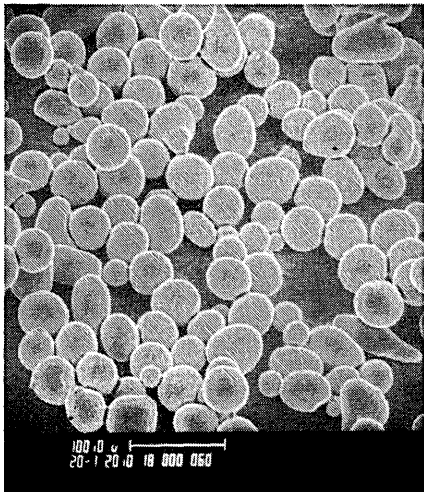
TL/DD/11325-31

200 × Fry Metal (63/37)



TL/DD/11325-32

200 ESL (63/37)



TL/DD/11325-33

CLEANING

The most critical process in surface mounting SO packages is in the cleaning cycle. The package is mounted very close to the surface of the substrate and has a tendency to collect residue left behind after reflow soldering.

Important considerations in cleaning are:

- Time between soldering and cleaning to be as short as possible. Residue should not be allowed to solidify on the substrate for long periods of time, making it difficult to dislodge.
- A low surface tension solvent (high penetration) should be employed. Solvents commercially available are:
 - Freon TMS (general purpose)
 - Freon TE35/TP35 (cold-dip cleaning)
 - Freon TES (general purpose)

It should also be noted that these solvents generally will leave the substrate surface hydrophobic (moisture repellent), which is desirable.

Prelete or 1,1,1-Trichloroethane
Kester 5120/5121

- A defluxer system which allows the workpiece to be subjected to a solvent vapor, followed by a rinse in pure solvent and a high-pressure spray lance are the basic requirements for low-volume production.
- For volume production, a conveyORIZED, multiple hot solvent spray/jet system is recommended.
- Rosin, being a natural occurring material, is not readily soluble in solvents, and has long been a stumbling block to the cleaning process. In recent developments, synthetic flux (SA flux), which is readily soluble in Freon TMS solvent, has been developed. This should be explored where permissible.

The dangers of an inadequate cleaning cycle are:

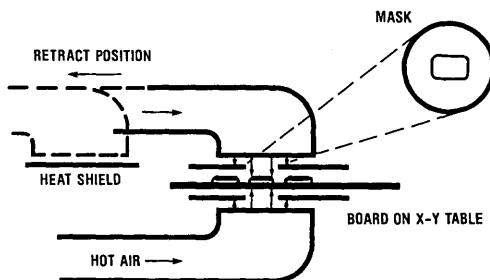
- Ion contamination, where ionic residue left on boards would cause corrosion to metallic components, affecting the performance of the board.
- Electro-migration, where ionic residue and moisture present on electrically-biased boards would cause dendritic growth between close spacing traces on the substrate, resulting in failures (shorts).

REWORK

Should there be a need to replace a component or re-align a previously disturbed component, a hot air system with appropriate orifice masking to protect surrounding components may be used.

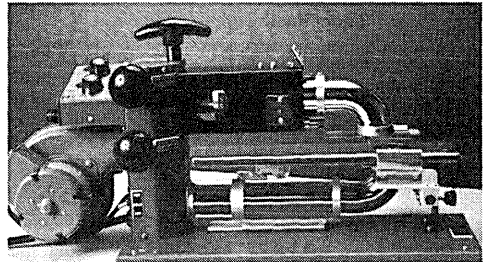
When rework is necessary in the field, specially-designed tweezers that thermally heat the component may be used to remove it from its site. The replacement can be fluxed at the

Hot-Air Solder Rework Station



TL/DD/11325-34

Hot-Air Rework Machine



TL/DD/11325-35

lead tips or, if necessary, solder paste can be dispensed onto the pads using a varimeter. After being placed into position, the solder is reflowed by a hot-air jet or even a standard soldering iron.

WAVE SOLDERING

In a case where lead insertions are made on the same board as surface-mounted components, there is a need to include a wave-soldering operation in the process flow.

Two options are used:

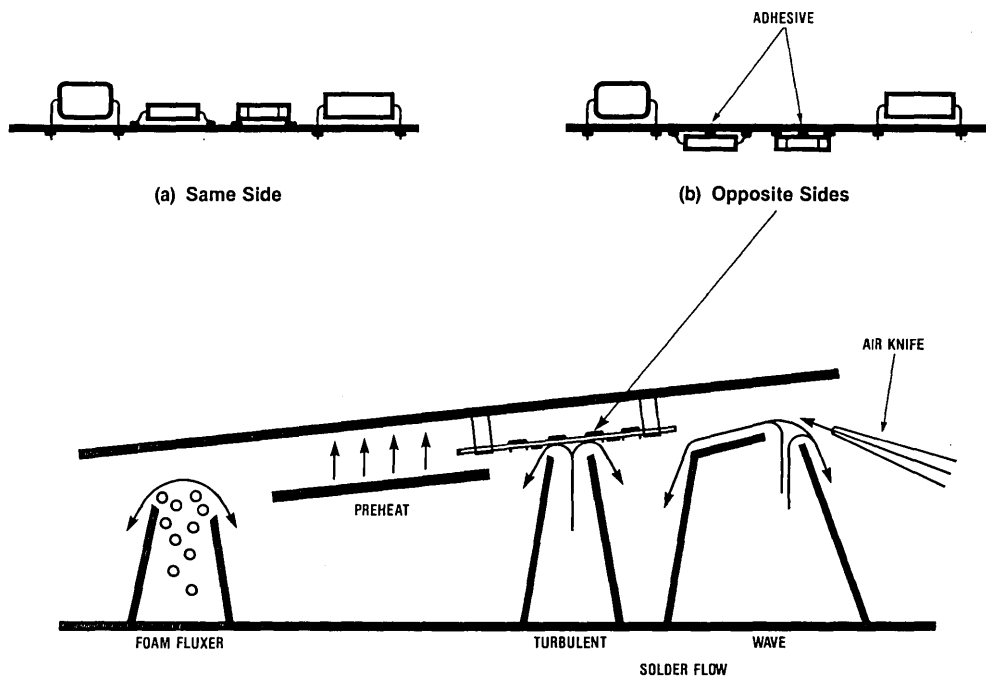
- Surface mounted components are placed and vapor phase reflowed before auto-insertion of remaining components. The board is carried over a standard wave-solder system and the underside of the board (only lead-inserted leads) soldered.
- Surface-mounted components are placed in position, but no solder paste is used. Instead, a drop of adhesive about 5 mils maximum in height with diameter not exceeding 25% width of the package is used to hold down the package. The adhesive is cured and then proceeded to auto-insertion on the reverse side of the board (surface-mounted side facing down). The assembly is then passed over a "dual wave" soldering system. Note that the surface-mounted components are immersed into the molten solder.

Lead trimming will pose a problem after soldering in the latter case, unless the leads of the insertion components are pre-trimmed or the board specially designed to localize certain areas for easy access to the trim blade.

The controls required for wave soldering are:

- Solder temperature to be 240–260°C. The dwell time of components under molten solder to be short (preferably kept under 2 seconds), to prevent damage to most components and semiconductor devices.
- RMA (Rosin Mildly Activated) flux or more aggressive OA (Organic Acid) flux are applied by either dipping or foam fluxing on boards prior to preheat and soldering. Cleaning procedures are also more difficult (aqueous, when OA flux is used), as the entire board has been treated by flux (unlike solder paste, which is more or less localized). Non-halide OA fluxes are highly recommended.
- Preheating of boards is essential to reduce thermal shock on components. Board should reach a temperature of about 100°C just before entering the solder wave.
- Due to the closer lead spacings (0.050" vs 0.100" for dual-in-line packages), bridging of traces by solder could occur. The reduced clearance between packages also causes "shadowing" of some areas, resulting in poor solder coverage. This is minimized by dual-wave solder systems.

Mixed Surface Mount and Lead Insertion



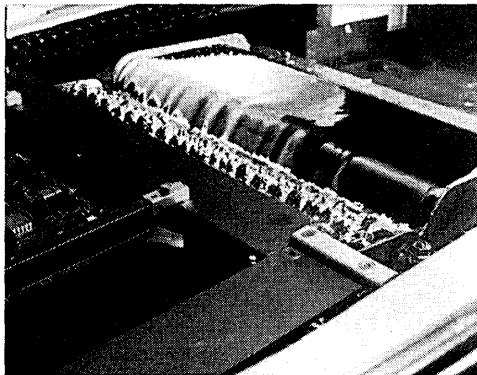
TL/DD/11325-36

A typical dual-wave system is illustrated below, showing the various stages employed. The first wave typically is in turbulence and given a transverse motion (across the motion of the board). This covers areas where "shadowing" occurs. A second wave (usually a broad wave) then proceeds to perform the standard soldering. The departing edge from the solder is such to reduce "icicles," and is still further reduced by an air knife placed close to the final soldering step. This air knife will blow off excess solder (still in the fluid stage) which would otherwise cause shorts (bridging) and solder bumps.

AQUEOUS CLEANING

- For volume production, a conveyerized system is often used with a heated recirculating spray wash (water temperature 130°C), a final spray rinse (water temperature 45–55°C), and a hot (120°C) air/air-knife drying section.
- For low-volume production, the above cleaning can be done manually, using several water rinses/tanks. Fast-drying solvents, like alcohols that are miscible with water, are sometimes used to help the drying process.
- Neutralizing agents which will react with the corrosive materials in the flux and produce material readily soluble in water may be used; the choice depends on the type of flux used.
- Final rinse water should be free from chemicals which are introduced to maintain the biological purity of the water. These materials, mostly chlorides, are detrimental to the assemblies cleaned because they introduce a fresh amount of ionizable material.

Dual Wave



TL/DD/11325-37

CONFORMAL COATING

Conformal coating is recommended for high-reliability PCBs to provide insulation resistance, as well as protection against contamination and degradation by moisture.

Requirements:

- Complete coating over components and solder joints.
- Thixotropic material which will not flow under the packages or fill voids, otherwise will introduce stress on solder joints on expansion.
- Compatibility and possess excellent adhesion with PCB material/components.
- Silicones are recommended where permissible in application.

SMD Lab Support

FUNCTIONS

Demonstration—Introduce first-time users to surface-mounting processes.

Service—Investigate problems experienced by users on surface mounting.

Reliability Builds—Assemble surface-mounted units for reliability data acquisition.

Techniques—Develop techniques for handling different materials and processes in surface mounting.

Equipment—In conjunction with equipment manufacturers, develop customized equipments to handle high density, new technology packages developed by National.

In-House Expertise—Availability of in-house expertise on semiconductor research/development to assist users on packaging queries.

Plastic Leaded Chip Carrier (PLCC) Packaging

General Description

The Plastic Leaded Chip Carrier (PLCC) is a miniaturized low cost semiconductor package designed to replace the Plastic Dual-In-Line Package (P-DIP) in high density applications. The PLCC utilizes a smaller lead-to-lead spacing—0.050" versus 0.100" - and leads on all four sides to achieve a significant footprint reduction over the P-DIP. The rolled under J-bend leadform separates this package style from other plastic quad packages with flat or gull wing lead forms. As with virtually all packages of 0.050" or less lead spacing, the PLCC requires surface mounting to printed circuit boards as opposed to the more conventional thru-hole mounting of the P-DIP.

History

The Plastic Leaded Chip Carrier with J-bend leadform was first introduced in 1976 as a premolded plastic package. The premolded version has yet to become popular but the quad format with J-Bend leads has been adapted to traditional post molded packaging technology (the same technology used to manufacture the P-DIP). In 1980 National Semiconductor developed a post molded version of the PLCC. The J-bend leadform allowed them to adopt the footprint connection pattern already registered with JEDEC for the leadless chip carrier (LCC). In 1981 a task force was organized within JEDEC to develop a PLCC registration for package I/O counts of 20, 28, 44, 52, 68, 84, 100, and 124. A registered outline was completed in 1984 (JEDEC Outline MO-047) after many changes and improvements over the original proposals. This first PLCC registration covers square packages with an equal number of leads on all sides. A second registration, MO-052, was completed in 1985 for rectangular packages with I/O counts of 18, 22, 28 and 32. Since 1980 many additional semiconductor manufacturers and packaging subcontractors have developed PLCC capability. There are now well over 20 sources with the number growing steadily.

Surface Mounting

Surface mounting refers to component attachment whereby the component leads or pads rest on the surface of the PCB instead of the traditional approach of inserting the leads into through-holes which go through the board. With surface mounting there are solder pads on the PCB which align with the leads or pads on the component. The resulting solder joint forms both the mechanical and electrical connection.

ADVANTAGES

The primary reason for surface mounting is to allow leads to be placed closer together than the 0.100" standard for DIPs with through-hole mounting. Through-hole mounting on smaller than 0.100" spacing is difficult to achieve in production and generally avoided. The move to 0.050" lead spacing offered with the current generation of surface mounted components, along with a switch from a dual-in-line format to a quad format, has achieved a threefold increase in component mounting density. A need to achieve greater density is a major driving force in today's marketplace.

MANUFACTURING TECHNIQUES

Learning how to surface mount components to printed circuit boards requires the user to become educated in new assembly processes not typically associated with through-hole insertion/wave soldering assembly methods.

Surface mounting involves three basic process steps:

- 1) Application of solder or solder paste to the printed circuit board.
- 2) Positioning of the component onto the printed circuit board
- 3) Reflowing of the solder or solder paste.

