Zilog

November 1988

Z8000™ Family Data Book

Zilog

November 1988

Z8000™ Family Data Book

Zilog

November 1988

Z8000 Family
Data Book

Z8000 Family Data Book
Table of Contents

Zilog

November 1988

Zilog Z8000 Family Architecture
A High-Performance 16-Bit Architecture With 32-Bit Migration
Z8000 ™16 Bit CPU's
Z80,000™ 32 Bit CPU's

In the office, in the factory, even in the home--every day the number of people using microprocessors grows. And every day, these people dream of new applications and better systems. Systems that are faster, more reliable, easier to use, and yet cost less.

To the designer, this vision of the future means building systems with more processing power, communications interfaces, efficient use of large memories, and software that is both more sophisticated and more reliable. To get these systems to market quickly and with minimal development costs, the designer needs powerful building blocks--circuits and software designed to work together.

Zilog's Z8000 Family was born of this vision. Using advances in VLSI technology, the wealth of experience with 16-bit architecture and the overwhelming success of its 8-bit microprocessor family, Zilog conceived the Z8000 Family as a bold answer to the needs of the system designer.

The Established Leader

Zilogs Z80 CPU has become synonymous with high-performance, low-cost computing. Its system-oriented instruction set, efficient use of package and pins, broad range of peripherals, and extensive hardware and software support have earned it first place in the 8-bit world. Used in applications that range from intelligent terminals to powerful microcomputer systems and device controllers, the Z80, Zilog's embodiment of the 8-bit solution, has become the standard of the industry.

16 Bits and Beyond

With this successful precedent clearly in mind, Zilog decided to extend the Z80 tradition to 16 and 32 bits. The new processors place the power of 16 and 32 bits in the designer's hands. Like the Z80 they create a way to apply advanced architectural concepts to solve the real world problems of high performance microprocessor users. They also pave the way for new, compatible industry standards in the 16- and 32-bit CPUs, in peripherals and in software. By drawing on the architecture of minicomputers and mainframes, Zilog looked for and found a breakthrough, the Z8000 Family.

A broad range of processing power and the need to manage vast amounts of memory are inherent in 16- and 32-bit systems. But small systems and real-time performance must not be penalized by these facts. It was essential, therefore, that each device be designed as an integral part of a family concept.

The Z8000 Family is built around a defined set of interconnections and protocols called the Z-BUS, so circuit connections for present and future family members are all compatible. Memory management, DMA transfer, and extended processing have all been planned from the beginning. At the low end, Z80 users can now interface to 16- and 32-bit processors by using the new highly integrated Z280 CPU, a 16-bit CPU that has Z80 code on ZBUS. At the high end Z8000 users can now integrate to the Z80000 32-bit microprocessors and still run their 16-bit software. A high-speed, shared parallel bus, the Z-BUS provides all functions with a communication interface, as Figure 1 illustrates.

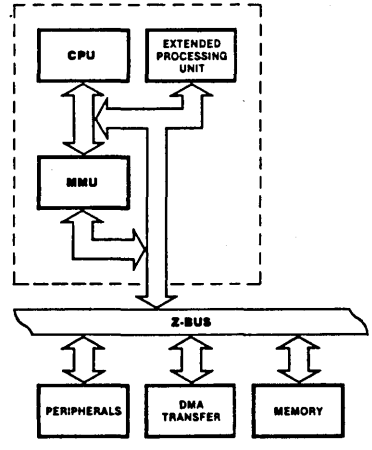


Figure 1. Z-BUS Connects All Functions

System Flexibility.

Even the smallest Z8000 systems offer high throughput and easy programming far superior to any existing microprocessor alternative. In mid-range applications, Z8000 components offer very powerful solutions to the design problems of word processing, intelligent terminals, data communications, instrumentation, and process control. In a complex network of multiple processors, smart peripheral components, and a distributed memory configuration, the Z8000 Family provides performance and versatility exceeding that of much larger--and far more expensive microprocessors.

Higher Throughput

The powerful instruction set, high execution speed, regular architecture, and numerous special features of the Z8000 microprocessors dramatically increase system throughput. Intelligent Z8000 peripheral controllers and extended processing units unburden the CPU and boost throughput even further.

The processing power of the Zilog Z8000 16-bit microprocessor can be boosted beyond its intrinsic capability by Extended Processing Architecture. Simply stated, EPA allows the Z8000 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream, as Figure 2 illustrates.

The use of extended processors to boost the main CPU's performance capability has been proven with large mainframe computers and minicomputers. In these systems, specialized functions such as array processing, special input/output processing, and data communications processing are typically assigned to extended processor hardware. These extended processors are complex computers in their own right.

An Unmatched CPU

The Z8000 microprocessor is not just a wider data path, more registers, more data types, more addressing modes, more instructions, and more addressing space. It brings big-machine concepts to the level of components. Its general-register architecture avoids bottlenecks associated with dedicated or implied registers. Special features support parallel processors, operating systems, compilers, and the implementation of virtual memory.

The Z8000 CPU is also a very fast machine. Its throughput is greater than that of any other 16-bit microprocessor with comparable clock speeds. And the Z8000 CPU is available with speeds ranging from a moderate 6 MHz clock rate that allows you the choice of slow-access, low-cost memories to a high-speed 10 MHz clock rate for high-performance systems. From the three versions of the Z8000 microprocessors, you can select the one best suited to your needs: the Z8001 for large memory applications, the Z8002 for small memory applications, or the Z160 for the low cost, medium memory size applications.

Peripheral Problem Solvers

The Z8000 Peripherals offer more than simple answers to the basic needs of a microcomputer system. Complicated system tasks that previously required burdensome MSI circuitry can now be handled off-line. Even such highly specialized functions as data encryption/decryption are performed by Zilog peripherals. These multifunction peripherals are extensively programmable so each can be precisely tailored to its application. Each can be made to perform complex, intelligent tasks on its own--to unburden the CPU, reduce bus traffic, and increase system throughput.

Counting, timing, and parallel I/O, for example, are made easy by the Z8036 Z-CIO Counter and Parallel I/O Circuit with its three 16-bit counter/timers and three 8-bit parallel I/O ports. It can even function as a programmable interrupt-priority controller.

Ease of implementation characterizes the interface between the 16-bit, multiplexed Z8000 CPU and its Z-BUS peripherals. The Z-BUS ensures not only that communications between Family members are consistently simple, but also that the resolution of interrupt priorities requires minimal CPU involvement.

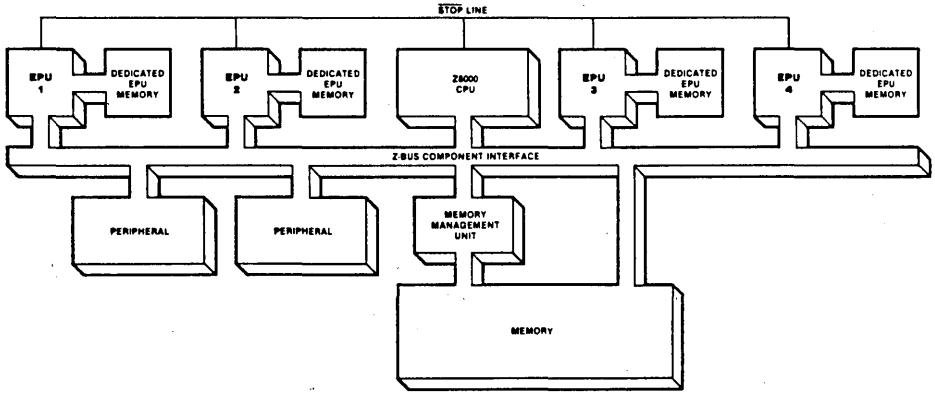


Figure2. Typical Extended Processor Configuration

Data communications are deftly handled by the Z8030 Z-SCC Serial Communications Controller; a dual-channel multi-protocol component that supports all popular communications formats. Now also available in CMOS.

Direct memory access is supported by the Z8016 DTC Transfer Controller, a fast dual-channel device that enhances memory and I/O data transfers within stand-alone processor or parallel processor environments.

Elements of asynchronous parallel-processing systems are interconnected by the Z8038 FIFO unit, a surprisingly flexible device whose buffer depth can be expanded without limit using the Z8060 FIFO Buffer Unit.

The Z8068 Z-DCP Data Ciphering Processor, supporting three standard ciphering options and key parity checking, provides encryption and decryption of data where needed. The Z-DCP can input, output, and encipher simultaneously.

The Z8010 Z-MMU Memory Management Unit provides flexibility in code segmenter page relocation and sophistication in memory protection rarely found in the microprocessor world. This device encourages modular software development--a critical factor as programs reach new levels of complexity.

Universal Peripherals

Extend the Range of Applications to increase the range of applications for its peripherals, Zilog selected certain of the multiplexed Z-BUS-compatible Z8000 Peripherals to be produced in a second bus version: a non-multiplexed 16-bit CPU-compatible line, called the Z8500 Universal Peripheral. The Z8536 CIO, Z8530 SCC, and the Z08516 DTC are all "Universal" versions of their Z8000 counterparts and as such incorporate the same extensive features to perform the same impressive functions. The Z8038 Z-FIO, and by extension the Z8060 FIFO, are compatible with both bus versions, and so endow both Families with their multiple strengths.

To help meet the timing requirements of both Zilog and non-Zilog microprocessors, the Z8581 CGC Clock Generator and Controller has been added to the Z8500 Universal Peripherals group. The CGC outputs drive the Z80 and Z8000 CPUs clock inputs directly; no bus interface is required. Selective clock-stretching abilities provide a variety of timing outputs to make this a versatile chip suitable for VLSI and LSI devices. Table 1 summarizes the Z8000 CPU and peripheral offering.

Simply put, the Z8000 Family offers more for less money. The Z8000 microprocessors give mid-range minicomputer performance at microprocessor cost. At component prices, Z8000 peripheral controllers perform complex system functions that previously required an entire PC board.

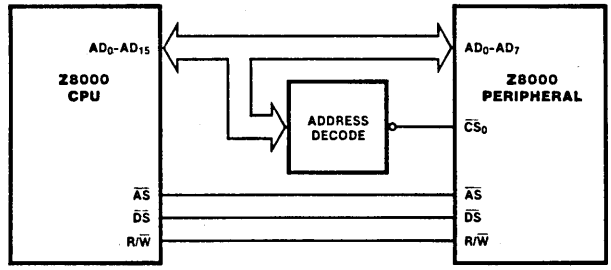


Figure 3. Z8000 Interface

The 32-bit Migration

Zilog has now completed the migration from the 16-bit to the 32-bit CPU, with the Z80,000 and Z80320 or the Z320. Software compatible to the Z8000 CPUs, the Z80,000 and Z320 provide flexibility at the cost of a 16-bit CPU. Oriented to the applications in which high throughput is required, its file of 16 general-purpose 32-bit registers handles bytes, words, and long words with equal facility. The rich instruction set combines powerful addressing modes and operations in a manner that aids assembly-language coding of time-critical applications, and still provides the completeness desirable for efficient compiler-generated code.

The Z320 CPU can be configured under software control to use 16-bit logical addresses (ideally suited for high-speed controller applications) or 32-bit addresses (for large-system tasks). The 32-bit address modes support both a linear addressing space and an alternative segmented addressing space, which are selected by the user according to the application's requirements.

Other system features include System and Normal modes of operation, a sophisticated trapping mechanism, a high-performance bus structure, and built-in multiprocessor support with global memory access arbitration signals for easy request and acknowledge handshaking.

An on-chip cache and memory management unit (MMU), coupled with a sophisticated instruction pipeline, enable the Z320 to execute instructions at a rate of up to one instruction per processor cycle. The 256-byte cache provides an automatic buffering mechanism to hold the most recently fetched instructions and data on the chip. Thus, subsequent references to these items do not require lengthy memory transactions but instead can be fetched in a single processor cycle.