As with any process, there are many details involved to achieve acceptable throughput and acceptable quality. National Semiconductor offers a surface mounting guide which deals with the specifics of successful surface mounting. We encourage the user to review this document and to contact us if further information on surface mounting is desired.

Benefits of the PLCC

There are four principle advantages offered the user by switching from P-DIP to PLCC. These four advantages are outlined below as follows:

1. Increased Density—
 - Typically 3-to-1 size reduction of printed circuit boards. See *Figure 1* for a footprint comparison between PLCC and P-DIP. This can be as high as 6-to-1 in certain applications.
 - Surface mounting allows components to be placed on both sides of the board.
 - Surface mount and thru-hole mount components can be placed on the same board.
 - The large diameter thru-holes can be reduced in number, entirely eliminated, or reduced in size (if needed for via connection).
2. Increased Performance—
 - Shorter traces on printed circuit boards.
 - Better high frequency operation.
 - Shorter leads in package. *Figure 2* and Table I compare PLCC and P-DIP mechanical and electrical characteristics.
3. Increased Reliability—
 - Leads are well protected.
 - Fewer connectors.
 - Simplified rework.
 - Vibration and shock resistant.
4. Reduced Cost—
 - Fewer or smaller printed circuit boards.
 - Less hardware.
 - Same low cost printed circuit board material.
 - Plastic packaging material.
 - Reduced number of costly plated-through-holes.
 - Fewer circuit layers.

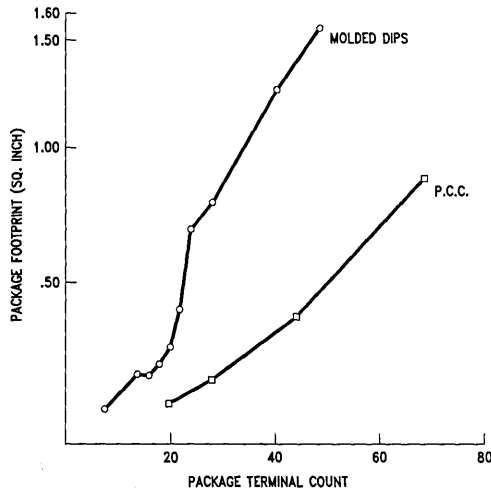


FIGURE 1. Footprint Area of PLCC vs. P-DIP

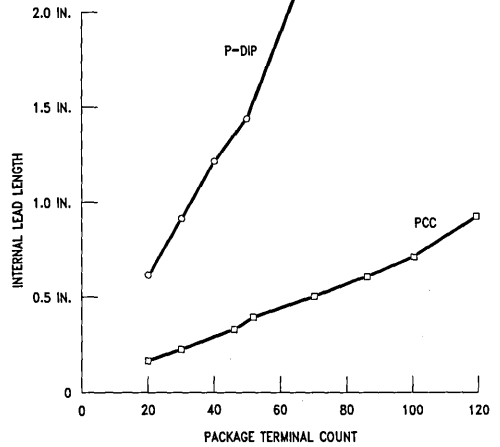


FIGURE 2. Longest Internal Lead PLCC vs. P-DIP

TABLE I. Electrical Performance of PLCC vs. P-DIP (44 I/O PLCC vs. 40 I/O P-DIP, both with Copper Leads)

Criteria	Shortest Lead		Longest Lead	
	PLCC	P-DIP	PLCC	P-DIP
Lead Resistance (Measured)	3Ω	4Ω	6Ω	7Ω
Lead-to-Lead Capacitance (Measured on Adjacent Leads)	0.1 pF	0.1 pF	0.3 pF	3.0 pF
Lead Self-Inductance (Calculated)	3.2 nH	1.4 nH	3.5 nH	19.1 nH

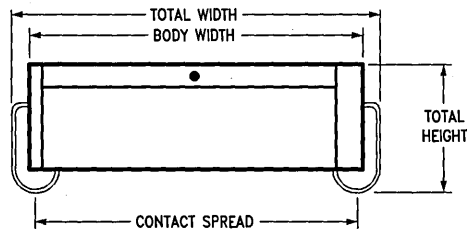


FIGURE 3. Package Outline

TABLE II. Principle Dimensions Inches/(Millimeters) (Refer to Figure 3)

Lead Count	Total Width		Total Height		Body Width		Contact Spread	
	Min	Max	Min	Max	Min	Max	Min	Max
20	0.385 sq. (9.779)	0.395 sq. (10.03)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.345 sq. (8.763)	0.355 sq. (9.017)	0.310 sq. (7.874)	0.330 sq. (8.382)
28	0.485 sq. (12.32)	0.495 sq. (12.57)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.445 sq. (11.30)	0.455 sq. (11.56)	0.410 sq. (10.41)	0.430 sq. (10.92)
44	0.685 sq. (17.40)	0.695 sq. (17.65)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.645 sq. (16.38)	0.655 sq. (16.64)	0.610 sq. (15.49)	0.630 sq. (16.00)

TABLE II. Principle Dimensions Inches/(Millimeters) (Refer to Figure 3) (Continued)

Lead Count	Total Width		Total Height		Body Width		Contact Spread	
	Min	Max	Min	Max	Min	Max	Min	Max
68	0.985 sq. (25.02)	0.995 sq. (25.27)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.945 sq. (24.00)	0.955 sq. (24.26)	0.910 sq. (23.11)	0.930 sq. (23.62)
84	1.185 sq. (30.10)	1.195 sq. (30.36)	0.165 sq. (4.191)	0.180 sq. (4.572)	1.150 sq. (29.21)	1.158 sq. (29.41)	1.110 sq. (28.20)	1.130 sq. (28.70)
124	1.685 sq. (49.13)	1.695 sq. (49.39)	0.180 sq. (4.572)	0.200 sq. (5.080)	1.650 sq. (41.91)	1.658 sq. (42.11)	1.610 sq. (40.90)	1.630 sq. (41.40)

TABLE III. Package Thermal Resistance (Deg. C/Watt, Junction-to-Ambient, Board Mount)

Lead Count	Device Size		
	1,000 Mil ²	10,000 Mil ²	100,000 Mil ²
20	102	85	67
28	95	73	55
44	54	47	40
68	44	40	38
84*	40	35	30
124*	40	35	30

*Estimated values

Package Design Criteria

Experience has taught us there are certain criteria to the PLCC design which must be followed to provide the user with the proper mechanical and thermal performance. These requirements should be carefully reviewed by the user when selecting suppliers for devices in PLCC. Some of these are covered by the JEDEC registration and some are not. These important requirements are listed in Table IV.

Reliability

National Semiconductor utilizes an assembly process for the PLCC which is similar to our P-DIP assembly process. We also utilize identical materials. This is a very important point when considering reliability. Many years of research

and development have gone into steadily improving our P-DIP quality and maintaining a leadership position in plastic package reliability. All of this technology can be directly applied to the PLCC. Table V shows the results of applying this technology to the PLCC. As we make further advances in plastic package reliability, these will also be applied to the PLCC.

Sockets

There are several manufacturers currently offering sockets for the plastic chip carrier. Following is a listing of those manufacturers. The listing is divided into test/burn-in and production categories. There may be some individual sockets that will cover both requirements.

TABLE IV. Package Design Criteria

Criteria	Required to Comply with JEDEC Registration
Minimum Inside Bend Radius of Lead at Shoulder Equal or Greater than Lead Thickness—to Prevent Lead Cracking/Fatigue	Not Required
Minimum One Mil Clearance Between Lead and Plastic Body at all Points—to Provide Lead Compliancy and Prevent Shoulder Joint Cracking/Fatigue	Not Required
Copper Leads for Low Thermal Resistance	Not Required
Minimum 10 Mil Lead Thickness for Low Thermal Resistance and Good Handling Properties	Not Required
Minimum 26 Mil Lead Shoulder Width to Prevent Interlocking of Devices During Handling	Yes
Maximum 4 Mils coplanarity Across Seating Plane of all Leads	Yes

TABLE V. Reliability Test Data
(Expressed as Failures per Units Tested)

Device/Package	OPL	TMCL	TMSK	BHTL	ACLV
LM324/20 Lead	0/96	0/199	0/50	0/97	0/300
LF353/20 Lead	0/50	0/50	—	0/45	0/100
DS75451/20 Lead	0/47	—	0/50	0/93	0/179
DM875191/28 Lead	0/154	0/154	0/154	0/154	0/154
DM875181/28 Lead	0/77	0/77	0/77	0/77	0/77

OPL = Dynamic high temperature operating life at 125°C or 150°C, 1,000 hours.

TMCL = Temperature cycle, Air-to-Air, -40°C to +125°C or -65°C to +150°C, 2,000 cycles.

TMSK = Thermal shock, Liquid-to-Liquid, -65°C to +150°C, 100 cycles.

BHTL = Biased humidity temperature life, 85°C, 85% humidity, 1,000 hours.

ACLV = Autoclave, 15 psi, 121°C, 100% humidity, 1,000 hours.

Production Sockets

AMP

Harrisburg, PA
(715) 564-0100

Augat

Attleboro, MA
(617) 222-2202

Burndy

Norwalk, CT
(203) 838-4444

Methode

Rolling Meadows, IL
(312) 392-3500

Textool

Irving, TX
(214) 259-2676

Thomas & Betts

Raritan, NJ
(201) 469-4000

Test/Burn-In Sockets

Plastronics

Irving, TX
(214) 258-1906

Textool

Irving, TX
(214) 259-2676

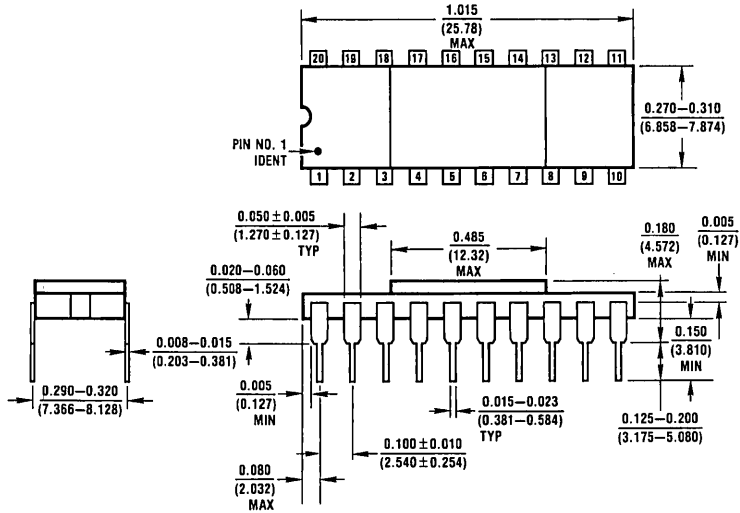
Yamaichi

c/o Nepenthe Dist.
(415) 856-9332

ADDITIONAL INFORMATION AND SERVICES

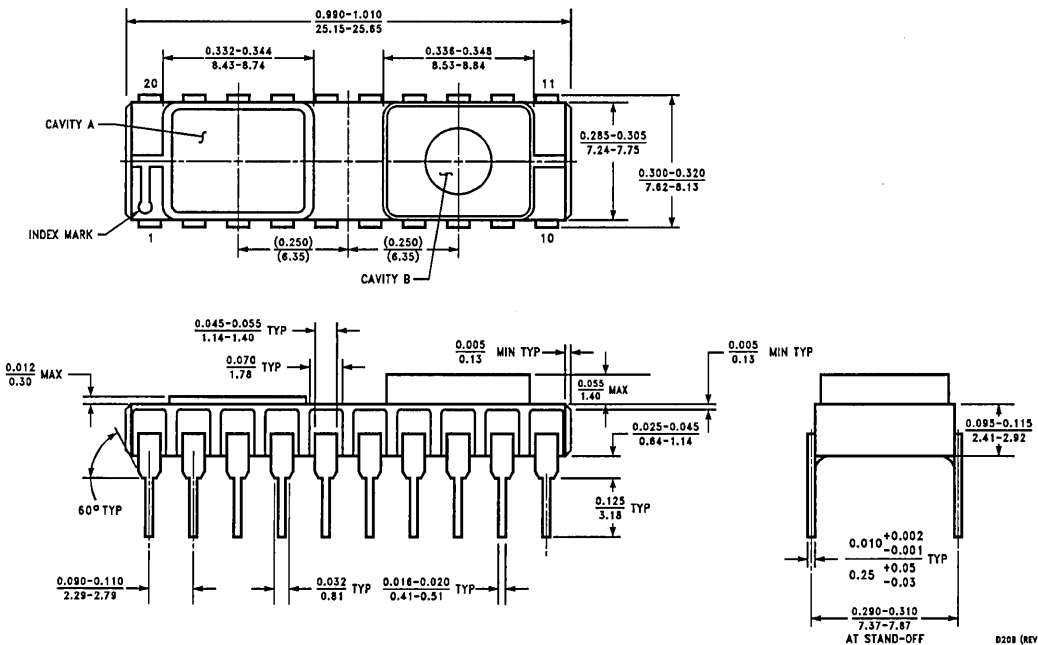
National Semiconductor offers additional Databooks which cover surface mount technology in much greater detail. We also have a surface mount laboratory to provide demonstrations and customer support, as well as technology development. Feel free to contact us about these additional resources.