The memory management unit on the chip contains all the information needed to translate the most recently used logical addresses generated by the CPU into the physical addresses used by the memory system. With each address translation, access attributes are automatically checked to determine whether or not the access is permitted. The MMU can be used to implement a virtual memory or can be disabled entirely for applications that do not need memory management.

Peripheral Support.

The Z320 uses Zilogs Z-BUS so the entire Z8000 family of circuits are available for use with it. Multifunction Z-BUS peripherals are extensively programmable, so each can be precisely tailored to an application.

Z-BUS Component Interconnect

The Z-BUS is a high-speed parallel shared bus that links the Z8000, Z320 microprocessor families and Extended Processing Units with the peripherals needed to implement complete systems. Through a common communications interface, Z-BUS peripherals and CPUs support the following types of transactions:

Data Transfer. 16 or 32 bits of data can be moved between bus controllers (such as a CPU) and associated peripherals.

Interrupts. Interrupts can be generated by peripherals and serviced by CPUs over the bus.

Resource Control. A daisy chain priority mechanism supports distributed management of shared resources which includes peripheral devices and the bus itself.

The heart of the Z-BUS is a set of multiplexed address/data lines and the signals that control these lines. Multiplexing data and address onto the same lines makes more efficient use of pins and facilitates expansion of the number of data and address bits. Multiplexing also allows straight-forward addressing of a peripheral's internal registers, which greatly simplifies I/O programming.

A daisy-chained priority mechanism resolves interrupt and resource requests, thus allowing distributed control of the bus and eliminating the need for separate priority controllers.

The resource-control daisy chain also allows wide physical separation of components.

Furthermore, Z-BUS is asynchronous in the sense that peripherals need not be sychronized with the CPU clock. All timing information is provided by Z-BUS signals.
As a result of a common hardware interface and protocol, users can be assured that adequate system support for their Z8000 or Z320 system design is readily available for the Z-BUS peripherals and Extended Processing Units:

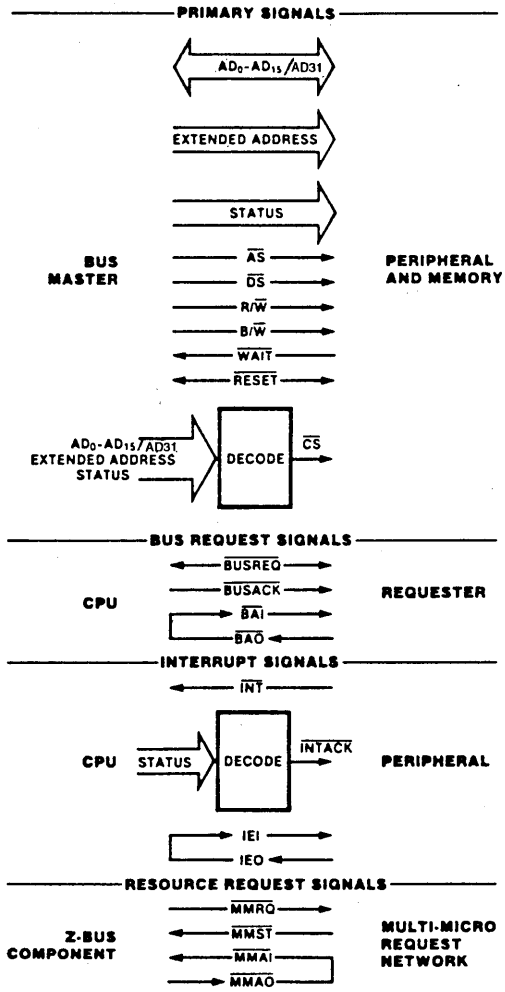


Figure 4. Z-BUS Signals

TABLE - I Z8000 PRODUCT OFFERING

Part No	Device Name	Description	Speed	Package
Z08001/2	Z8001/ Z8002 16-bit CPU	16-bit Internal/External CPU, 16x 16-bit General Purpose Registers 8M Byte Addressing	6,10	P,V,C,D,L
Z08160	Z160 16-bit CPU	16-bit Internal/External CPU, 16x 16-bit General Purpose Registers 2M Byte Addressing	6,10	V
Z80320 Z80000	Z320 Z80K 32-bit CPU	32-bit Internal/External CPU, 32x 32-bit General Purpose Registers 4G Byte Addressing	8,10	V,G
Z08010	MMU	Provides Dynamic Memory Segment Relocation of Blocks from 256 to 65,532 Bytes, Protection Features	6,10	P
Z08016 Z08516	DTC	2 Independent, Multi-functional Channels that Control Memory Transfers up to 6M Byte/sec	4,6	P,V
Z08030 Z08530	SCC	2 Independent, Full Duplex Channels Transfer Rates from 0 to 1.5M bits/sec, Sync & Asynchronous Modes	4,6,8	P,D,V,C,L
Z08036 Z08536	CIO	2 Independent 8-bit General Purpose Devices, satisfies most Counter/Timer and Parallel I/O needs	4,6	P,D,V,C,L
Z08038	FIO	128 byte, Async, Bidirectional FIFO Buffer with I/O Control Logic on Board	4*,6	D,P,V,C,L
Z08060	FIFO	128 x 8-bit Memory, Bidirectional Asynchronous Data Transfer Capability	1MB/sec	P,C
Z08068	DCP	Encrypts and Decrypts Data Using Three Standard Ciphering Modes	4	P
Z08581	CGC	2 Independent 20 MHz Oscillators Output Directly to Z80, Z8000 and other CPUs	6,10	P,C,L
Z80C30	CMOS Z-SCC	CMOS Version is Pin Compatible with the Standard NMOS Device	6, 8,10	P,V,C

TABLE - I Z8000 PRODUCT OFFERING (Cont.)

Part No	Device Name	Description	Speed	Package
Z85C30	CMOS SCC	CMOS Version is Pin Compatible with the Standard NMOS Device	6,8,10	P,V,C
Z0765A	765A	Floppy Disk Controller (FDC) Floppy Disk interfaces Microprocessors to Control up to Four Floppy Disk Drives	8	P,V
Z7220A	7220A	High-Performance Graphics Display Controller Interfaces with Micro-processors to Generate Displays	8	V
Z05380	SCSI	CMOS, Asynchronous SCSI Protocol	1.0MB/sec	P,V

Packages: P=Plastic DIP, V=PLCC, C=Ceramic, D=Cerdip, G=Pin Grid Array, L=LCC, *=Available while 4 MHz supply lasts.

Z160™ CPU
Central Processing Unit

October 1988

FEATURES

- **Fully software compatible member of the Z8000® architecture.**
- Instruction set more powerful than many minicomputers
- **Directly addresses 2 Mbytes in 32 segments.**
- Eight user-selectable addressing modes
- Seven data types that range from bits to 32-bit long words and byte and word strings
- System and Normal operating modes
- Separate code, data, and stack spaces
- Sophisticated interrupt structure
- Resource-shaping capabilities for multiprocessing systems
- Multi-programming support
- Compiler support
- 32-bit operations, including signed multiply and divide
- Z-BUS compatible
- **6 and 10 MHz clock rate**
- **Small, low-cost 44-pin PLCC package for surface mount applications.**

GENERAL DESCRIPTION

The Z8000 is an advanced high-end 16-bit microprocessor that spans a wide variety of applications ranging from simple stand-alone computers to complex parallel-processing systems. Essentially a monolithic minicomputer central processing unit, the Z8000 CPU is characterized by an instruction set more powerful than many minicomputers; abundant resources in registers, data types, addressing modes and addressing range, and a regular architecture that enhances throughput by avoiding critical bottlenecks such as implied or dedicated registers.

CPU resources include sixteen 16-bit general-purpose registers, seven data types that range from bits to 32-bit long words and byte and word strings, and eight user-selectable addressing modes. The 110 distinct instruction types can be combined with the various data types and addressing modes to form a powerful set of 414 instructions. Moreover, the instruction set is regular; most instructions can use any of the five main addressing modes and can operate on byte, word, and long-word data types.

The CPU can operate in either the system or normal mode. The distinction between these two modes permits privileged operations, thereby improving operating system organization and implementation. Multiprogramming is supported by the "atomic" Test and Set instruction; multiprocessing by a combination of instruction and

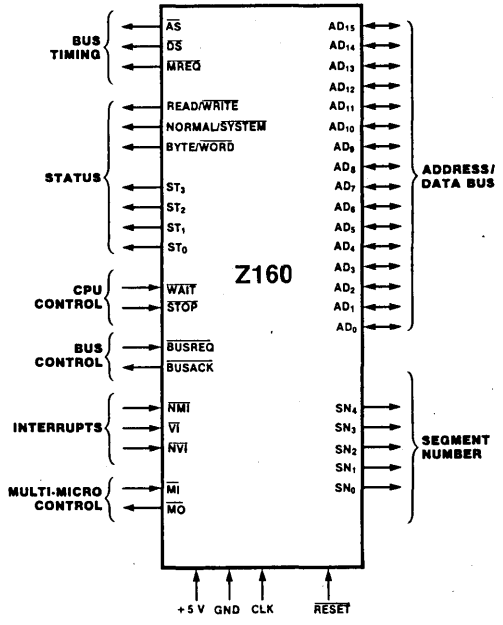


Figure 1. Z160 CPU Pin Functions

hardware features; and compilers by multiple stacks, special instructions, and addressing modes.

The Z160 is a segmented CPU (Figure 1). It can directly address 2 megabytes of memory. The two operating modes - system and normal - and the distinction between code, data, and stack spaces within each mode allows memory extension up to 12 megabytes.

The Z160 is fabricated with high-density, high-performance scaled n-channel silicon-gate depletion-load technology, and is housed in leadless chip carriers (LCC).

REGISTER ORGANIZATION

The Z160 CPU is a register-oriented machine that offers sixteen 16-bit general-purpose registers and a set of special system registers. All general-purpose registers can be used as accumulators and all but one as index registers or memory pointers.

Register flexibility is created by grouping and overlapping multiple registers (Figures 2 and 3). For byte operations, the first eight 16-bit registers (R0... R7) are treated as sixteen 8-bit registers (RL0, RH0..., RL7, RH7). The sixteen 16-bit registers are grouped in pairs (RR0... RR14) to form 32-bit long-word registers. Similarly, the register set is grouped in quadruples (RQ0... RQ12) to form 64-bit registers.

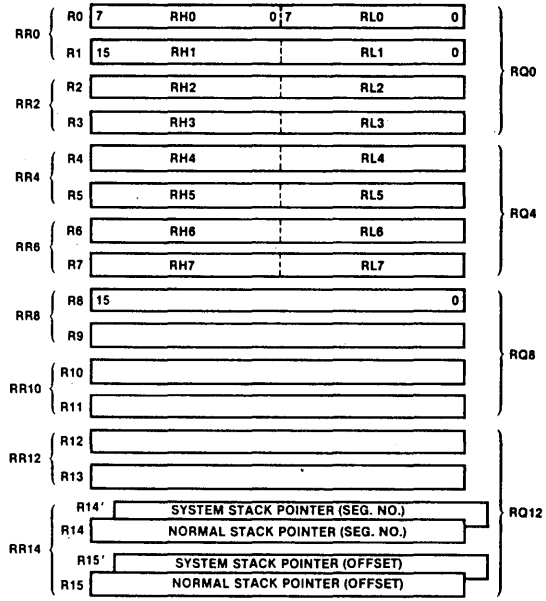


Figure 2. Z160 General-Purpose Registers

STACKS

The Z160 can use stacks located anywhere in memory. Call and Return instructions as well as interrupts and traps use implied stacks. The distinction between normal and system stacks separates system information from the application program information. Two stack pointers are available: the system stack pointer and the normal stack pointer. Because they are part of the general-purpose register group, the user can manipulate the stack pointers with any instruction available for register operations.

The Z160 register pair RR14 is the implied stack pointer. Register R14 contains the 5-bit segment number and R15 contains the 16-bit offset.

REFRESH

The Z160 CPU contains a counter that can be used to automatically refresh dynamic memory. The refresh counter register consists of a 9-bit row counter, a 6-bit rate counter, and an enable bit (Figure 4). The 9-bit row counter can address up to 256 rows and is incremented by two each time the rate counter reaches end-of-count. The rate counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler (n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed by 1 to 64 μs with a 4 MHz clock. Refresh can be disabled by programming the refresh enable/disable bit.

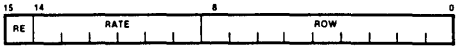
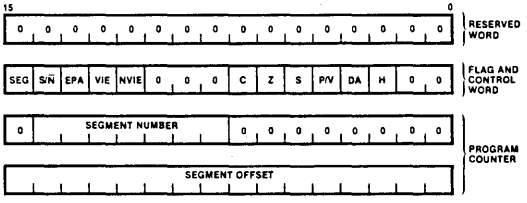


Figure 4. Refresh Counter

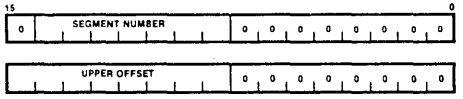
PROGRAM STATUS INFORMATION

This group of status registers contains the program counter, flags, and control bits. When an interrupt or trap occurs, the entire group is saved and a new program status group is loaded.

Figure 5 illustrates the program status groups of the Z160. The program status group consists of four words: a two-word program counter, the flag and control word, and an unused word reserved for future use. Five bits of the first PC word designate one of the 32 memory segments. The second word supplies the 16-bit offset that designates a memory location within the segment.



Z08001/Z160 Program Status Registers



Z08001/Z160 Program Status Area Pointer

Figure 5. Z160 CPU Special Registers

INTERRUPT AND TRAP STRUCTURE

The Z160 provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention and are generally triggered by peripherals needing service. Traps are sychronous events resulting from the execution of certain instructions. Both are processed in a similar manner by the CPU.

The CPU supports three types of interrupts (non-maskable, vectored, and non-vectored) and three traps [system call, Extended Process Architecture (EPA) instruction and privileged instructions]. The vectored and non-vectored interrupts are maskable.

The traps occur when instructions limited to the system mode are used in normal mode, or as a result of the System Call instruction, or for an EPA instruction. The descending order or priority for traps and interrupts is: internal traps, nonmaskable interrupt, vectored interrupt and non-vectored interrupt.

When an interrupt or trap occurs, the current program status is automatically pushed on the system stack. The program status consists of the processor status (PC and FCW) plus a 16-bit identifier. The identifier contains the reason or source of the trap or interrupt. For internal traps, the identifier is the first word of the trapped instruction. For external traps or interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle.

After saving the current program status, the new program status is automatically loaded from the program status area in system memory. This area is designated by the program status area pointer (PSAP).

DATA TYPES

Z160 instructions can operate on bits, BCD digits (4 bits), bytes (8 bits), words (16 bits), long words (32 bits), and byte strings and word strings (up to 64 kilobytes long). Bits can be set, reset, and tested; digits are used in BCD arithmetic operations; bytes are used for characters or small integer values; words are used for integer values, instructions and nonsegmented addresses; long words are used for long integer values and segmented addresses. All data elements except strings can reside either in registers or memory. Strings are stored in memory only.

The basic data element is the byte. The number of bytes used when manipulating a data element is either implied by the operation or—for strings and multiple register operations—explicitly specified in the instruction.

SEGMENTATION AND MEMORY MANAGEMENT

High-level languages, sophisticated operating systems, large programs and data bases, and decreasing memory prices are all accelerating the trend toward larger memory requirements in microcomputer systems. The Z160 meets this requirement with a two megabyte addressing space. This large address space is directly accessed by the CPU using a segmented addressing scheme.

Segmented Addressing

A segmented addressing space-compared with linear addressing-is closer to the way a programmer uses memory because each procedure and data space resides in its own segment. The two megabytes of Z160 addressing space is divided into 32 relocatable segments up to 64 kilobytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any location relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately. The segmented Z160 can run any code written for the nonsegmented Z8002 in any one of its 32 segments, provided it is set to the nonsegmented mode.

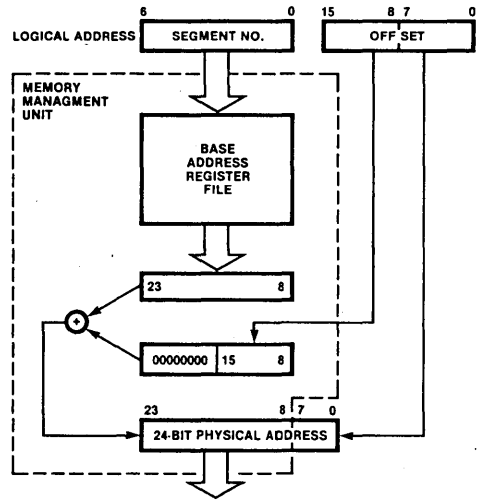


Figure 6. Logical-to-Physical Address Translation

In hardware, segmented addresses are contained in a register pair or long-word memory location. The segment number and offset can be manipulated separately or together by all the available word and long-word operations.

When contained in an instruction, a segmented address has two different representations: long offset and short offset. The long offset occupies two words, whereas the short offset requires only one and combines in one word the 5-bit segment number with an 8-bit offset (range 0-256). The short offset mode allows very dense encoding of addresses and minimizes the need for long addresses required by direct accessing of this large address space.

EXTENDED PROCESSING ARCHITECTURE

The Zilog Extended Processing Architecture (EPA) provides an extremely flexible and modular approach to expanding both the hardware and software capabilities of the Z8000 CPU. Features of the EPA include:

- Specialized instructions for external processors or software traps may be added to CPU instruction set.
- Increases throughput of the system by using up to four specialized external processors in parallel with the CPU.
- Permits modular design of Z8000-based systems.
- Provides easy management of multiple microprocessor configurations via "single instruction stream" communication.
- Simple interconnection between extended processing units and Z8000 CPU requires no additional external supporting logic.
- Supports debugging of suspect hardware against proven software.
- Standard features on all Zilog Z8000 CPUs.

Specific benefits include:

- EPUs can be added as the system grows and as EPUs with specialized functions are developed.
- Control of EPUs is accomplished via a "single instruction stream" in the Z8000 CPU, eliminating many significant system software and bus contention management obstacles that occur in other multiprocessor (e.g., master-slave) organization schemes.

The processing power of the Zilog Z8000 16-bit microprocessor can be boosted beyond its intrinsic capability by Extended Processing Architecture. Simply stated, EPA allows the Z8000 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream (Figure 7).

The use of extended processors to boost the main CPU's performance capability has been proven with large mainframe computers and minicomputers. In these systems, specialized functions such as array processing, special input/output processing, and data communications processing are typically assigned to extended processor hardware. These extended processors are complex computers in their own right.

The Zilog Extended Processing Architecture combines the best concepts of these proven performance boosters with the latest in high-density MOS integrated-circuit design. The result is an elegant expansion of design capability—a powerful microprocessor architecture capable of connecting single-chip EPUs that permits very effective parallel processing and makes for a smoothly integrated instruction stream from the Z8000 programmer's point of view. A typical addition to the current Z8000 instruction set is a set of Floating Point Instructions.

The Extended Processing Units connect directly to the Z8000 Bus (Z-BUS) and continuously monitor the CPU instruction stream. When an extended instruction is detected, the appropriate EPU responds, obtaining or

placing data or status information on the Z-BUS using the Z8000-generated control signals and performing its function as directed.

The Z8000 CPU is responsible for instructing the EPU and delivering operands and data to it. The EPU recognizes instructions intended for it and executes them, using data supplied with the instruction and/or data within its internal registers. There are four classes of EPU instructions:

- Data transfers between main memory and EPU registers
- Data transfers between CPU registers and EPU registers
- EPU internal operations
- Status transfers between the EPUs and the Z8000 CPU Flag and Control Word register (FCW)

Four Z8000 addressing modes may be utilized with transfers between EPU registers and the CPU and main memory:

- Register
- Indirect Register
- Direct Address
- Index

In addition to the hardware-implemented capabilities of the Extended Processing Architecture, there is an extended instruction trap mechanism to permit software simulation of EPU functions. A control bit in the Z8000 FCW register indicates whether actual EPUs are present or not. If not, when an extended instruction is detected, the Z8000 traps on the instruction, so that a software "trap handler" can emulate the desired EPU function—a very useful development tool. The EPA software trap routine supports the debugging of suspect hardware against proven software. This feature will increase in significance as designers become familiar with the EPA capability of the Z8000 CPU.

This software trap mechanism facilitates the design of systems for later addition of EPUs: initially, the extended function is executed as a trap subroutine; when the EPU is finally attached, the trap subroutine is eliminated and the EPA control bit is set. Application software is unaware of the change.

Extended Processing Architecture also offers protection against extended instruction overlapping. Each EPU connects to the Z8000 CPU via the STOP line so that if an EPU is requested to perform a second extended instruction function before it has completed the previous one, it can put the CPU into the Stop/Refresh state until execution of the previous extended instruction is complete.

EPA and CPU instruction execution are shown in Figure 8. The CPU begins operation by fetching an instruction and determining whether it is a CPU or an EPU command. The EPU meanwhile monitors the Z-BUS for its own instructions. If the CPU encounters an EPU command, it checks to see whether an EPU is present; if not, the EPU may be simulated by an EPU instruction trap software routine; if an EPU is present, the necessary data and/or address is placed on the Z-BUS. If the EPU is free when the instruction and data for it appear, the extended instruction is executed. If the EPU is still processing a previous instruction, it activates the CPU's STOP line to lock the CPU off at the Z-BUS until execution is complete. After the instruction is finished, the EPU deactivates the STOP line and CPU transactions continue.