20 Lead Hermetic Dual-In-Line Package (D) NS Package Number D20A



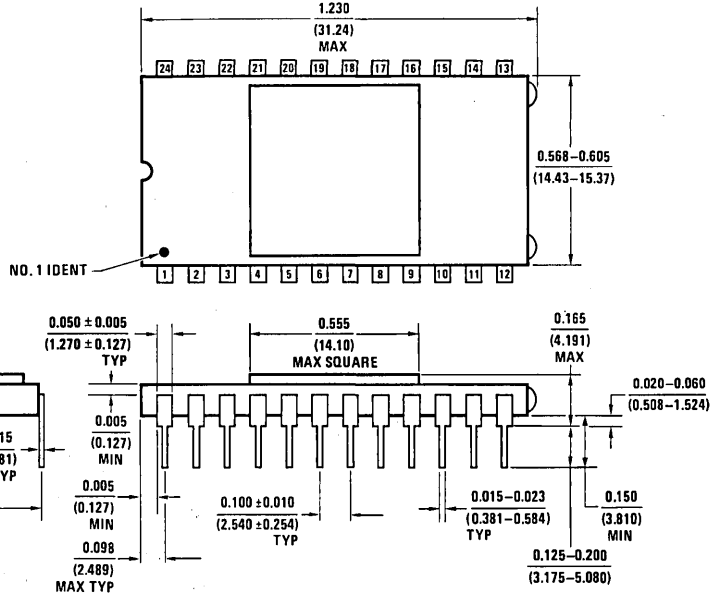
D20A (REV D)

20 Lead Hermetic Dual-In-Line Package (D) NS Package Number D20B



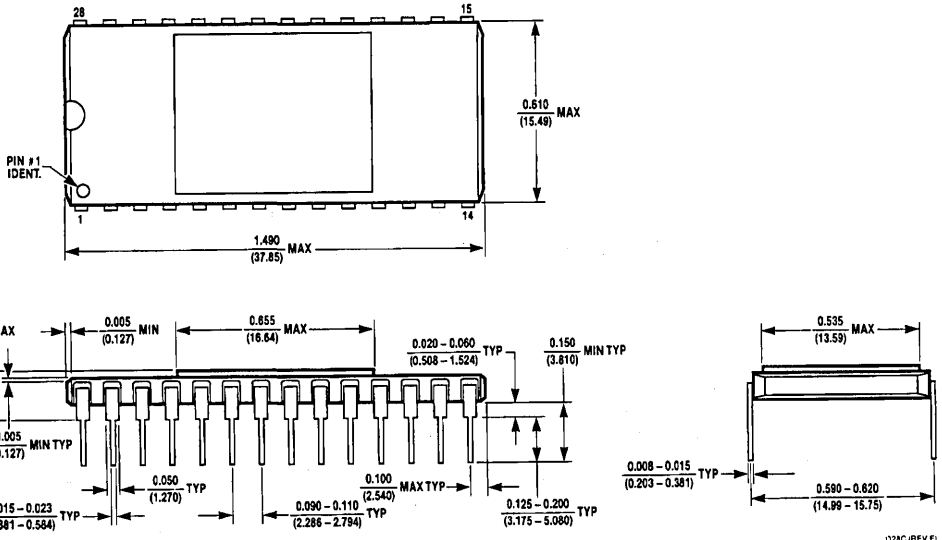
D20B (REV H)

24 Lead Hermetic Dual-In-Line Package (D) NS Package Number D24C



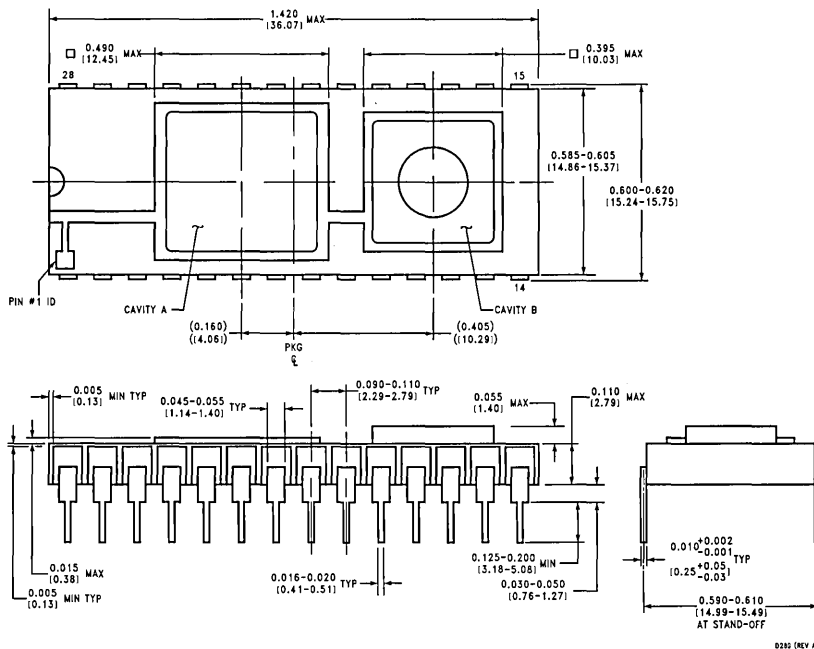
D24C (REV G)

28 Lead Hermetic Dual-In-Line Package (D) NS Package Number D28C

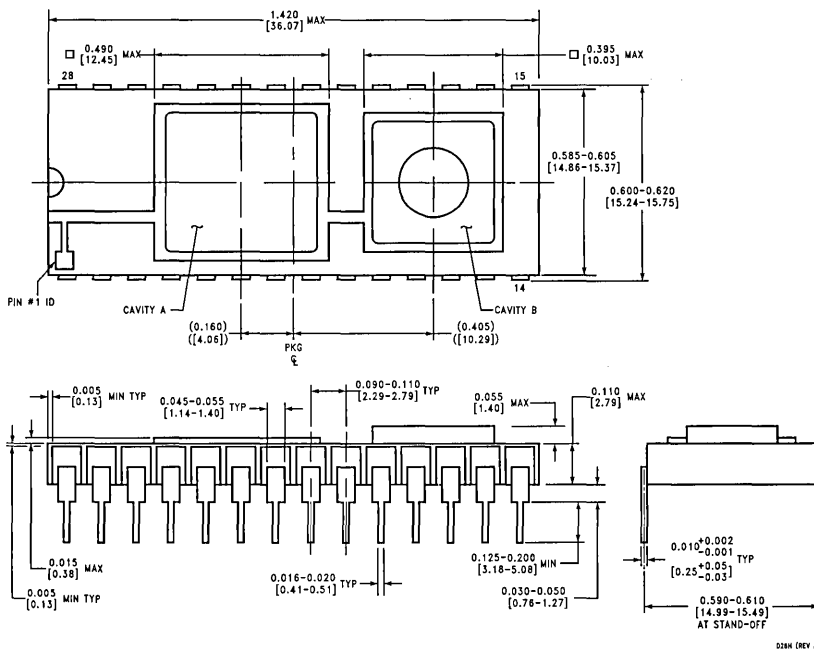


D28C (REV F)

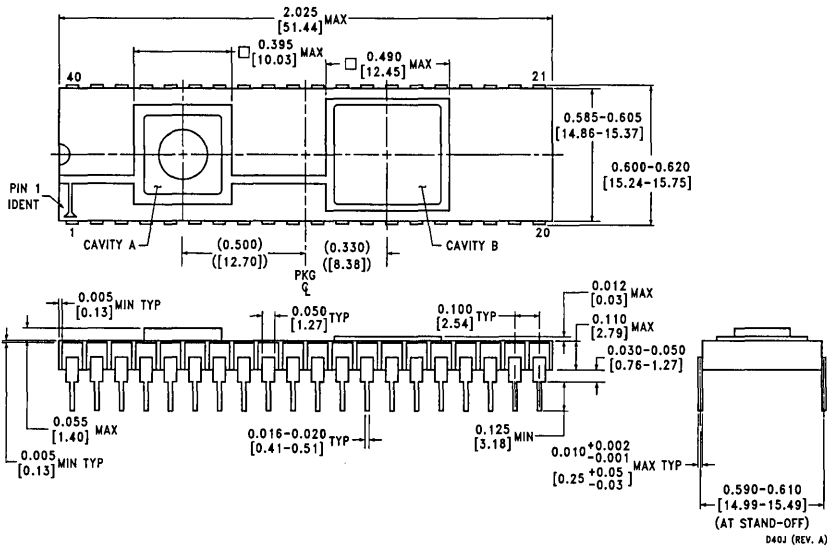
28 Lead Sidebraze Hermetic Dual-In-Line Package (D) NS Package Number D28G



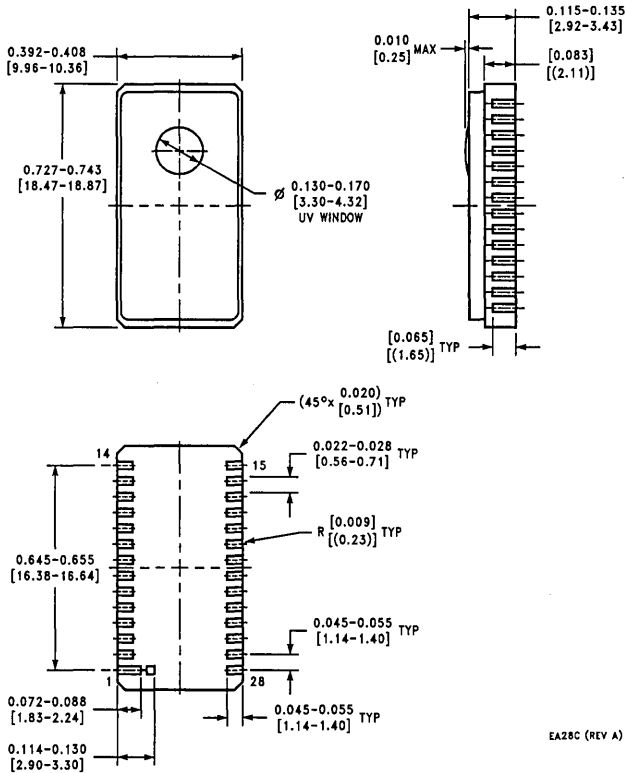
28 Lead Sidebraze Hermetic Dual-In-Line Package (D) NS Package Number D28H



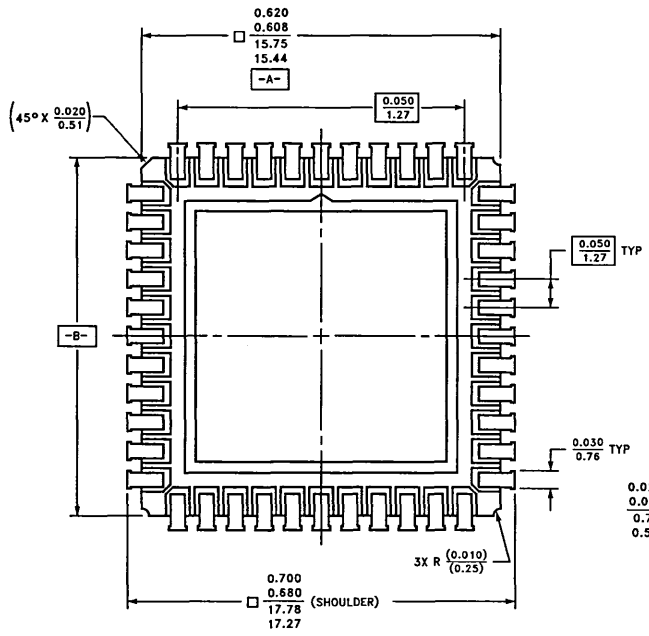
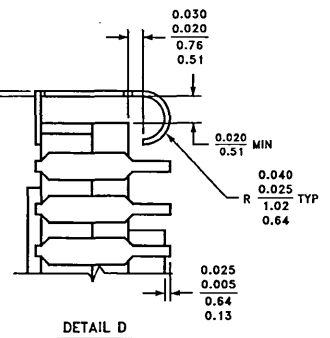
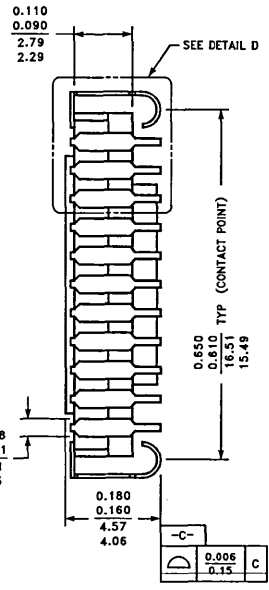
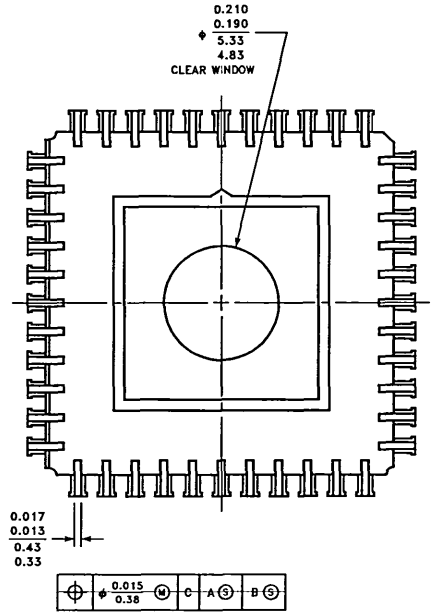
40 Lead Hermetic Dual-In-Line Package (D) NS Package Number D40J