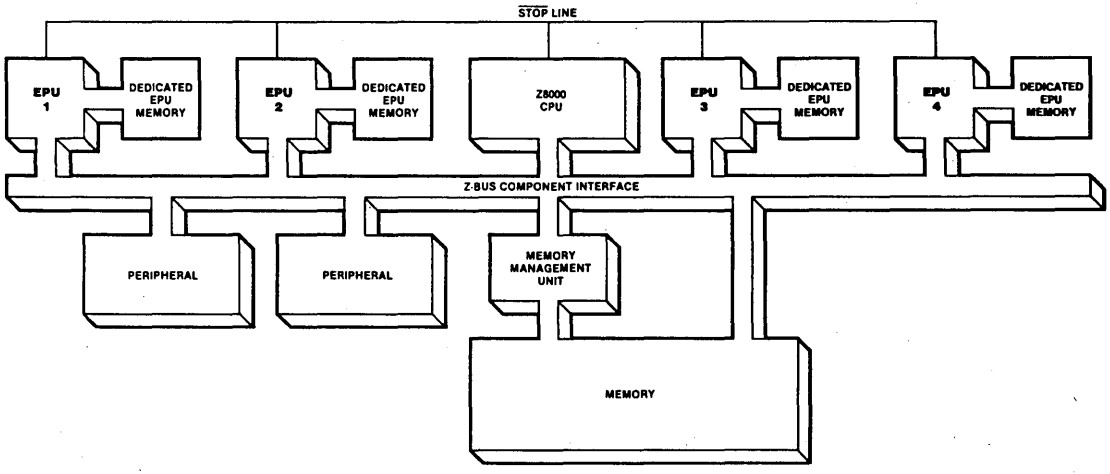
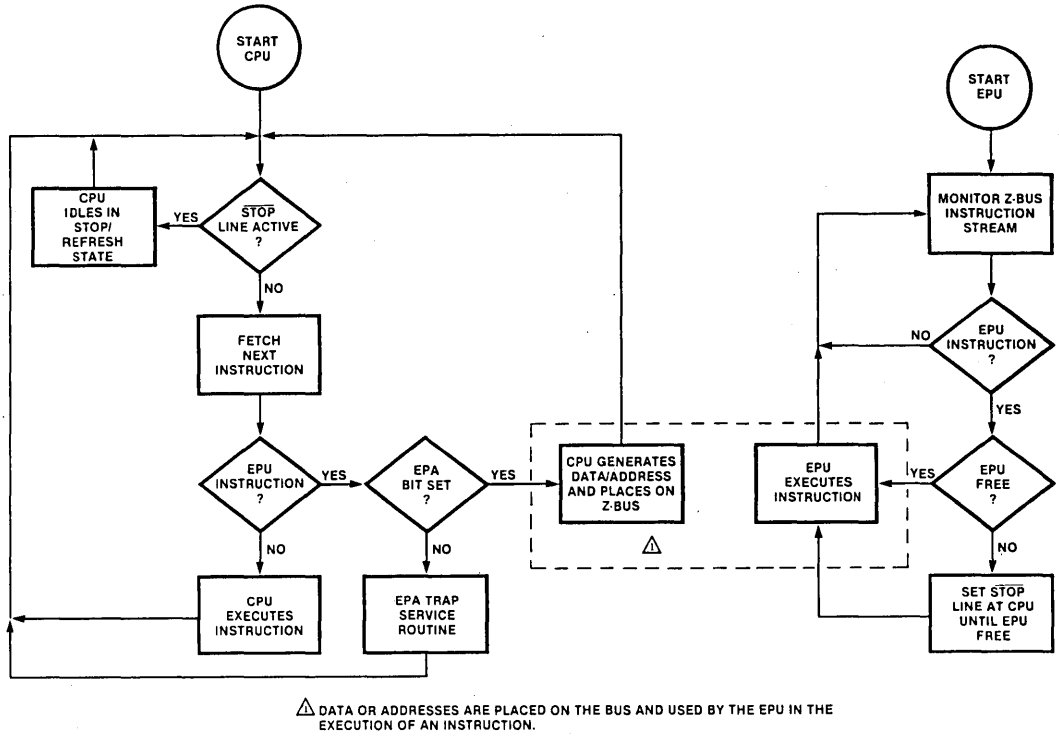


Figure 7. Typical Extended Processor Configuration



⚠ DATA OR ADDRESSES ARE PLACED ON THE BUS AND USED BY THE EPU IN THE EXECUTION OF AN INSTRUCTION.

Figure 8. EPA and Z8000 CPU Instruction Execution

INPUT/OUTPUT

A set of I/O instructions performs 8-bit or 16-bit transfers between the CPU and I/O devices. I/O devices are addressed with a 16-bit I/O port address. The I/O port address is similar to a memory address; however, I/O address space need not be part of the memory address space. I/O port and memory addresses coexist on the same bus lines and they are distinguished by the status outputs.

Two types of I/O instructions are available: standard and special. Each has its own address space. The I/O instructions include a comprehensive set of In, Out, and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status information distinguishes between standard and special I/O references.

MULTI-MICROPROCESSOR SUPPORT

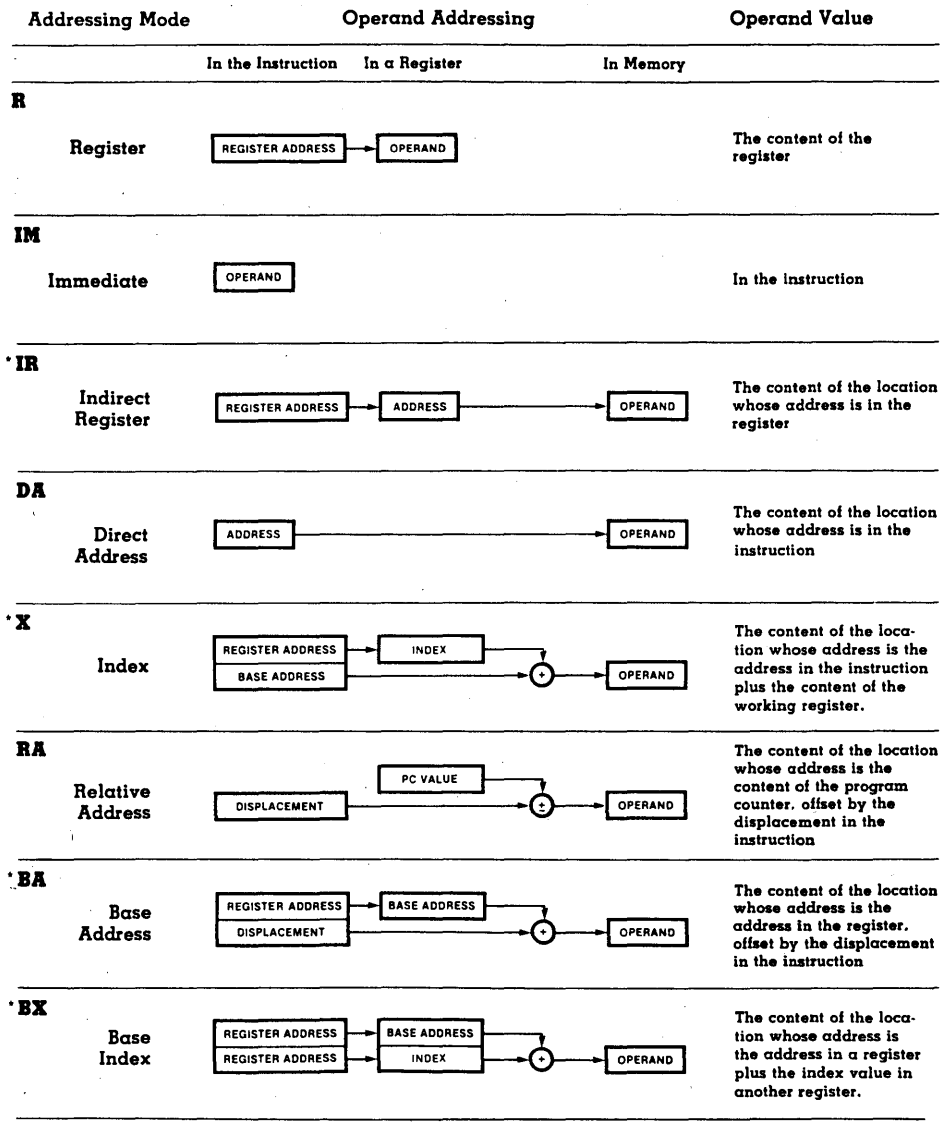
Multi-microprocessor systems are supported in hardware and software. A pair of CPU pins is used in conjunction with certain instructions to coordinate multiple microprocessors. The Multi-Micro Out pin issues a request for the resource, while the Multi-Micro In pin is used to recognize the state of the resource. Thus, any CPU in a multiple microprocessor system can exclude all other asynchronous CPUs from a critical shared resource.

Multi-microprocessor systems are supported in software by the instructions Multi-Micro Request, Test Multi-Micro In, Set Multi-Micro Out, and Reset Multi-Micro Out. In addition, the eight megabyte CPU address space is beneficial in multiple microprocessor systems that have large memory requirements.

ADDRESSING MODES

The information included in Z8000 instructions consists of the function to be performed, the type and size of data elements to be manipulated, and the location of the data elements. Locations are designated by register addresses, memory addresses, or I/O addresses. The addressing mode of a given instruction defines the address space it references and the method used to compute the address itself. Addressing modes are explicitly specified or implied by the instruction.

Figure 9 illustrates the eight addressing modes: Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Index (X), Relative Address (RA), Base Address (BA), and Base Index (BX). In general, an addressing mode explicitly specifies either register address space or memory address space. Program memory address space and I/O address space are usually implied by the instruction.



*Do not use R0 or RR0 as indirect, index, or base registers.

Figure 9. Addressing Modes

INSTRUCTION SET SUMMARY

The Z8000 provides the following types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

LOAD AND EXCHANGE

			Clock Cycles*						
		Addr.	Word, Byte			Long Word			
Mnemonics	Operands	Modes	NS	SS	SL	NS	SS	SL	Operation
CLR	dst	R	7	7	7				**Clear**
CLRB		IR	8	8	8				dst ← 0
		DA	11	12	14				
		X	12	12	15				
EX	R, src	R	6	6	6				**Exchange**
EXB		IR	12	12	12				R ↔ src
		DA	15	16	18				
		X	16	16	19				
LD	R, src	R	3	3	3	5	5	5	**Load into Register**
LDB		IM	7	7	7	11	11	11	R ← src
LDL		IM	5 (byte only)						
		IR	7	7	7	11	11	11	
		DA	9	10	12	12	13	15	
		X	10	10	13	13	13	16	
		BA	14	14	14	17	17	17	
		BX	14	14	14	17	17	17	
LD	dst, R	IR	8	8	8	11	11	11	**Load into Memory** (Store)
LDB		DA	11	12	14	14	15	17	dst ← R
LDL		X	12	12	15	15	15	18	
		BA	14	14	14	17	17	17	
		BX	14	14	14	17	17	11	
LD	dst, IM	IR	11	11	11				**Load Immediate into Memory**
LDB		DA	14	15	17				dst ← IM
		X	15	15	18				
LDA	R, src	DA	12	13	15				**Load Address**
		X	13	13	16				R ← source address
		BA	15	15	15				
		BX	15	15	15				
LDAR	R, src	RA	15	15	15				**Load Address Relative**
									R ← source address
LDK	R, src	IM	5	5	5				**Load Constant**
									R ← n (n = 0... 15)
LDM	R, src, n	IR	11	11	11 + 3n				**Load Multiple**
		DA	14	15	17 + 3n				R ← src (n consecutive words)
		X	15	15	18 + 3n				(n = 1... 16)

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

LOAD AND EXCHANGE (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
LDM	dst, R, n	IR	11	11	11 + 3n				**Load Multiple** (Store Multiple)
		DA	14	15	17 + 3n				dst ← R (n consecutive words)
		X	15	15	18 + 3n				(n = 1... 16)
LDR	R, src	RA	14	14	14	17	17	17	**Load Relative**
LDRB									R ← src
LDRL									(range −32768... +32767)
LDR	dst, R	RA	14	14	14	17	17	17	**Load Relative** (Store Relative)
LDRB									dst ← R
LDRL									(range −32768... +32767)
POP	dst, IR	R	8	8	8	12	12	12	**Pop**
POPL		IR	12	12	12	19	19	19	dst ← IR
		DA	16	16	18	23	23	25	Autoincrement contents of R
		X	16	16	19	23	23	26	
PUSH	IR, src	R	9	9	9	12	12	12	**Push**
PUSHL		IM	12	12	12	19	19	19	Autodecrement contents of R
		IR	13	13	13	20	20	20	IR ← src
		DA	14	14	16	21	21	23	
		X	14	14	17	21	21	24	