28 Terminal Leadless Chip Carrier (E) NS Package Number EA28C

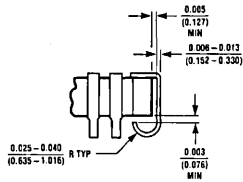
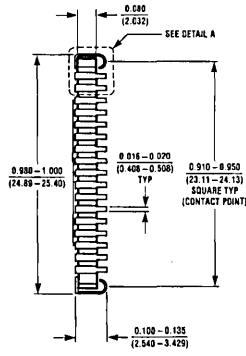
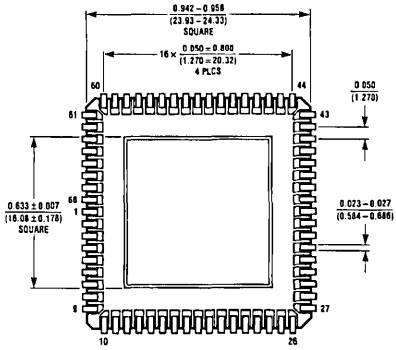


44 Lead Chip Carrier (E)
NS Package Number EL44B



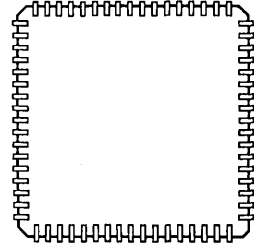
EL44B (REV.A)

68 Lead Chip Carrier (E) NS Package Number EL68A

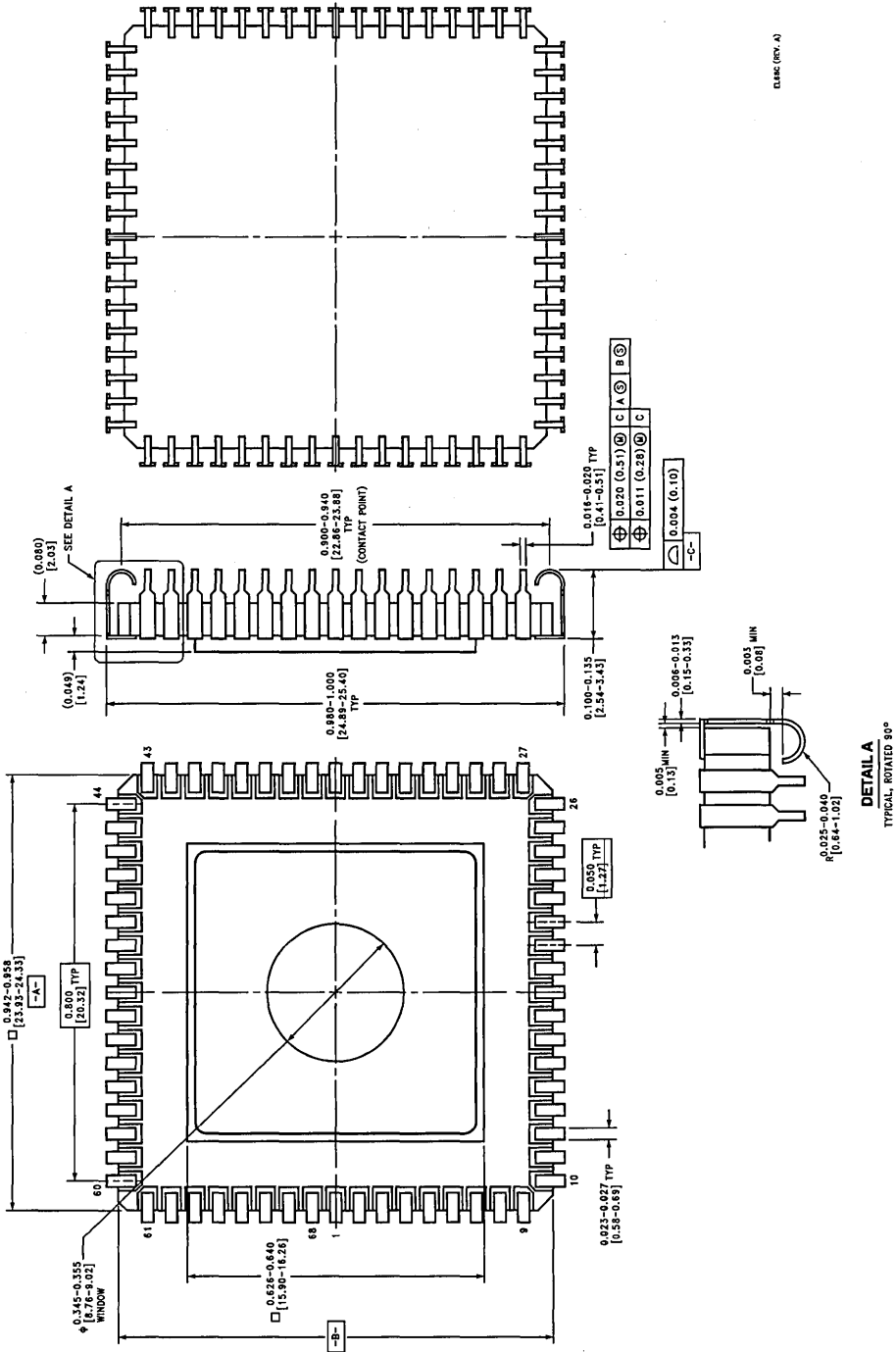


DETAIL A
SCALE: 10x

D144-REV B

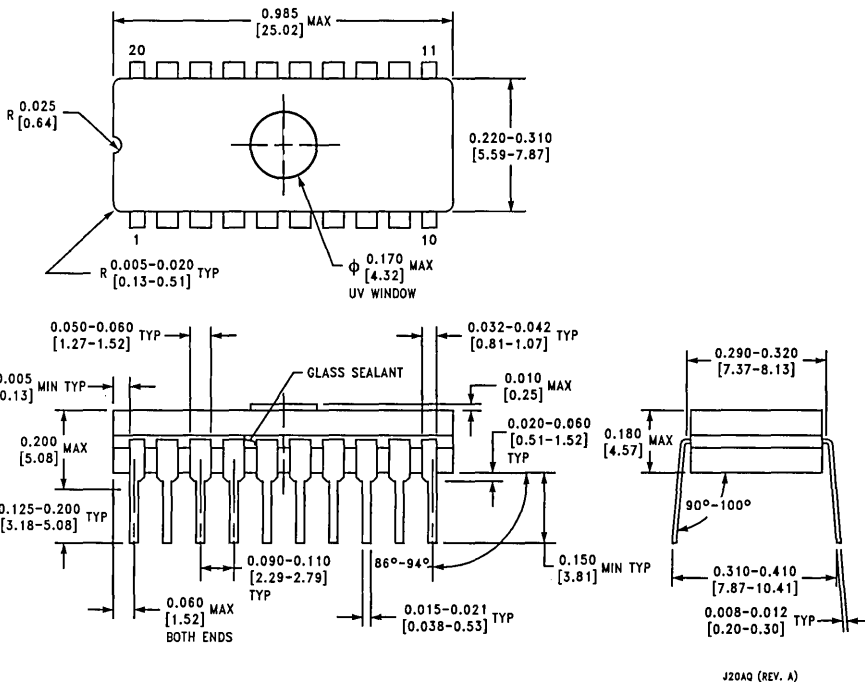


68 Lead Chip Carrier (E) NS Package Number EL68C

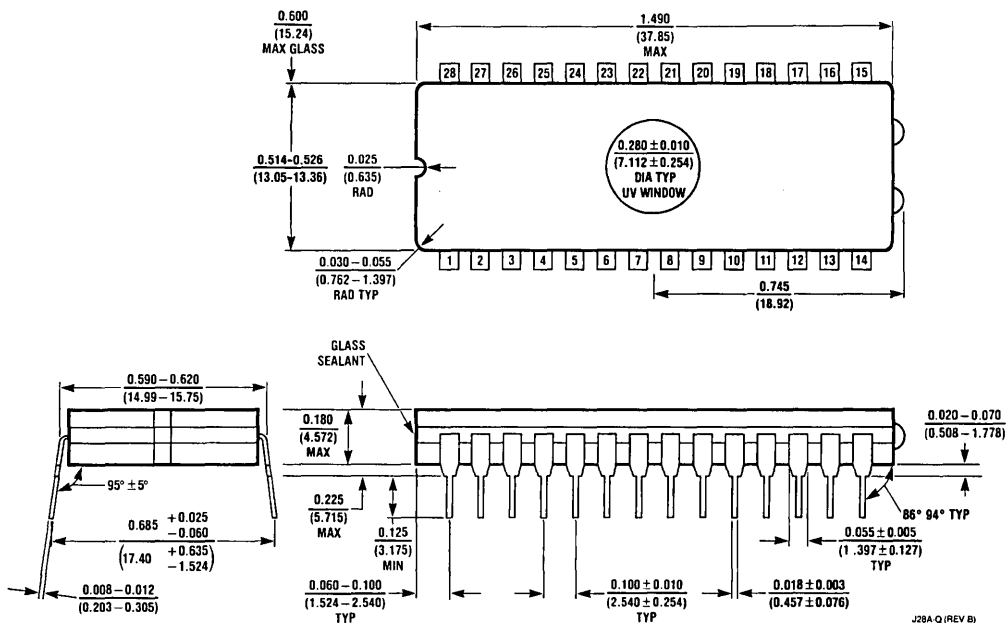


EL68C (REV. A)

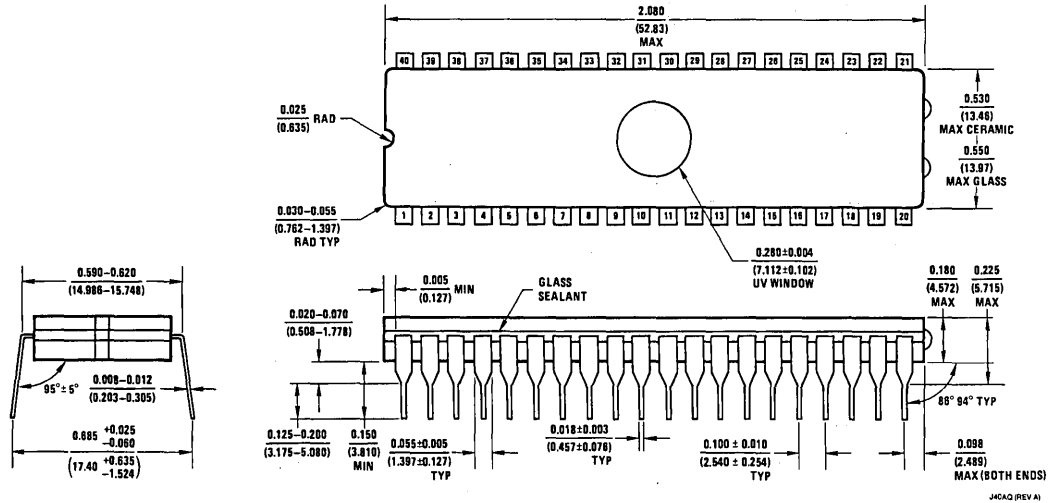
20 Lead Ceramic Dual-In-Line Package (J) NS Package Number J20AQ



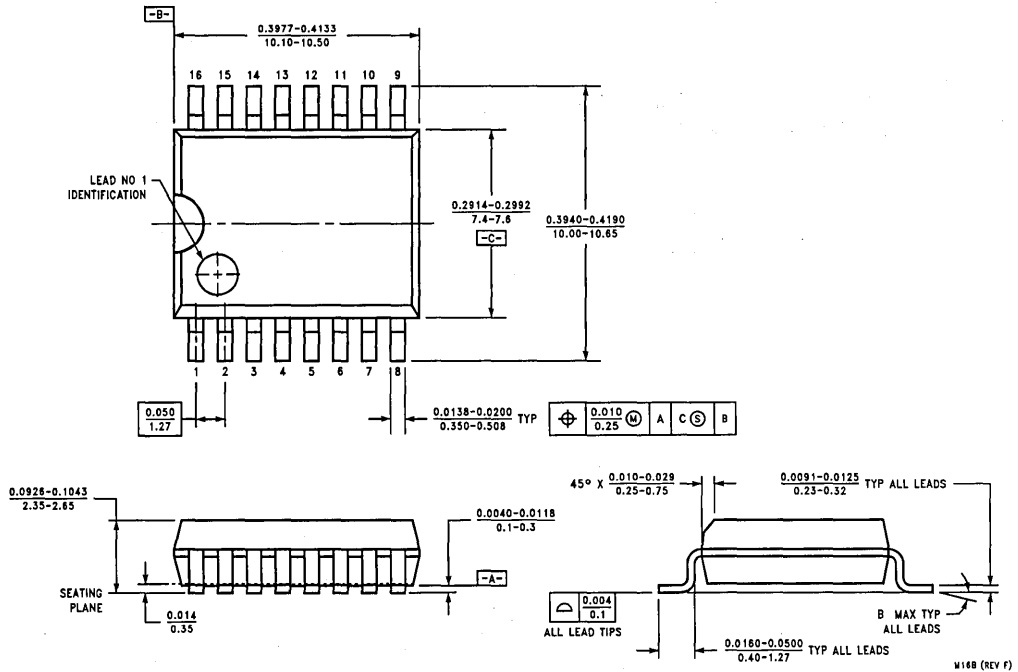
28 Lead Ceramic Dual-In-Line Package (J) NS Package Number J28AQ



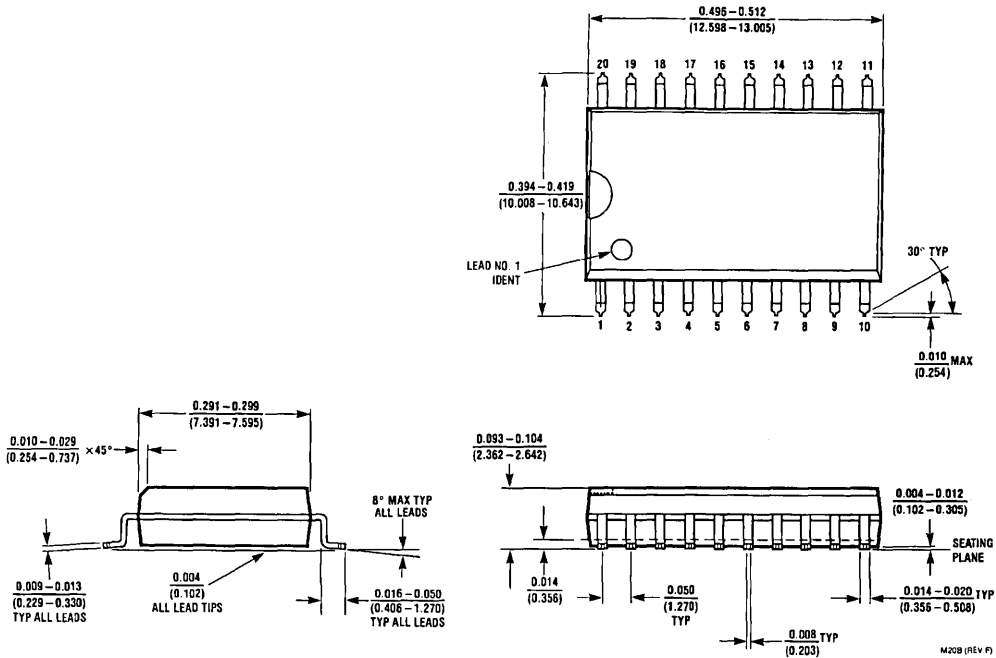
40 Lead Ceramic Dual-In-Line Package (J) NS Package Number J40AQ



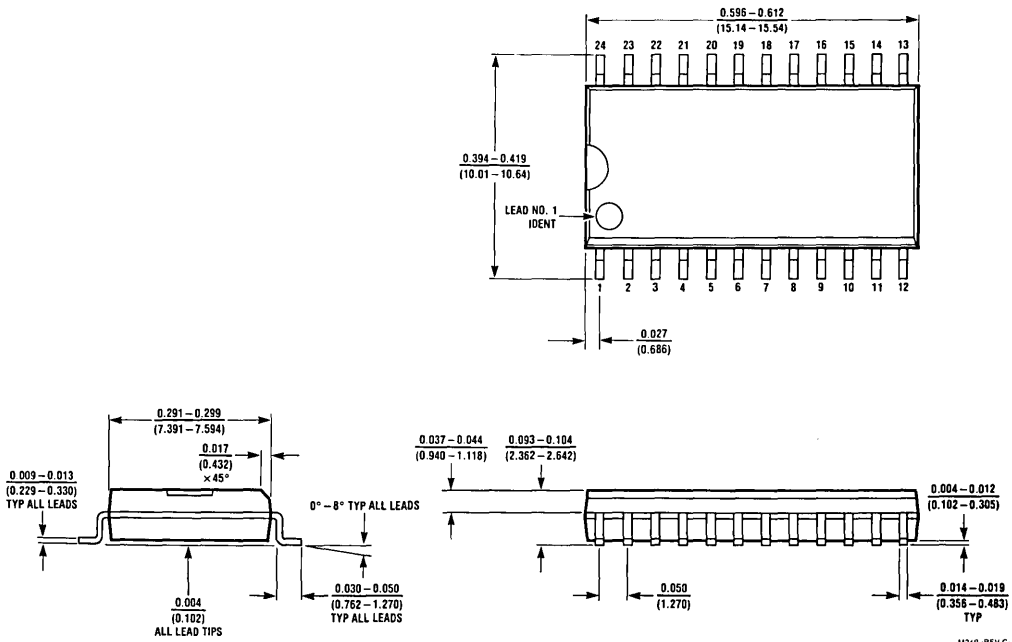
16 Lead (0.300" Wide) Molded Small Outline Package (M) NS Package Number M16B



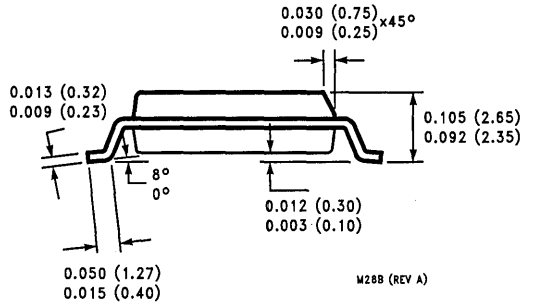
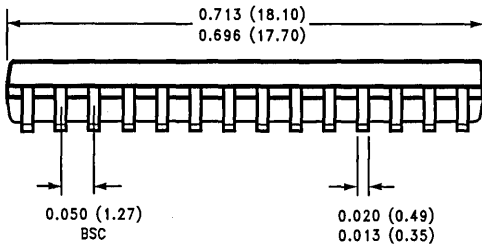
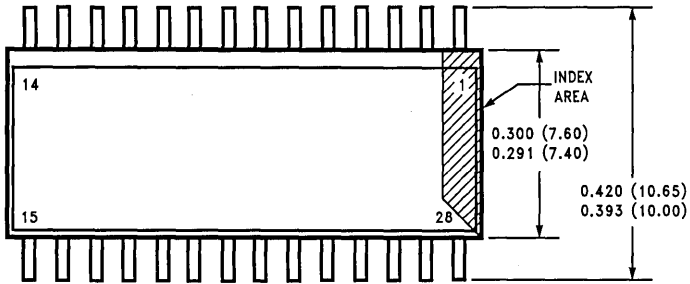
20 Lead (0.300" Wide) Molded Small Outline Package (M) NS Package Number M20B



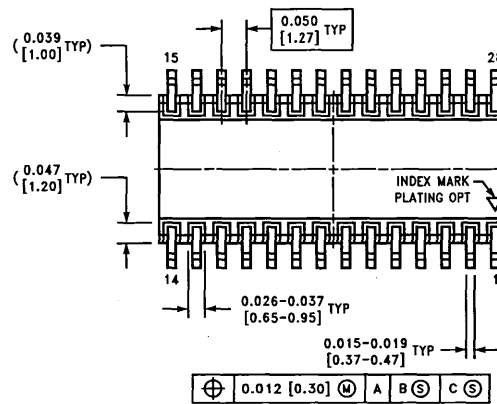
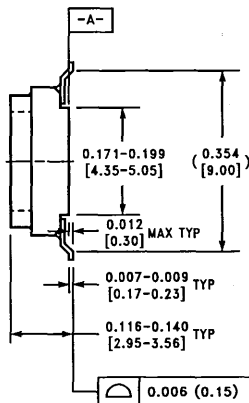
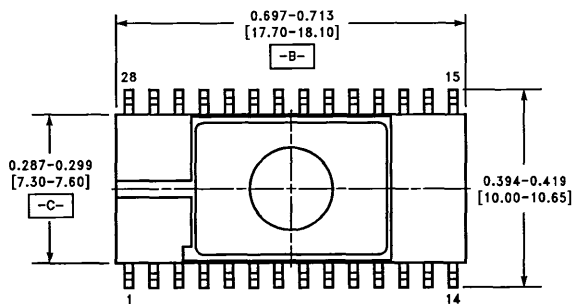
24 Lead (0.300" Wide) Molded Small Outline Package (M) NS Package Number M24B



28 Lead (0.300" Wide) Molded Small Outline Package (M) NS Package Number M28B

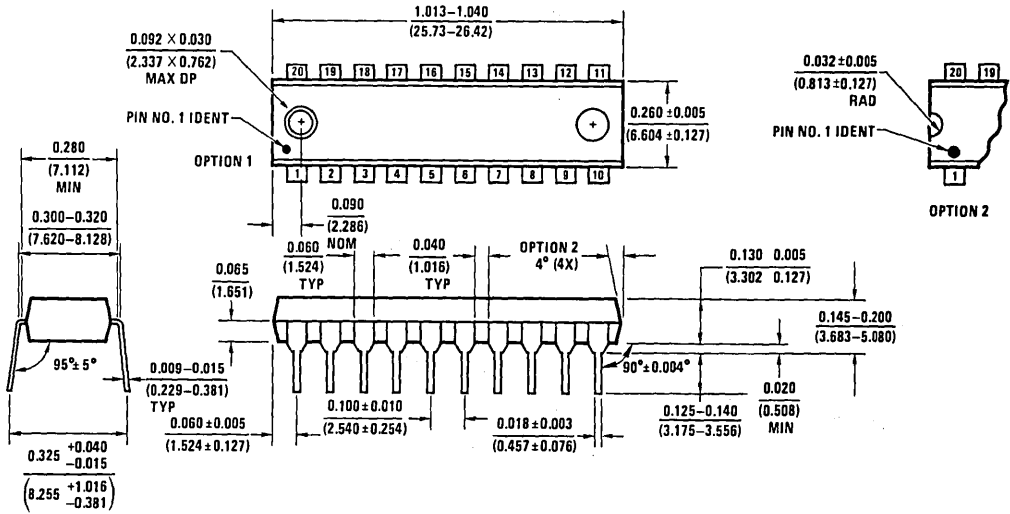


28 Lead (0.300" Wide) Molded Small Outline Package (M)
NS Package Number MC28B



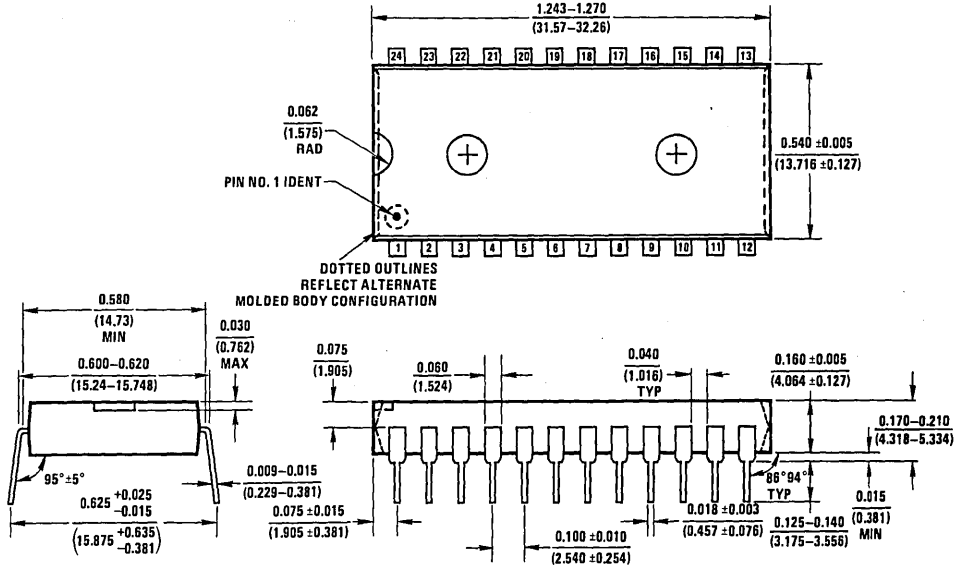
MC28B (REV A)

20 Lead Molded Dual-In-Line Package (N) NS Package Number N20A



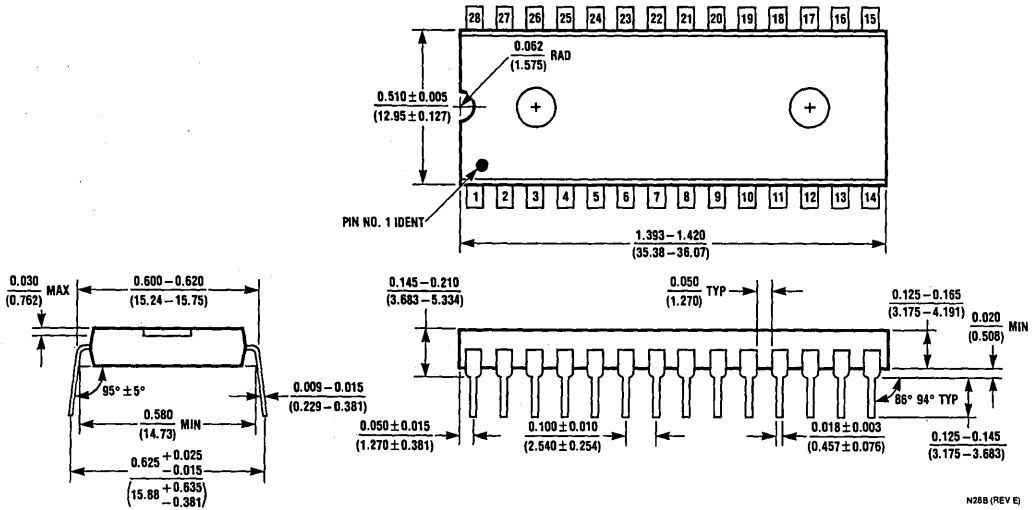
N20A (REV G)