ARITHMETIC

ADC	R, src	R	5	5	5				**Add with Carry**
ADCB									R ← R + src + carry
ADD	R, src	R	4	4	4	8	8	8	**Add**
ADDB		IM	7	7	7	14	14	14	R ← R + src
ADDL		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP	R, src	R	4	4	4	8	8	8	**Compare with Register**
CPB		IM	7	7	7	14	14	14	R − src
CPL		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP	dst, IM	IR	11	11	11				**Compare with Immediate**
CPB		DA	14	15	17				dst − IM
		X	15	15	18				
DAB	dst	R	5	5	5				**Decimal Adjust**
DEC	dst, n	R	4	4	4				**Decremented by n**
DECB		IR	11	11	11				dst ← dst − n
		DA	13	14	16				(n = 1... 16)
		X	14	14	17				

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

ARITHMETIC (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
DIV	R, src	R	107	107	107	744	744	744	**Divide** (signed)
DIVL		IM	107	107	107	744	744	744	Word: $R_{n+1} \leftarrow R_{n,n+1}$ + src
		IR	107	107	107	744	744	744	$R_n \leftarrow$ remainder
		DA	108	109	111	745	746	748	Long Word: $R_{n+2,n+3} \leftarrow R_{n...n+3}$ + src
		X	109	109	112	746	746	749	$R_{n,n+2} \leftarrow$ remainder
EXTS	dst	R	11	11	11	11	11	11	**Extend Sign**
EXTSB									Extend sign of low order half of dst
EXTSL									through high order half of dst
INC	dst, n	R	4	4	4				**Increment by n**
INCB		IR	11	11	11				dst ← dst + n
		DA	13	14	16				(n = 1... 16)
		X	14	14	17				
MULT	R, src	R	70	70	70	282†	282†	282†	**Multiply** (signed)
MULTL		IM	70	70	70	282†	282†	282†	Word: $R_{n,n+1} \leftarrow R_{n+1}$ • src
		IR	70	70	70	282†	282†	282†	Long Word: $R_{n...n+3} \leftarrow R_{n+2,n+3}$
		DA	71	72	74	283†	284†	286†	†Plus seven cycles for each 1 in the
		X	72	72	75	284†	284†	287†	multiplicand
NEG	dst	R	7	7	7				**Negate**
NEGB		IR	12	12	12				dst ← 0 − dst
		DA	15	16	18				
		X	16	16	19				
SBC	R, src	R	5	5	5				**Subtract with Carry**
SBCB									R ← R − src − carry
SUB	R, src	R	4	4	4	8	8	8	**Subtract**
SUBB		IM	7	7	7	14	14	14	R ← R − src
SUBL		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

LOGICAL

AND	R, src	R	4	4	4				**AND**
ANDB		IM	7	7	7				R ← R AND src
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				
COM	dst	R	7	7	7				**Complement**
COMB		IR	12	12	12				dst ← NOT dst
		DA	15	16	18				
		X	16	16	19				
OR	R, src	R	4	4	4				**OR**
ORB		IM	7	7	7				R ← R OR src
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

LOGICAL (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
TCC **TCCB**	cc, dst	R	5	5	5				**Test Condition Code** Set LSB if cc is true
TEST **TESTB** **TESTL**	dst	R	7	7	7	13	13	13	**Test** dst OR 0
		IR	8	8	8	13	13	13	
		DA	11	12	14	16	17	19	
		X	12	12	15	17	17	20	
XOR **XORB**	R, src	R	4	4	4				**Exclusive OR** R ← R XOR src
		IM	7	7	7				
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				

PROGRAM CONTROL

CALL	dst	IR	10	15	15				**Call Subroutine** Autodecrement SP @ SP ← PC PC ← dst
		DA	12	18	20				
		X	13	18	21				
CALR	dst	RA	10	10	15				**Call Relative** Autodecrement SP @ SP ← PC PC←PC + dst (range −4094 to +4096)
DJNZ **DBJNZ**	R, dst	RA	11	11	11				**Decrement and Jump if Non-Zero** R ← R − 1 If R ≠ 0: PC←PC + dst(range − 254 to 9)
IRET†	—	—	13	13	16				**Interrupt Return** PS ← @ SP Autoincrement SP
JP	cc, dst	IR	10	10	15	(taken)			**Jump Conditional** If cc is true: PC ← dst
		IR	7	7	7	(not taken)			
		DA	7	8	10				
		X	8	8	11				
JR	cc, dst	RA	6	6	6				**Jump Conditional Relative** If cc is true: PC ← PC + dst (range −256 to +254)
RET	cc	—	10	10	13	(taken)			**Return Conditional** If cc is true: PC ← @ SP Autoincrement SP
			7	7	7	(not taken)			
SC	src	IM	33	33	39				**System Call** Autodecrement SP @ SP ← old PS Push instruction PS ← System Call PS

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

BIT MANIPULATION

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
BIT **BITB**	dst, b	R IR DA X	4 8 10 11	4 8 11 11	4 8 13 14				**Test Bit Static** Z flag ← NOT dst bit specified by b
BIT **BITB**	dst, R	R	10	10	10				**Test Bit Dynamic** Z flag ← NOT dst bit specified by contents of R
RES **RESB**	dst, b	R IR DA X	4 11 13 14	4 11 14 14	4 11 16 17				**Reset Bit Static** Reset dst bit specified by b
RES **RESB**	dst, R	R	10	10	10				**Reset Bit Dynamic** Reset dst bit specified by contents R
SET **SETB**	dst, b	R IR DA X	4 11 13 14	4 11 14 14	4 11 16 17				**Set Bit Static** Set dst bit specified by b
SET **SETB**	dst, R	R	10	10	10				**Set Bit Dynamic** Set dst bit specified by contents of R
TSET **TSETB**	dst	R IR DA X	7 11 14 15	7 11 15 15	7 11 17 18				**Test and Set** S flag ← MSB of dst dst ← all 1s

ROTATE AND SHIFT

RL **RLB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left** by n bits (n = 1, 2)
RLC **RLCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left through Carry** by n bits (n = 1, 2)
RLDB	R, src	R	9	9	9				**Rotate Digit Left**
RR **RRB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right** by n bits (n = 1, 2)
RRC **RRCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right through Carry** by n bits (n = 1, 2)
RRDB	R, src	R	9	9	9				**Rotate Digit Right**
SDA **SDAB** **SDAL**	dst, R	R	(15 + 3 n)			(15 + 3 n)			**Shift Dynamic Arithmetic** Shift dst left or right by contents of R
SDL **SDLB** **SDLL**	dst, R	R	(15 + 3 n)			(15 + 3 n)			**Shift Dynamic Logical** Shift dst left or right by contents of R

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

ROTATE AND SHIFT (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
SLA **SLAB** **SLAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Arithmetic** by n bits
SLL **SLLB** **SLLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Logical** by n bits
SRA **SRAB** **SRAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Arithmetic** by n bits
SRL **SRLB** **SRLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Logical** by n bits

BLOCK TRANSFER AND STRING MANIPULATION

CPD **CPDB**	R_X,src,R_Y,cc	IR	20	20	20				**Compare and Decrement** R_X − src Autodecrement src address $R_Y \leftarrow R_Y - 1$
CPDR **CPDRB**	R_X,src,R_Y,cc	IR	(11 + 9 n)						**Compare, Decrement, and Repeat** R_X − src Autodecrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPI **CPIB**	R_X,src,R_Y,cc	IR	20	20	20				**Compare and Increment** R_X − src Autoincrement src address $R_Y \leftarrow R_Y - 1$
CPIR **CPIRB**	R_X,src,R_Y,cc	IR	(11 + 9 n)						**Compare, Increment, and Repeat** R_X − src Autoincrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPSD **CPSDB**	dst,src,R,cc	IR	25	25	25				**Compare String and Decrement** dst − src Autodecrement dst and src addresses R ← R − 1
CPSDR **CPSDRB**	dst,src,R,cc	IR	(11 + 14 n)						**Compare String, Decrement, and Repeat** dst − src Autodecrement dst and src addresses R ← R − 1 Repeat until cc is true or R = 0

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

BLOCK TRANSFER AND STRING MANIPULATION (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CPSI **CPSIB**	dst,src,R,cc	IR	25	25	25				**Compare String and Increment** dst – src Autoincrement dst and src addresses R ← R – 1
CPSIR **CPSIRB**	dst,src,R,cc	IR	(11 + 14 n)						**Compare String, Increment and Repeat** dst – src Autoincrement dst and src addresses R ← R – 1 Repeat until cc is true or R = 0
LDD **LDDB**	dst,src,R	IR	20	20	20				**Load and Decrement** dst ← src Autodecrement dst and src addresses R ← R – 1
LDDR **LDDRB**	dst,src,R	IR	(11 + 9 n)						**Load, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R – 1 Repeat until R = 0
LDI **LDIB**	dst,src,R	IR	20	20	20				**Load and Increment** dst ← src Autoincrement dst and src addresses R ← R – 1
LDIR **LDIRB**	dst,src,R	IR	(11 + 9 n)						**Load, Increment and Repeat** dst ← src Autoincrement dst and src addresses R ← R – 1 Repeat until R = 0
TRDB	dst,src,R	IR	25	25	25				**Translate and Decrement** dst ← src (dst) Autodecrement dst address R ← R – 1
TRDRB	dst,src,R	IR	(11 + 14 n)						**Translate, Decrement and Repeat** dst ← src (dst) Autodecrement dst address R ← R – 1 Repeat until R = 0
TRIB	dst,src,R	IR	25	25	25				**Translate and Increment** dst ← src (dst) Autoincrement dst address R ← R – 1

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
*Privileged instruction. Executed in system mode only.

BLOCK TRANSFER AND STRING MANIPULATION (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
TRIRB	dst,src,R	IR	(11 + 14 n)						**Translate, Increment and Repeat** dst ← src (dst) Autoincrement dst address R ← R − 1 Repeat until R = 0
TRTDB	src1,src2,R	IR	25	25	25				**Translate and Test, Decrement** RH1 ← src2 (src1) Autodecrement src 1 address R ← R − 1
TRTDRB	src1,src2,R	IR	(11 + 14 n)						**Translate and Test, Decrement, and Repeat** RH1 ← src2 (src1) Autodecrement src1 address R ← R − 1 Repeat until R = 0 or RH1 = 0
TRTIB	src1,src2,R	IR	25	25	25				**Translate and Test, Increment** RH1 ← src2 (src1) Autoincrement src1 address R ← R − 1
TRTIRB	src1,src2,R	IR	(11 + 14 n)						**Translate and Test, Increment and Repeat** RH1 ← src2 (src1) Autoincrement src 1 address R ← R − 1 Repeat until R = 0 or RH1 = 0

INPUT/OUTPUT

Mnemonics	Operands	Addr. Modes	Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
IN†	R,src	IR	10	10	10				**Input** R ← src
INB†		DA	12	12	12				
IND†	dst,src,R	IR	21	21	21				**Input and Decrement** dst ← src Autodecrement dst address R ← R − 1
INDB†									
INDR†	dst,src,R	IR	(11 + 10 n)						**Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R − 1 Repeat until R = 0
INDRB†									
INI†	dst,src,R	IR	21	21	21				**Input and Increment** dst ← src Autoincrement dst address R ← R − 1
INIB†									

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

INPUT/OUTPUT (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
INIR† **INIRB**†	dst,src,R	IR	(11 + 10 n)						**Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R − 1 Repeat until R = 0
OUT† **OUTB**†	dst,R	IR	10	10	10				**Output** dst ← R
		DA	12	12	12				
OUTD† **OUTDB**†	dst,src,R	IR	21	21	21				**Output and Decrement** dst ← src Autodecrement src address R ← R − 1
OTDR† **OTDRB**†	dst,src,R	IR	(11 + 10 n)						**Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R − 1 Repeat until R = 0
OUTI† **OUTIB**†	dst,src,R	IR	21	21	21				**Output and Increment** dst ← src Autoincrement src address R ← R − 1
OTIR† **OTIRB**†	dst,src,R	IR	(11 + 10 n)						**Output, Increment, and Repeat** dst ← src Autoincrement src address R ← R − 1 Repeat until R = 0
SIN† **SINB**†	R,src	DA	12	12	12				**Special Input** R ← src
SIND† **SINDB**†	dst,src,R	IR	21	21	21				**Special Input and Decrement** dst ← src Autodecrement dst address R ← R − 1
SINDR† **SINDRB**†	dst,src,R	IR	11 + 10 n)						**Special Input, Decrement, and** Repeat dst ← src Autodecrement dst address R ← R − 1 Repeat until R = 0
SINI† **SINIB**†	dst,src,R	IR	21	21	21				**Special Input and Increment** dst ← src Autoincrement dst address R ← R − 1