24 Lead Molded Dual-In-Line Package (N) NS Package Number N24A

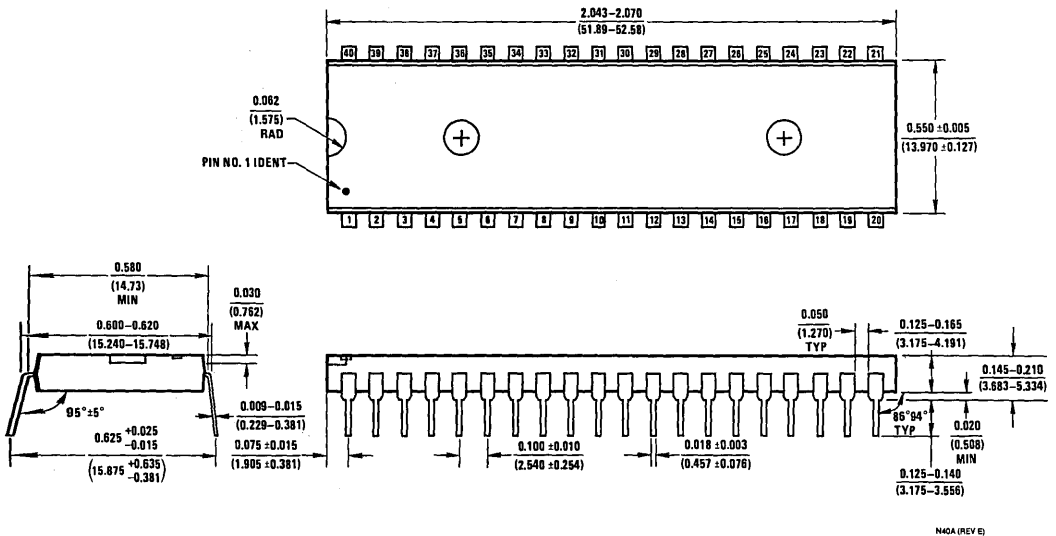


N24A (REV E)

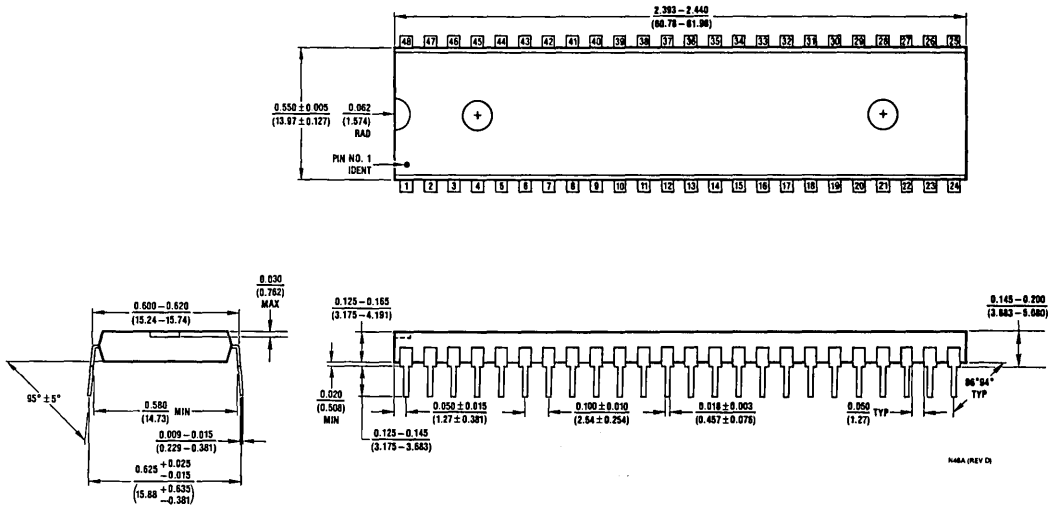
28 Lead Molded Dual-In-Line Package (N) NS Package Number N28B



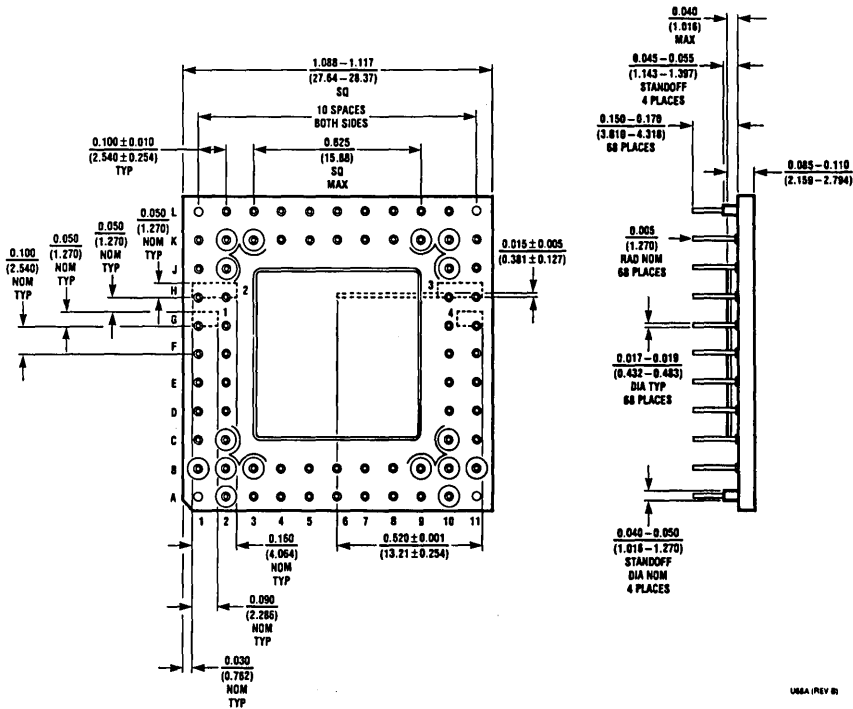
40 Lead Molded Dual-In-Line Package (N) NS Package Number N40A



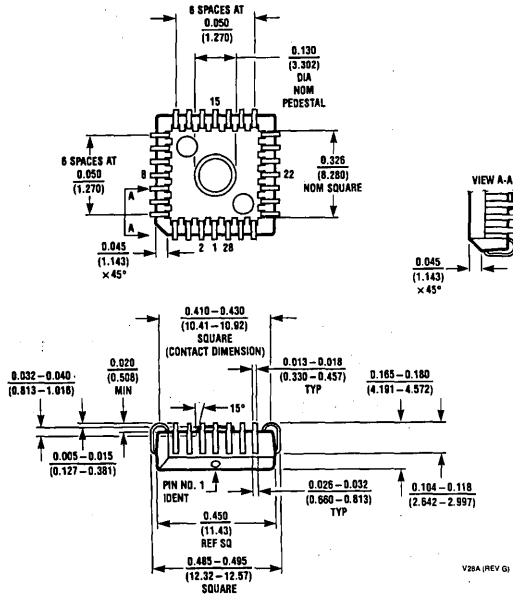
48 Lead Molded Dual-In-Line Package (N) NS Package Number N48A



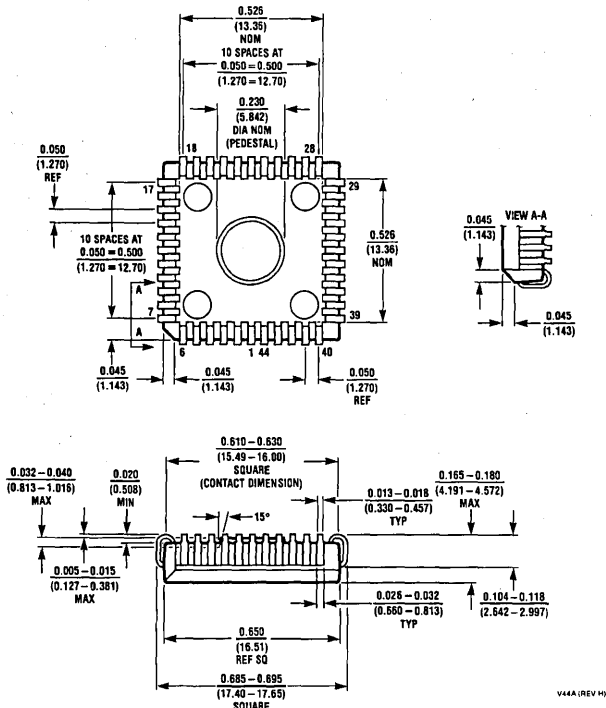
68 Lead Ceramic Pin Grid Array (U) NS Package Number U68A



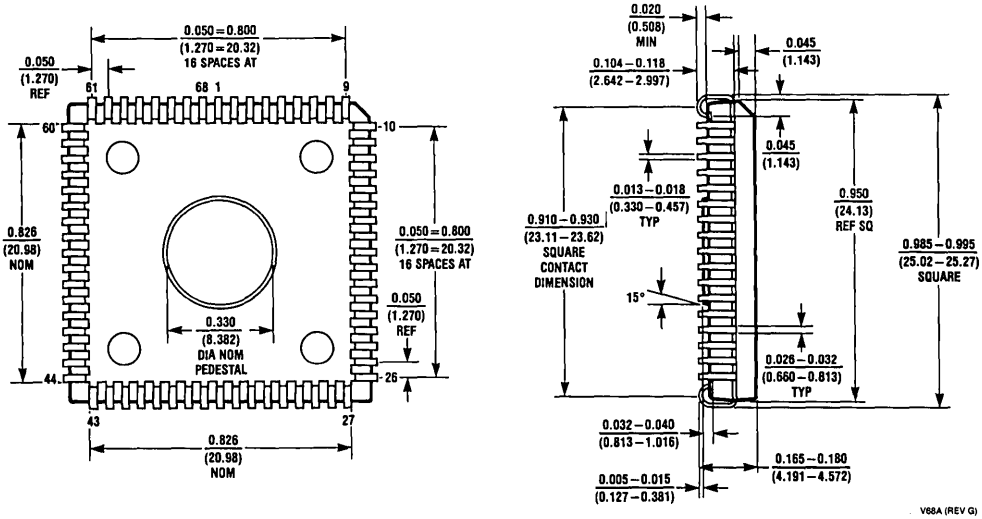
28 Lead Plastic Chip Carrier (V) NS Package Number V28A



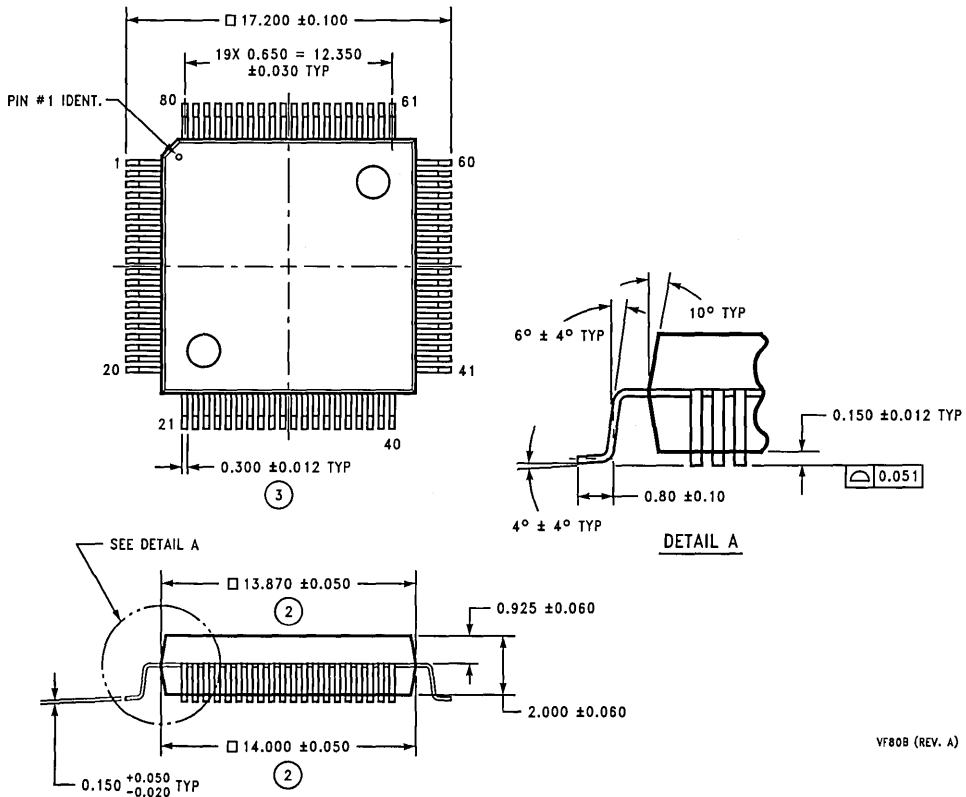
44 Lead Plastic Chip Carrier (V) NS Package Number V44A



68 Lead Plastic Chip Carrier (V) NS Package Number V68A



80 Lead Plastic Quad Flatpack (V) NS Package Number VF80B





Bookshelf of Technical Support Information

National Semiconductor Corporation recognizes the need to keep you informed about the availability of current technical literature.

This bookshelf is a compilation of books that are currently available. The listing that follows shows the publication year and section contents for each book.

Please contact your local National sales office for possible complimentary copies. A listing of sales offices follows this bookshelf.

We are interested in your comments on our technical literature and your suggestions for improvement.

Please send them to:

Technical Communications Dept. M/S 16-300
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

ALS/AS LOGIC DATABOOK—1990

Introduction to Advanced Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky

ASIC DESIGN MANUAL/GATE ARRAYS & STANDARD CELLS—1987

SSI/MSI Functions • Peripheral Functions • LSI/VLSI Functions • Design Guidelines • Packaging

CMOS LOGIC DATABOOK—1988

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC
MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • Surface Mount

DATA ACQUISITION LINEAR DEVICES—1989

Active Filters • Analog Switches/Multiplexers • Analog-to-Digital Converters • Digital-to-Analog Converters
Sample and Hold • Temperature Sensors • Voltage Regulators • Surface Mount

DATA ACQUISITION DATABOOK SUPPLEMENT—1992

New devices released since the printing of the 1989 Data Acquisition Linear Devices Databook.