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

INPUT/OUTPUT (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
SINIR† **SINIRB**†	dst,src,R	IR	(11 + 10 n)						**Special Input, Increment, and** Repeat dst ← src Autoincrement dst address R ← R − 1 Repeat until R = 0
SOUT† **SOUTB**†	dst,src	DA	12	12	12				**Special Output** dst ← src
SOUTD† **SOUTDB**†	dst,src,R	IR	21	21	21				**Special Output and Decrement** dst ← src Autodecrement src address R ← R − 1
SOTDR† **SOTDRB**†	dst,src,R	IR	(11 + 10 n)						**Special Output, Decrement, and** Repeat dst ← src Autodecrement src address R ← R − 1 Repeat until R = 0
SOUTI† **SOUTIB**†	dst,src,R	IR	21	21	21				**Special Output and Increment** dst ← src Autoincrement src address R ← R − 1
SOTIR† **SOTIRB**†	dst,src,R	R	(11 + 10 n)						**Special Output, Increment, and** Repeat dst ← src Autoincrement src address R ← R − 1 Repeat until R = 0

CPU CONTROL

COMFLG	flags	—	7	7	7				**Complement Flag** (Any combination of C, Z, S, P/V)
DI†	int	—	7	7	7				**Disable Interrupt** (Any combination of NVI, VI)
EI†	int	—	7	7	7				**Enable Interrupt** (Any combination of NVI, VI)
HALT†	—	—	(8 + 3 n)						**HALT**
LDCTL†	CTLR,src	R	7	7	7				**Load into Control Register** CTLR ← src
LDCTL†	dst,CTLR	R	7	7	7				**Load from Control Register** dst ← CTLR

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

CPU CONTROL (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
LDCTLB	FLGR,src	R	7	7	7				**Load into Flag Byte Register** FLGR ← src
LDCTLB	dst,FLGR	R	7	7	7				**Load from Flag Byte Register** dst ← FLGR
LDPS†	src	IR	12	16	16				**Load Program Status** PS ← src
		DA	16	20	22				
		X	17	20	23				
MBIT†	—	—	7	7	7				**Test Multi-Micro Bit** Set S if MI is Low; reset S if MI is High
MREQ†	dst	R		(12 + n)					**Multi-Micro Request**
MRES†	—	—	5	5	5				**Multi-Micro Reset**
MSET†	—	—	5	7	7				**Multi-Micro Set**
NOP	—	—	7	7	7				**No Operation**
RESFLG	flag	—	7	7	7				**Reset Flag** (Any combination of C, Z, S, P/V)
SETFLG	flag	—	7	7	7				**Set Flag** (Any combination of C, Z, S, P/V)

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

CONDITION CODES

Code	Meaning	Flag Settings	CC Field
F	Always false	—	0000
T	Always true	—	1000
Z	Zero	Z = 1	0110
NZ	Not zero	Z = 0	1110
C	Carry	C = 1	0111
NC	No Carry	C = 0	1111
PL	Plus	S = 0	1101
MI	Minus	S = 1	0101
NE	Not equal	Z = 0	1110
EQ	Equal	Z = 1	0110
OV	Overflow	P/V = 1	0100
NOV	No overflow	P/V = 0	1100
PE	Parity is even	P/V = 1	0100
PO	Parity is odd	P/V = 0	1100
GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001
LT	Less than (signed)	(S XOR P/V) = 1	0001
GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010
LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010
UGE	Unsigned greater than or equal	C = 0	1111
ULT	Unsigned less than	C = 1	0111
UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011
ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Note that some condition codes have identical flag settings and binary fields in the instruction:
Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO

STATUS CODE LINES

Note: A full list of instructions and cycle times is contained in the Z8000 data manual.

ST_0-ST_3	Definition
0000	Internal operation
0001	Memory refresh
0010	I/O reference
0011	Special I/O reference (e.g., to an MMU)
0100	Segment trap acknowledge
0101	Non-maskable interrupt acknowledge
0110	Non-vectored interrupt acknowledge
0111	Vectored interrupt acknowledge
1000	Data memory request
1001	Stack memory request
1010	Data memory request (EPU)
1011	Stack memory request (EPU)
1100	Program reference, nth word
1101	Instruction fetch, first word
1110	Extension processor transfer
1111	Reserved

PIN DESCRIPTION

AD_0-AD_{15}. *Address/Data* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used for I/O and to address memory.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates addresses are valid.

$\overline{BUSACK}$. *Bus Acknowledge* (output active Low). A Low on this line indicates the CPU has relinquished control of the bus.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). This line must be driven Low to request the bus from the CPU.

B/$\overline{W}$. *Byte/Word* (output, Low = Word, 3-state). This signal defines the type of memory reference on the 16-bit address/data bus.

CLK. *System Clock* (input). CLK is a 5V single-phase time-base input.

$\overline{DS}$. *Data Strobe* (output, active Low, 3-state). This line times the data in and out of the CPU.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). A Low on this line indicates that the address/data bus holds a memory address.

$\overline{MI}$, $\overline{MO}$. *Multi-Micro In, Multi-Micro Out* (input and output, active Low). These two lines form a resource-request daisy chain that allows one CPU in a multi-microprocessor system to access a shared resource.

$\overline{NMI}$. *Non-Maskable Interrupt* (edge triggered, input, active Low). A high-to-low transition on $\overline{NMI}$ requests a non-maskable interrupt. The $\overline{NMI}$ interrupt has the highest priority of the three types of interrupts.

N/$\overline{S}$. *Normal/System Mode* (output, Low = System Mode, 3-state). N/$\overline{S}$ indicates the CPU is in the normal or system mode.

$\overline{NVI}$. *Non-Vectored Interrupt* (input, active Low). A Low on this line requests a non-vectored interrupt.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the CPU.

R/$\overline{W}$. *Read/Write* (output, Low = Write, 3-state). R/$\overline{W}$ indicates that the CPU is reading from or writing to memory or I/O.

SN_0-SN_4. ***Segment Number*** **(outputs, active High, 3-state). These lines provide the 5-bit segment number used to address one of 32 segments.**

ST_0-ST_3. *Status* (outputs, active High, 3-state). These lines specify the CPU status (see Status Code Lines).

$\overline{STOP}$. *Stop* (input, active Low). This input can be used to single-step instruction execution.

$\overline{VI}$. *Vectored Interrupt* (input, active Low). A Low on this line requests a vectored interrupt.

$\overline{WAIT}$. *Wait* (input, active Low). This line indicates to the CPU that the memory or I/O device is not ready for data transfer.

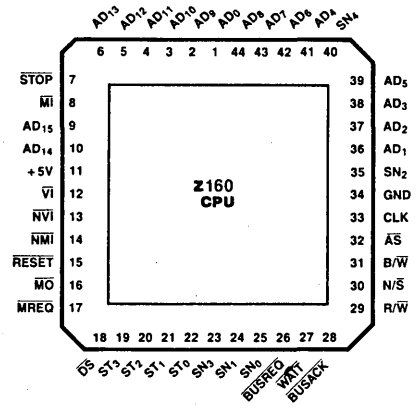


Figure 10b. 44-pin PLCC, Pin Assignments

Z160 CPU TIMING

The Z160 CPU executes instructions by stepping through sequences of basic machine cycles, such as memory read or write, I/O device read or write, interrupt acknowledge, and internal execution. Each of these basic cycles requires three to ten clock cycles to execute. Instructions that require more clock cycles to execute are broken up into several machine cycles. Thus no machine cycle is longer than ten clock cycles and fast response to a Bus Request is guaranteed.

The instruction opcode is fetched by a normal memory read operation. A memory refresh cycle can be inserted just after the completion of any first instruction fetch (IF_1) cycle and can also be inserted while the following instructions are being executed: MULT, MULTL, DIV, DIVL, HALT, all Shift instructions, all Block Move instructions, and the Multi-Micro Request instruction (MREQ).

The following timing diagrams show the relative timing relationships of all CPU signals during each of the basic operations. When a machine cycle requires additional clock cycles for CPU internal operation, one to five clock cycles are added. Memory and I/O read and write, as well as interrupt acknowledge cycles, can be extended by activating the $\overline{WAIT}$ input. For exact timing information, refer to the composite timing diagram.

Note that the $\overline{WAIT}$ input is not synchronized in the Z160 and that the setup and hold times for $\overline{WAIT}$, relative to the clock, must be met. If asynchronous $\overline{WAIT}$ signals are generated, they must be synchronized with the CPU clock before entering the Z160 .

MEMORY READ AND WRITE

Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 outputs. During a memory read cycle, a 16-bit address is placed on the AD_0-AD_{15} outputs early in the first clock period, as shown in Figure 12. The 5-bit segment number is output on SN_0-SN_4 one clock period earlier than the 16-bit address offset.)

A valid address is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The state of the $\overline{WAIT}$ input is sampled in the middle of the second clock cycle by the falling edge of Clock. If $\overline{WAIT}$ is Low, an additional clock period is added between T_2 and T_3. $\overline{WAIT}$ is sampled again in the middle of this wait cycle, and additional wait states can be inserted: this allows interfacing slow memories. No control outputs change during wait states.

Although memory is word organized, memory is addressed as bytes. All instructions are word-aligned, using even addresses. Within a 16-bit word, the most significant byte (D_8-D_{15}) is addressed by the low-order address (A_0 = Low), and the least significant byte (D_0-D_7) is addressed by the high-order address (A_0 = High).

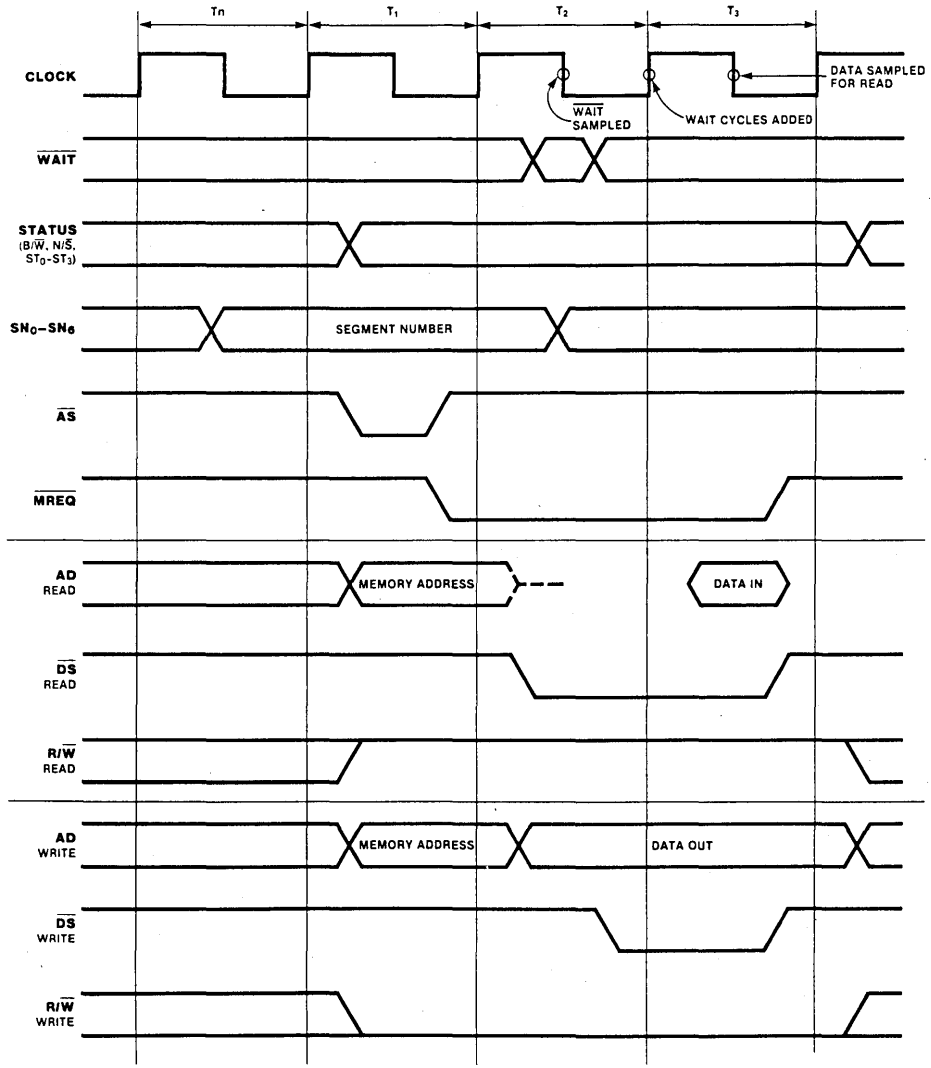


Figure 12. Memory Read and Write Timing

INPUT/OUTPUT

I/O timing is similar to memory read/write timing, except that one wait state is automatically (T_{WA}) inserted between T_2 and T_3 (Figure 13). The segmented **Z160** uses 16-bit I/O addresses.

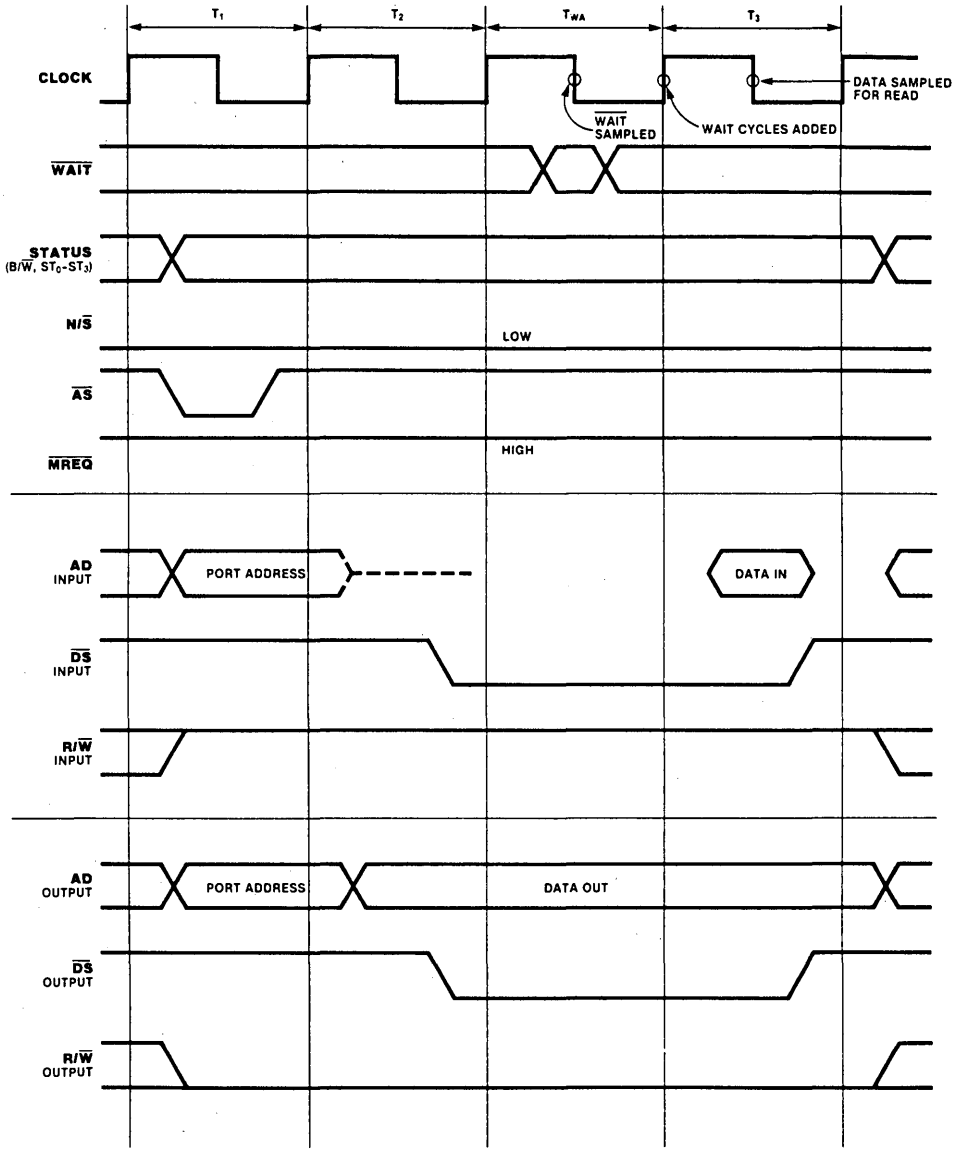


Figure 13. Input/Output Timing

INTERRUPT AND SEGMENT TRAP REQUEST AND ACKNOWLEDGE

The Z160 CPU recognizes three intererupt inputs (nonmaskable, vectored, and nonvectored). Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge detected and sets the internal $\overline{NMI}$ latch. The $\overline{VI}$ and $\overline{NVI}$ inputs, as well as the state of the internal NMI latch, are sampled at the end of T_2 in the last machine cycle of any instruction.

In response to an interrupt or trap, the subsequent IF_1 cycle is exercised, but ignored. The internal state of the CPU is not altered and the instruction will be refetched and executed after the return from the interrupt routine. The program counter is not updated, but the system stack pointer is decremented in preparation for pushing starting information onto the system stack.

The next machine cycle is the interrupt acknowledge cycle. This cycle has five automatic wait states, with additional wait states possible, as shown in Figure 14.

After the last wait state, the CPU reads the information on AD_0-AD_{15} and temporarily stores it, to be saved on the stack later in the acknowledge sequence. This word identifies the source of the interrupt or trap. For the nonvectored and nonmaskable interrupts, all 16 bits can represent peripheral device status information. For the vectored interrupt, the low byte is the jump vector, and the high byte can be extra user status.

After the acknowledge cycle, the N/$\overline{S}$ output indicates the automatic change to system mode.

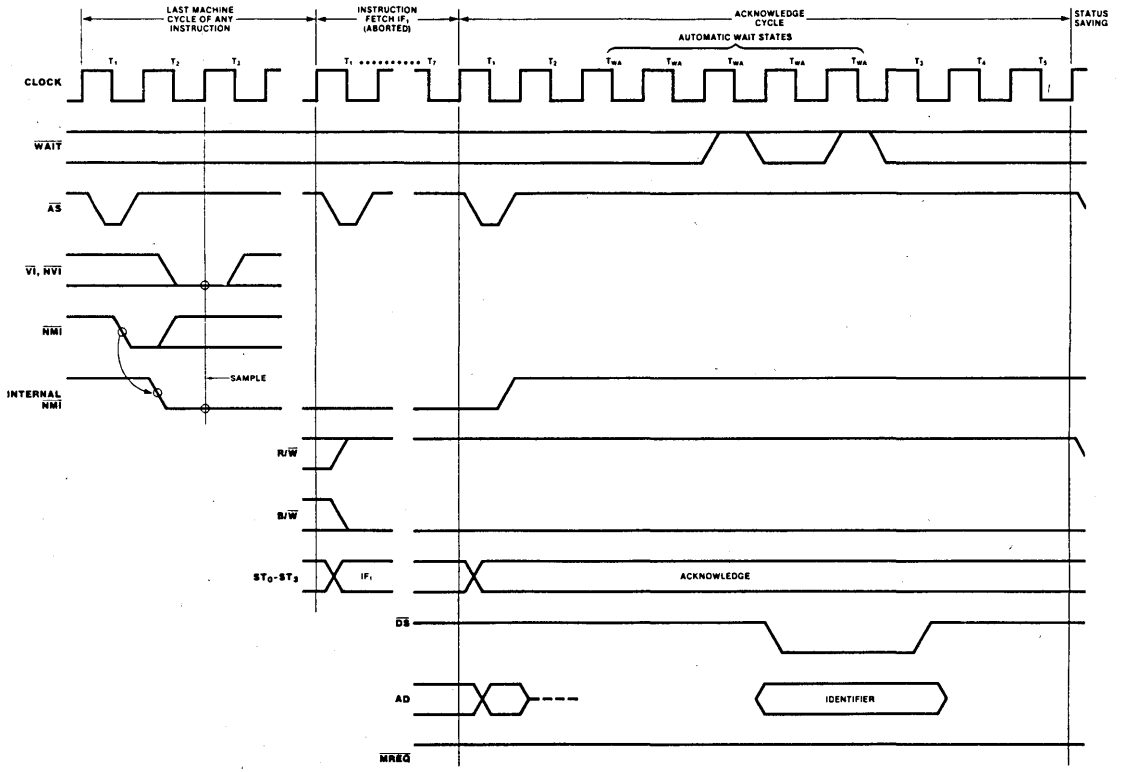


Figure 14. Interrupt and Segment Trap Request/Acknowledge Timing

STATUS SAVING SEQUENCE

The machine cycles, following the interrupt acknowledge or segmentation trap acknowledge cycle, push the old status information on the system stack in the following order: the 16-bit program counter; the 5-bit segment number; the flag control word; and finally, the interrupt/trap identifier. Subsequent machine cycles fetch the new program status from the program status area, and then branch to the interrupt/trap service routine.

BUS REQUEST ACKNOWLEDGE TIMING

A Low on the $\overline{\text{BUSREQ}}$ input indicates to the CPU that another device is requesting the Address/Data and control buses. The asynchronous $\overline{\text{BUSREQ}}$ input is synchronized at the beginning of any machine cycle (Figure 15). $\overline{\text{BUSREQ}}$ takes priority over WAIT. If $\overline{\text{BUSREQ}}$ is Low, an internal synchronous $\overline{\text{BUSREQ}}$ signal is generated, which—after completion of the current machine cycle—causes the $\overline{\text{BUSACK}}$ output to go Low and all bus outputs to go into the high-impedance state. The requesting device—typically a DMA—can then control the bus.

When $\overline{\text{BUSREQ}}$ is released, it is synchronized with the rising clock edge; the $\overline{\text{BUSACK}}$ output goes High one clock period later, indicating that the CPU will again take control of the bus.