DISCRETE SEMICONDUCTOR PRODUCTS DATABOOK—1989

Selection Guide and Cross Reference Guides • Diodes • Bipolar NPN Transistors
Bipolar PNP Transistors • JFET Transistors • Surface Mount Products • Pro-Electron Series
Consumer Series • Power Components • Transistor Datasheets • Process Characteristics

DRAM MANAGEMENT HANDBOOK—1991

Dynamic Memory Control • Error Detection and Correction • Microprocessor Applications for the
DP8408A/09A/17/18/19/28/29 • Microprocessor Applications for the DP8420A/21A/22A
Microprocessor Applications for the NS32CG821

EMBEDDED CONTROLLERS DATABOOK—1992

COP400 Family • COP800 Family • COPS Applications • HPC Family • HPC Applications
MICROWIRE and MICROWIRE/PLUS Peripherals • Microcontroller Development Tools

EMBEDDED SYSTEM PROCESSOR DATABOOK—1989

Embedded System Processor Overview • Central Processing Units • Slave Processors • Peripherals
Development Systems and Software Tools

FDDI DATABOOK—1991

FDDI Overview • DP83200 FDDI Chip Set • Development Support • Application Notes and System Briefs

F100K ECL LOGIC DATABOOK & DESIGN GUIDE—1990

Family Overview • 300 Series (Low-Power) Datasheets • 100 Series Datasheets • 11C Datasheets
ECL BiCMOS SRAM, ECL PAL, and ECL ASIC Datasheets • Design Guide • Circuit Basics • Logic Design
Transmission Line Concepts • System Considerations • Power Distribution and Thermal Considerations
Testing Techniques • Quality Assurance and Reliability • Application Notes

FACT™ ADVANCED CMOS LOGIC DATABOOK—1990

Description and Family Characteristics • Ratings, Specifications and Waveforms
Design Considerations • 54AC/74ACXXX • 54ACT/74ACTXXX • Quiet Series: 54ACQ/74ACQXXX
Quiet Series: 54ACTQ/74ACTQXXX • 54FCT/74FCTXXX • FCTA: 54FCTXXXA/74FCTXXXA

FAST® ADVANCED SCHOTTKY TTL LOGIC DATABOOK—1990

Circuit Characteristics • Ratings, Specifications and Waveforms • Design Considerations • 54F/74FXXX

FAST® APPLICATIONS HANDBOOK—1990

Reprint of 1987 Fairchild FAST Applications Handbook

Contains application information on the FAST family: Introduction • Multiplexers • Decoders • Encoders
Operators • FIFOs • Counters • TTL Small Scale Integration • Line Driving and System Design
FAST Characteristics and Testing • Packaging Characteristics

GENERAL PURPOSE LINEAR DEVICES DATABOOK—1989

Continuous Voltage Regulators • Switching Voltage Regulators • Operational Amplifiers • Buffers • Voltage Comparators
Instrumentation Amplifiers • Surface Mount

GRAPHICS HANDBOOK—1989

Advanced Graphics Chipset • DP8500 Development Tools • Application Notes

IBM DATA COMMUNICATIONS HANDBOOK—1992

IBM Data Communications • Application Notes

INTERFACE DATABOOK—1990

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral Power Drivers • Display Drivers
Memory Support • Microprocessor Support • Level Translators and Buffers • Frequency Synthesis • Hi-Rel Interface

LINEAR APPLICATIONS HANDBOOK—1991

The purpose of this handbook is to provide a fully indexed and cross-referenced collection of linear integrated circuit applications using both monolithic and hybrid circuits from National Semiconductor.

Individual application notes are normally written to explain the operation and use of one particular device or to detail various methods of accomplishing a given function. The organization of this handbook takes advantage of this innate coherence by keeping each application note intact, arranging them in numerical order, and providing a detailed Subject Index.

LOCAL AREA NETWORK DATABOOK—1992

Integrated Ethernet Network Interface Controller Products • Ethernet Physical Layer Transceivers
Ethernet Repeater Interface Controller Products • Hardware and Software Support Products • FDDI Products • Glossary

LS/S/TTL DATABOOK—1989

Contains former Fairchild Products

Introduction to Bipolar Logic • Low Power Schottky • Schottky • TTL • TTL—Low Power

MASS STORAGE HANDBOOK—1989

Rigid Disk Pulse Detectors • Rigid Disk Data Separators/Synchronizers and ENDECs
Rigid Disk Data Controller • SCSI Bus Interface Circuits • Floppy Disk Controllers • Disk Drive Interface Circuits
Rigid Disk Preamplifiers and Servo Control Circuits • Rigid Disk Microcontroller Circuits • Disk Interface Design Guide

MEMORY DATABOOK—1990

PROMs, EPROMs, EEPROMs • TTL I/O SRAMs • ECL I/O SRAMs

MICROPROCESSOR DATABOOK—1989

Series 32000 Overview • Central Processing Units • Slave Processors • Peripherals
Development Systems and Software Tools • Application Notes • NSC800 Family

PROGRAMMABLE LOGIC DATABOOK & DESIGN MANUAL—1992

PLD Design and Fabrication • Low Density GAL and PAL Devices • High Density MAPL Family • PLD Development Tools

REAL TIME CLOCK HANDBOOK—1991

Real Time Clocks and Timer Clock Peripherals • Application Notes

RELIABILITY HANDBOOK—1986

Reliability and the Die • Internal Construction • Finished Package • MIL-STD-883 • MIL-M-38510
The Specification Development Process • Reliability and the Hybrid Device • VLSI/VHSIC Devices
Radiation Environment • Electrostatic Discharge • Discrete Device • Standardization
Quality Assurance and Reliability Engineering • Reliability and Documentation • Commercial Grade Device
European Reliability Programs • Reliability and the Cost of Semiconductor Ownership
Reliability Testing at National Semiconductor • The Total Military/Aerospace Standardization Program
883B/RETSM Products • MILS/RETSM Products • 883/RETSM Hybrids • MIL-M-38510 Class B Products
Radiation Hardened Technology • Wafer Fabrication • Semiconductor Assembly and Packaging
Semiconductor Packages • Glossary of Terms • Key Government Agencies • AN/ Numbers and Acronyms
Bibliography • MIL-M-38510 and DESC Drawing Cross Listing

SPECIAL PURPOSE LINEAR DEVICES DATABOOK—1989

Audio Circuits • Radio Circuits • Video Circuits • Motion Control Circuits • Special Function Circuits
Surface Mount

TELECOMMUNICATIONS—1992

COMBO and SLIC Devices • ISDN • Digital Loop Devices • Analog Telephone Components • Software
Application Notes

NOTES

NOTES

NOTES

NOTES

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS

ALABAMA

Huntsville
Arrow Electronics
(205) 837-6955
Bell Industries
(205) 837-1074
Hamilton/Avnet
(205) 837-7210
Pioneer Technology
(205) 837-9300
Time Electronics
(205) 721-1133

ARIZONA

Chandler
Hamilton/Avnet
(602) 961-1211
Phoenix
Arrow Electronics
(602) 437-0750
Tempe
Anthem Electronics
(602) 966-6600
Bell Industries
(602) 966-7800
Time Electronics
(602) 967-2000

CALIFORNIA

Agora Hills
Bell Industries
(818) 706-2608
Time Electronics
(818) 707-2890
Zeus Components
(818) 889-3838
Burbank
Elmo Semiconductor
(818) 768-7400
Calabasas
F/X Electronics
(818) 592-0120
Chatsworth
Anthem Electronics
(818) 775-1333
Arrow Electronics
(818) 701-7500
Time Electronics
(818) 998-7200
Costa Mesa
Hamilton Electro Sales
(714) 641-4100
Cypress
Bell Industries
(714) 895-7801
Gardena
Hamilton/Avnet
(213) 516-8600
Irvine
Anthem Electronics
(714) 768-4444
Rocklin
Anthem Electronics
(916) 624-9744
Bell Industries
(916) 652-0414
Roseville
Hamilton/Avnet
(916) 925-2216
San Diego
Anthem Electronics
(619) 453-9005
Arrow Electronics
(619) 565-4800
Hamilton/Avnet
(619) 571-1900
Time Electronics
(619) 586-0129

San Jose
Anthem Electronics
(408) 453-1200
Arrow Electronics
(408) 441-9700
Pioneer Technology
(408) 954-9100
Zeus Components
(408) 629-4789
Sunnyvale
Bell Industries
(408) 734-8570
Hamilton/Avnet
(408) 743-3300
Time Electronics
(408) 734-9888
Torrance
Time Electronics
(213) 320-0880
Tustin

Arrow Electronics
(714) 838-5422
Time Electronics
(714) 669-0100
Woodland Hills
Hamilton/Avnet
(818) 594-0404
Yorba Linda
Zeus Components
(714) 921-9000

COLORADO

Aurora
Arrow Electronics
(303) 373-5616
Denver
Bell Industries
(303) 691-9010
Englewood
Anthem Electronics
(303) 790-4500
Hamilton/Avnet
(303) 799-7800
Time Electronics
(303) 721-8882
CONNECTICUT
Danbury
Hamilton/Avnet
(203) 743-6077
Shelton
Pioneer Standard
(203) 929-5600
Wallingford
Arrow Electronics
(203) 265-7741
Waterbury
Anthem Electronics
(203) 575-1575

FLORIDA

Altamonte Springs
Bell Industries
(407) 339-0078
Pioneer Technology
(407) 834-9090
Zeus Components
(407) 788-9100
Deerfield Beach
Arrow Electronics
(305) 429-8200
Bell Industries
(305) 421-1997
Pioneer Technology
(305) 428-8877
Fort Lauderdale
Hamilton/Avnet
(305) 767-6377
Time Electronics
(305) 484-7778

Lake Mary
Arrow Electronics
(407) 333-9300
Orlando
Chip Supply
(407) 298-7100
Time Electronics
(407) 841-6565
St. Petersburg
Hamilton/Avnet
(813) 572-4329
Winter Park
Hamilton/Avnet
(407) 657-3300

GEORGIA

Duluth
Arrow Electronics
(404) 497-1300
Hamilton/Avnet
(404) 446-0611
Pioneer Technology
(404) 623-1003
Norcross
Bell Industries
(404) 662-0923
Time Electronics
(404) 368-0969

ILLINOIS

Addison
Pioneer Electronics
(708) 495-9680
Bensenville
Hamilton/Avnet
(708) 860-7700
Elk Grove Village
Bell Industries
(708) 640-1910
Itasca
Arrow Electronics
(708) 250-0500
Schaumburg
Anthem Electronics
(708) 884-0200
Time Electronics
(708) 303-3000

INDIANA

Carmel
Hamilton/Avnet
(317) 844-9333
Fort Wayne
Bell Industries
(219) 423-3422
Indianapolis
Advent Electronics Inc.
(317) 872-4910
Arrow Electronics
(317) 299-2071
Bell Industries
(317) 875-8200
Pioneer Standard
(317) 573-0880

IOWA

Cedar Rapids
Advent Electronics
(319) 363-0221
Arrow Electronics
(319) 395-7230
Hamilton/Avnet
(319) 362-4757

KANSAS

Lenexa
Arrow Electronics
(913) 541-9542
Hamilton/Avnet
(913) 888-8900

MARYLAND

Columbia
Anthem Electronics
(301) 995-6840
Arrow Electronics
(301) 596-7800
Time Electronics
(301) 964-3090
Zeus Components
(301) 997-1118
Gaithersburg
Pioneer Technology
(301) 921-0660

MASSACHUSETTS

Andover
Bell Industries
(508) 474-8880
Beverly
Sertech Laboratories
(508) 927-5820
Lexington
Pioneer Standard
(617) 861-9200
Norwood
Gerber Electronics
(617) 769-6000
Peabody
Hamilton/Avnet
(508) 531-7430
Time Electronics
(508) 532-9900
Tyngsboro
Port Electronics
(508) 649-4880
Wakefield
Zeus Components
(617) 246-8200
Wilmington
Anthem Electronics
(508) 657-5170
Arrow Electronics
(508) 658-0900

MICHIGAN

Grand Rapids
Pioneer Standard
(616) 698-1800
Grandville
Hamilton/Avnet
(616) 243-8805
Livonia
Arrow Electronics
(313) 462-2290
Pioneer Standard
(313) 525-1800
Novi
Hamilton/Avnet
(313) 347-4720
Wyoming
R. M. Electronics, Inc.
(616) 531-9300

MINNESOTA

Eden Prairie
Anthem Electronics
(612) 944-5454
Arrow Electronics
(612) 828-7140
Pioneer Standard
(612) 944-3355
Edina
Arrow Electronics
(612) 830-1800
Time Electronics
(612) 943-2433
Minnetonka
Hamilton/Avnet
(612) 932-0600

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS (Continued)

MISSOURI

Chesterfield
Hamilton/Avnet
(314) 537-1600
St. Louis
Arrow Electronics
(314) 567-6888
Time Electronics
(314) 391-6444

NEW JERSEY

Cherry Hill
Hamilton/Avnet
(609) 424-0100
Fairfield
Hamilton/Avnet
(201) 575-3390
Pioneer Standard
(201) 575-3510
Marlton
Arrow Electronics
(609) 596-8000
Time Electronics
(609) 596-6700
Pine Brook
Anthem Electronics
(201) 227-7960
Arrow Electronics
(201) 227-7880
Wayne
Time Electronics
(201) 758-8250

NEW MEXICO

Albuquerque
Alliance Electronics Inc.
(505) 292-3360
Bell Industries
(505) 292-2700
Hamilton/Avnet
(505) 345-0001

NEW YORK

Binghamton
Pioneer
(607) 722-9300
Buffalo
Summit Electronics
(716) 887-2800
Commack
Anthem Electronics
(516) 864-6600
Fairport
Pioneer Standard
(716) 381-7070
Hauppauge
Arrow Electronics
(516) 231-1000
Hamilton/Avnet
(516) 231-9444
Time Electronics
(516) 273-0100
Port Chester
Zeus Components
(914) 937-7400
Rochester
Arrow Electronics
(716) 427-0300
Hamilton/Avnet
(716) 292-0730
Summit Electronics
(716) 334-8110
Ronkonkoma
Zeus Components
(516) 737-4500
Syracuse
Hamilton/Avnet
(315) 437-2641
Time Electronics
(315) 432-0355

Westbury
Hamilton/Avnet Export Div.
(516) 997-6868
Woodbury
Pioneer Electronics
(516) 921-8700

NORTH CAROLINA

Charlotte
Hamilton/Avnet
(704) 527-2485
Pioneer Technology
(704) 527-8188
Durham
Pioneer Technology
(919) 544-5400
Raleigh
Arrow Electronics
(919) 876-3132
Hamilton/Avnet
(919) 878-0810
Time Electronics
(919) 874-9650

OHIO

Centerville
Arrow Electronics
(513) 435-5563
Cleveland
Pioneer
(216) 587-3600
Columbus
Time Electronics
(614) 794-3301
Dayton
Bell Industries
(513) 435-8660
Bell Industries-Military
(513) 434-8231
Hamilton/Avnet
(513) 439-6700
Pioneer Standard
(513) 236-9900
Zeus Components
(513) 293-6162
Solon
Arrow Electronics
(216) 248-3990
Hamilton/Avnet
(216) 349-5100
Westerville
Hamilton/Avnet
(614) 882-7004

OKLAHOMA

Tulsa
Arrow Electronics
(918) 252-7537
Hamilton/Avnet
(918) 664-0444
Pioneer Standard
(918) 665-7840
Radio Inc.
(918) 587-9123

OREGON

Beaverton
Anthem Electronics
(503) 643-1114
Arrow Electronics
(503) 626-7667
Hamilton/Avnet
(503) 627-0201
Lake Oswego
Bell Industries
(503) 635-6500
Portland
Time Electronics
(503) 684-3780

PENNSYLVANIA

Horsham
Anthem Electronics
(215) 443-5150
Pioneer Technology
(215) 674-4000
Mars
Hamilton/Avnet
(412) 281-4150
Pittsburgh
Pioneer
(412) 782-2300

TEXAS

Austin
Arrow Electronics
(512) 835-4180
Hamilton/Avnet
(512) 837-8911
Minco Technology Labs.
(512) 834-2022
Pioneer Standard
(512) 835-4000
Time Electronics
(512) 346-7346
Carrollton
Arrow Electronics
(214) 380-6464
Dallas
Hamilton/Avnet
(214) 308-8111
Pioneer Standard
(214) 386-7300
Houston
Arrow Electronics
(713) 530-4700
Hamilton/Avnet
(713) 240-7733
Pioneer Standard
(713) 495-4700
Richardson
Anthem Electronics
(214) 238-7100
Time Electronics
(214) 644-4644
Zeus Components
(214) 783-7010

UTAH

Midvale
Bell Industries
(801) 255-9611
Salt Lake City
Anthem Electronics
(801) 973-8555
Arrow Electronics
(801) 973-6913
Hamilton/Avnet
(801) 972-2800
West Valley
Time Electronics
(801) 973-8494

WASHINGTON

Bellevue
Arrow Electronics
(206) 643-4800
Bothell
Anthem Electronics
(206) 483-1700
Kirkland
Time Electronics
(206) 820-1525
Redmond
Bell Industries
(206) 867-5410
Hamilton/Avnet
(206) 241-8555

WISCONSIN

Brookfield
Arrow Electronics
(414) 792-0150
Pioneer Electronics
(414) 784-3480
Mequon
Taylor Electric
(414) 241-4321
Waukesha
Bell Industries
(414) 547-8879
Hamilton/Avnet
(414) 784-8205

CANADA

WESTERN PROVINCES

Burnaby
Hamilton/Avnet
(604) 420-4101
Semad Electronics
(604) 420-9889
Calgary
Electro Sonic Inc.
(403) 255-9550
Semad Electronics
(403) 252-5664
Zentronics
(403) 295-8838
Edmonton
Zentronics
(403) 468-9306
Markham
Semad Electronics Ltd.
(416) 475-3922

RICHMOND

Electro Sonic Inc.
(604) 273-2911
Zentronics
(604) 273-5575

SASKATOON

Zentronics
(306) 955-2207
Winnipeg
Zentronics
(204) 694-1957

EASTERN PROVINCES

Mississauga
Hamilton/Avnet
(416) 795-3825
Time Electronics
(416) 672-5300
Zentronics
(416) 564-9600
Nepean
Hamilton/Avnet
(613) 226-1700
Zentronics
(613) 226-8840
Ottawa
Electro Sonic Inc.
(613) 728-8333
Semad Electronics
(613) 727-8325
Pointe Claire
Semad Electronics
(514) 694-0860
St. Laurent
Hamilton/Avnet
(514) 335-1000
Zentronics
(514) 737-9700
Willowdale
ElectroSonic Inc.
(416) 494-1666
Winnipeg
Electro Sonic Inc.
(204) 783-3105



Databook User Information

The following product variations are listed in this book as Preliminary Information only. Unless otherwise stated, they are not currently scheduled for production release. These variations relate to specific packaging and temperature combinations.

COP98X, 94X, 92X, Family (Scheduled for Q3, 92 Release)

COP988 (Scheduled for Q4, 92 Production Release)

COP68X Family

COP688 Family

COP888EG/V

COP8620C/8622C/86L20C/86L22C

COP888CS/V

COP884CG/WM

HPC36003/26003/16003F20

HPC36003/26003/16003F30

HPC36083XXX/26083XXX/16083XXXF20

HPC36083XXX/26083XXX/16083XXXF30

HPC36004/26004/16004F20

HPC36004/26004/16004F30

HPC36064XXX/26064XXX/16064XXXF20

HPC36064XXX/26064XXX/16064XXXF30

HPC26164XXX/16164XXXF20

HPC46164XXX/36164XXX/26164XXX/16164XXXF30

HPC26400E/16400EV20

HPC46400E/36400E/26400E/16400EV30

The following is a list of HPC products currently in production in ceramic packages.

HPC16003V20

HPC16083XXXV20

HPC16004V20

HPC16064XXXV20

HPC467064EL20

HPC467064EL30

HPC167064EL20

SALES OFFICES

ALABAMA

Huntsville
(205) 721-9367

ARIZONA

Tempe
(602) 966-4563

CALIFORNIA

Rocklin
(916) 632-2750

San Diego
(619) 587-0666

Sunnyvale
(408) 721-8400

Tustin
(714) 259-8880

Woodland Hills
(818) 888-2602

COLORADO

Boulder
(303) 440-3400

Englewood
(303) 790-8090

FLORIDA

Boca Raton
(407) 997-9891

Maitland
(407) 875-8800

GEORGIA

Atlanta
(404) 551-1150

ILLINOIS

Schaumburg
(708) 397-8777

INDIANA

Carmel
(317) 843-7160

Fort Wayne
(219) 436-6844

IOWA

Cedar Rapids
(319) 395-0090

MARYLAND

Hanover
(410) 796-8900

MASSACHUSETTS

Burlington
(617) 221-4500

MICHIGAN

Livonia
(313) 464-0020

MINNESOTA

Bloomington
(612) 854-8200

MISSOURI

St. Louis
(314) 569-9830

NEW JERSEY

Paramus
(201) 599-0955

NEW YORK

Fairport
(716) 223-7700

Wappinger Falls
(914) 298-0680

NORTH CAROLINA

Raleigh
(919) 832-0661

OHIO

Dayton
(513) 435-6886

Independence
(216) 524-5577

ONTARIO

Mississauga
(416) 678-2920

Nepean
(613) 596-0411

OREGON

Portland
(503) 639-5442

PENNSYLVANIA

Horsham
(215) 672-6767

PUERTO RICO

Rio Piedras
(809) 758-9211

QUEBEC

Pointe Claire
(514) 426-2992

TEXAS

Austin
(512) 346-3990

Houston
(713) 894-4888

Richardson
(214) 234-3811

UTAH

Murray
(801) 268-1175

WASHINGTON

Kirkland
(206) 822-4004

WISCONSIN

Brookfield
(414) 782-1818

National Semiconductor Corporation

2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1-800-272-9959
TWX: (910) 339-9240

SALES OFFICES (Continued)

INTERNATIONAL OFFICES
Electronica NSC de Mexico SA

Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: 52-5-524-9402
Fax: 52-5-524-9342

National Semicondutores Do Brasil Ltda.

Av. Brig. Faria Lima, 1409
6.0 Andar
Cep. 01451 J. Paulistano
Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Telex: 391 1131931
Fax: (55/11) 212-1181 NSBR BR

National Semiconductor GmbH

Eschborner Lanstr. 130-132
D-6000 Frankfurt 90
Germany
Tel: (069) 78 91 09-0
Fax: (069) 7 89 43 83

National Semiconductor GmbH

Industriestrasse 10
D-8080 Furstenfeldbruck
Germany
Tel: (0-81-41) 103-0
Telex: 527-649
Fax: (08141) 103554

National Semiconductor GmbH

Misburger Strasse 81D
D3000 Hannover 61
Germany
Tel: (0511) 560040
Fax: (0511) 561740

National Semiconductor GmbH

Untere Waldplatte 37
D-7000 Stuttgart 80
Germany
Tel: 711 686 511
Fax: 711 686 5260

National Semiconductor (UK) Ltd.

The Maple, Kembrey Park
Swindon, Wiltshire SN2 6UT
United Kingdom
Tel: (07-93) 61-41-41
Telex: 444-674
Fax: (07-93) 69-75-22

National Semiconductor Benelux

Vorstlaan 100
B-1170 Brussels
Belgium
Tel: (02) 6-61-06-80
Telex: 61007
Fax: (02) 6-60-23-95

National Semiconductor (UK) Ltd.

Ringager 4A, 3
DK-2605 Brandy
Denmark
Tel: (02) 43-32-11
Telex: 15-179
Fax: (02) 43-31-11

National Semiconductor S.A.

Centre d'Affaires-La Boursidiere
Bâtiment Champagne, B.P. 90
Route Nationale 186
F-92357 Le Plessis Robinson
Paris, France
Tel: (1) 40-94-88-88
Telex: 631065
Fax: (1) 40-94-88-11

National Semiconductor (UK) Ltd.

Unit 2A
Clonskeagh Square
Clonskeagh Road
Dublin 14
Ireland
Tel: (01) 269-55-89
Telex: 91047
Fax: (01) 2830650

National Semiconductor S.p.A.

Strada 7, Palazzo R/3
I-20089 Rozzano
Milanofiori
Italy
Tel: (02) 57 50 03 00
Twx: 352647
Fax: (02) 57 50 04 00

National Semiconductor S.p.A.

Via del Cararaggio, 107
I-00147 Rome
Italy
Tel: (06) 5-13-48-80
Fax: (06) 5-13-79-47

National Semiconductor (UK) Ltd.

Isveien 45
Postboks 57
N-1393 Ostenstad
Norway
Tel: (2) 796500
Fax: (2) 796040

National Semiconductor AB

P.O. Box 1009
Grosshandlarvaegen 7
S-121 23 Johanneshov
Sweden
Tel: 46-8-7228050
Fax: 46-8-7228095
Telex: 10731 NSC S

National Semiconductor GmbH

Calle Agustin de Foxa, 27 (9°D)
E-28036 Madrid
Spain
Tel: (01) 733-2958
Telex: 46133
Fax: (01) 733-8018

National Semiconductor

Switzerland
Alte Winterthurerstrasse 53
Postfach 567
CH-8304 Wallisellen-Zurich
Switzerland
Tel: (01) 830-2727
Telex: 828-444
Fax: (01) 830-1900

National Semiconductor

Kaupparkartanonkatu 7 A22
SF-00930 Helsinki
Finland
Tel: (90) 33-80-33
Telex: 126116
Fax: (90) 33-81-30

National Semiconductor

Postbus 90
NL1380 AB Weesp
The Netherlands
Tel: (0-29-40) 3-04-48
Telex: 10-956
Fax: (0-29-40) 3-04-30

National Semiconductor Japan Ltd.

Sanseido Bldg. 5F
4-15-3 Nishi Shinjuku
Shinjuku-ku
Tokyo 160 Japan
Tel: (03) 3299-7001
Fax: (03) 3299-7000

National Semiconductor

Hong Kong Ltd.
13th Floor, Straight Block
Ocean Centre
5 Canton Road, Tsimshatsui East,
Kowloon, Hong Kong
Tel: (852) 737-1600
Telex: 51292 NSHKL
Fax: (852) 736-9960

National Semiconductor

(Australia) PTY, Ltd.
Bldg. 16, Business Park Dr.
Melbourne, 3168
Victoria, Australia
Tel: (03) 558-9999
Fax: 61-3-558-9998

National Semiconductor (PTE), Ltd.

200 Cantonment Road 13-02
Southpoint 200
Singapore 0208
Tel: 2252229
Telex: RS 50808
Fax: (65) 225-7080

National Semiconductor (Far East)
Ltd.
Taiwan Branch

9th Floor, No. 18
Sec. 1, Chang An East Road,
Taipei, Taiwan R.O.C.
Tel: (86) 521-3288
Telex: 22837 NSTW
Fax: 02 561-3054

National Semiconductor (Far East)
Ltd.
Korea Branch

13th Floor, Dai Han Life Insurance
63 Building,
60, Yoido-dong, Youngdeungpo-ku,
Seoul, Korea 150-763
Tel: (02) 784-8051
Telex: 24942 NSPKLO
Fax: (02) 784-8054