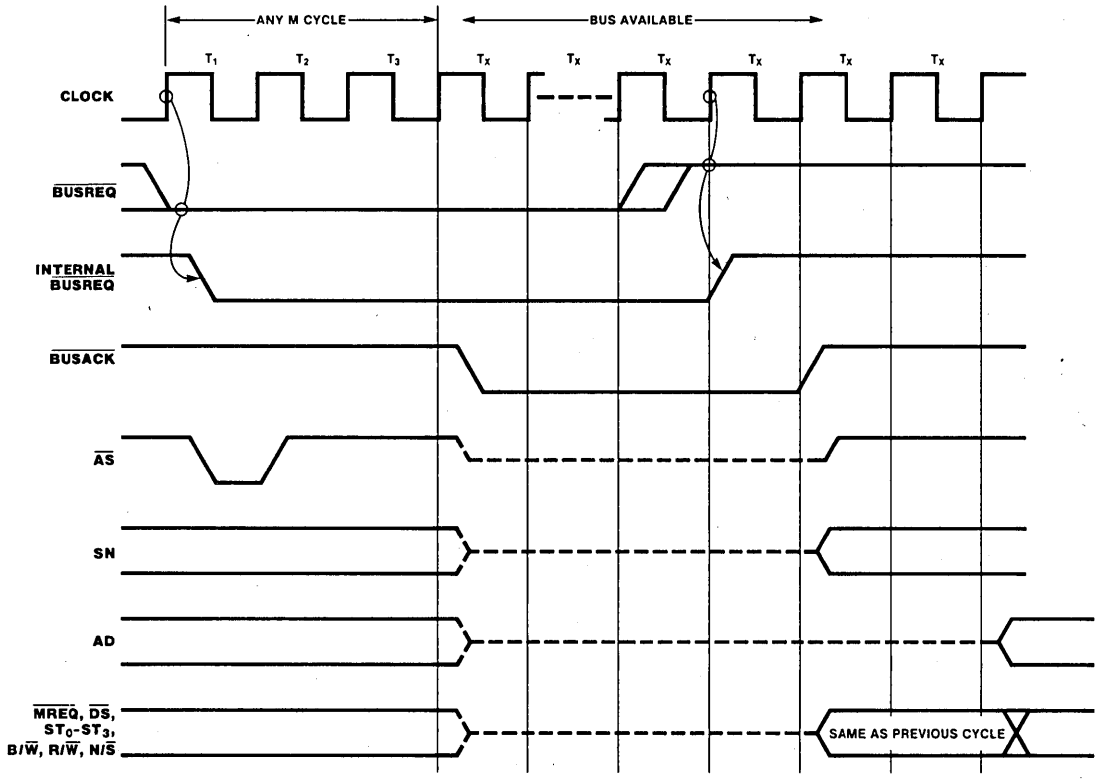


Figure 15. Bus Request/Acknowledge Timing

STOP

The STOP input is sampled by the last falling clock edge immediately preceding any IF_1 cycle (Figure 16) and before the second word of an EPA instruction is fetched. If $\overline{STOP}$ is found Low during the IF_1 cycle, a stream of memory refresh cycles is inserted after T_3, again sampling the $\overline{STOP}$ input on each falling clock edge in the middle of the T_3 states. During the EPA instruction, both EPA instruction words are fetched but any data transfer or subsequent instruction fetch is postponed until $\overline{STOP}$ is sampled High. This refresh operation does not use the refresh prescaler or its divide-by-four clock prescaler; rather, it double-increments the refresh counter every three clock cycles. When $\overline{STOP}$ is found High again, the next refresh cycle is completed, any remaining T states of the IF_1 cycle are then executed, and the CPU continues its operation.

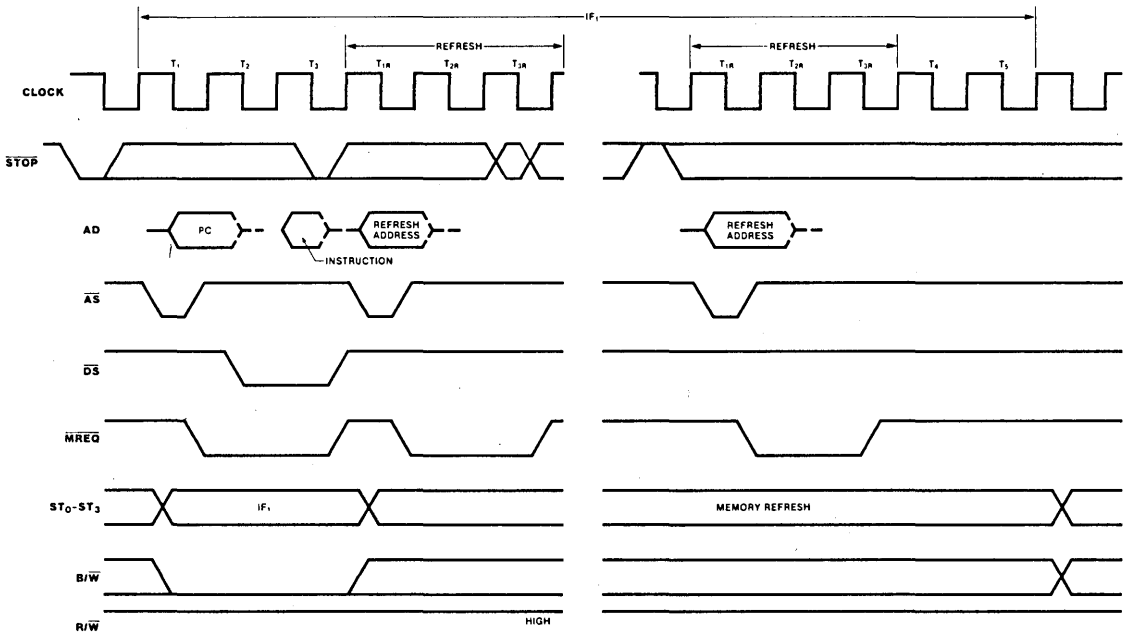


Figure 16. Stop Timing

INTERNAL OPERATION

Certain extended instructions, such as Multiply and Divide, and some special instructions need additional time for the execution of internal operations. In these cases, the CPU goes through a sequence of internal operation machine cycles, each of which is three to eight clock cycles long (Figure 17). This allows fast response to Bus Request and Refresh Request, because bus request or refresh cycles can be inserted at the end of any internal machine cycle.

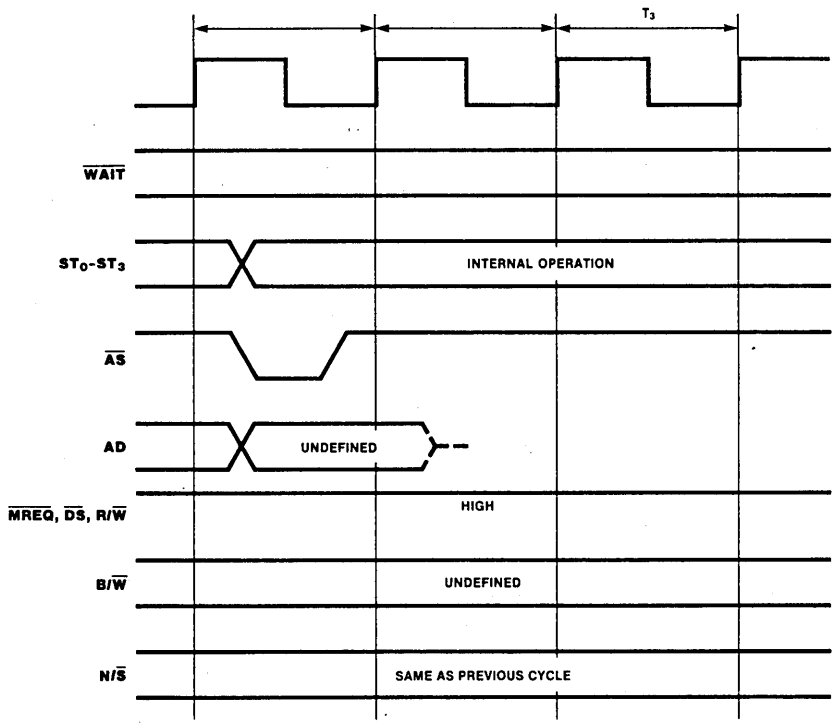


Figure 17. Internal Operation Timing

HALT

A HALT instruction executes an unlimited number of 3-cycle internal operations, interspersed with memory refresh cycles whenever requested. An interrupt, segmentation trap, or reset are the only exits from a HALT instruction.

The CPU samples the $\overline{VI}$, $\overline{NVI}$ and $\overline{NMI}$ inputs at the beginning of every T_3 cycle. If an input is found active during two consecutive samples, the subsequent IF_1 cycle is exercised, but ignored, and the normal interrupt acknowledge cycle is started.

MEMORY REFRESH

When the 6-bit prescaler in the refresh counter has been decremented to zero, a refresh cycle consisting of three T-states is started as soon as possible (that is, after the next IF_1 cycle or Internal Operation cycle).

The 9-bit refresh counter value is put on the low-order side of the address bus (AD_0-AD_8); AD_9-AD_{15} are undefined (Figure 18). Since the memory is word-organized, A_0 is always Low during refresh and the refresh counter is always incremented by two, thus stepping through 256 consecutive refresh addresses on AD_1-AD_8. Unless disabled, the presettable prescaler runs continuously and the delay in starting a refresh cycle is therefore not cumulative.

While the $\overline{STOP}$ input is Low, a continuous stream of memory refresh cycles, each three T-states long, is executed without using the refresh prescaler.

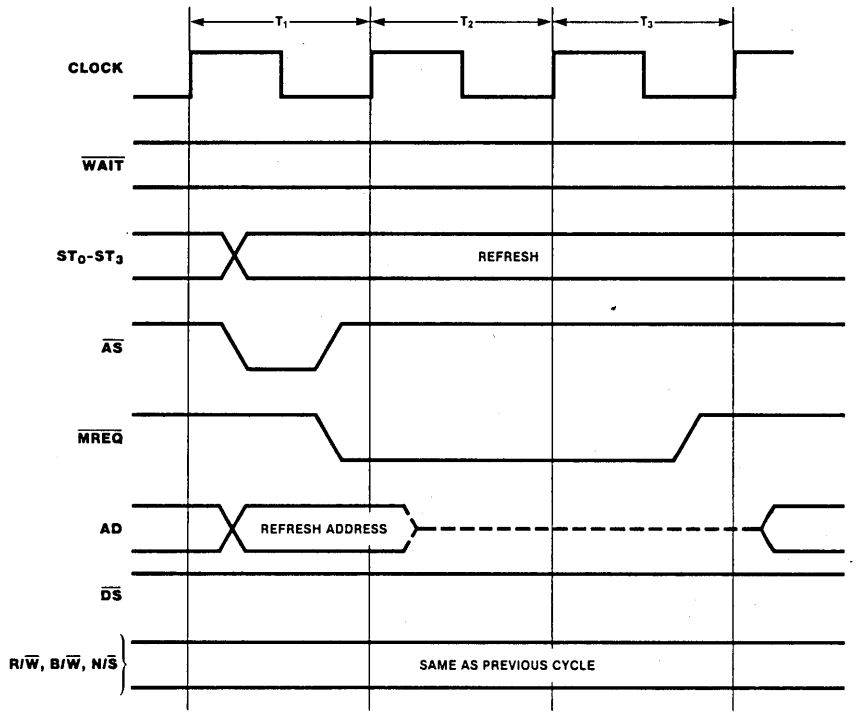


Figure 18. Memory Refresh Timing

RESET

A Low on the $\overline{RESET}$ input causes the following results within five clock cycles (Figure 19):

- AD_0-AD_{15} are 3-stated
- $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, ST_0-ST_3, $\overline{BUSACK}$, and $\overline{MO}$ are forced High
- **SN_0-SN_4 are forced Low**
- Refresh is disabled
- R/$\overline{W}$, B/$\overline{W}$, and N/$\overline{S}$ are not affected

When $\overline{RESET}$ has been High for three clock periods, three consecutive memory read cycles are executed in the system mode. The first cycle reads the flag and control word from location 0002, the next reads the 7-bit program counter segment number from location 0004, the next reads the 16-bit PC offset from location 0006, and the following IF_1 cycle starts the program.

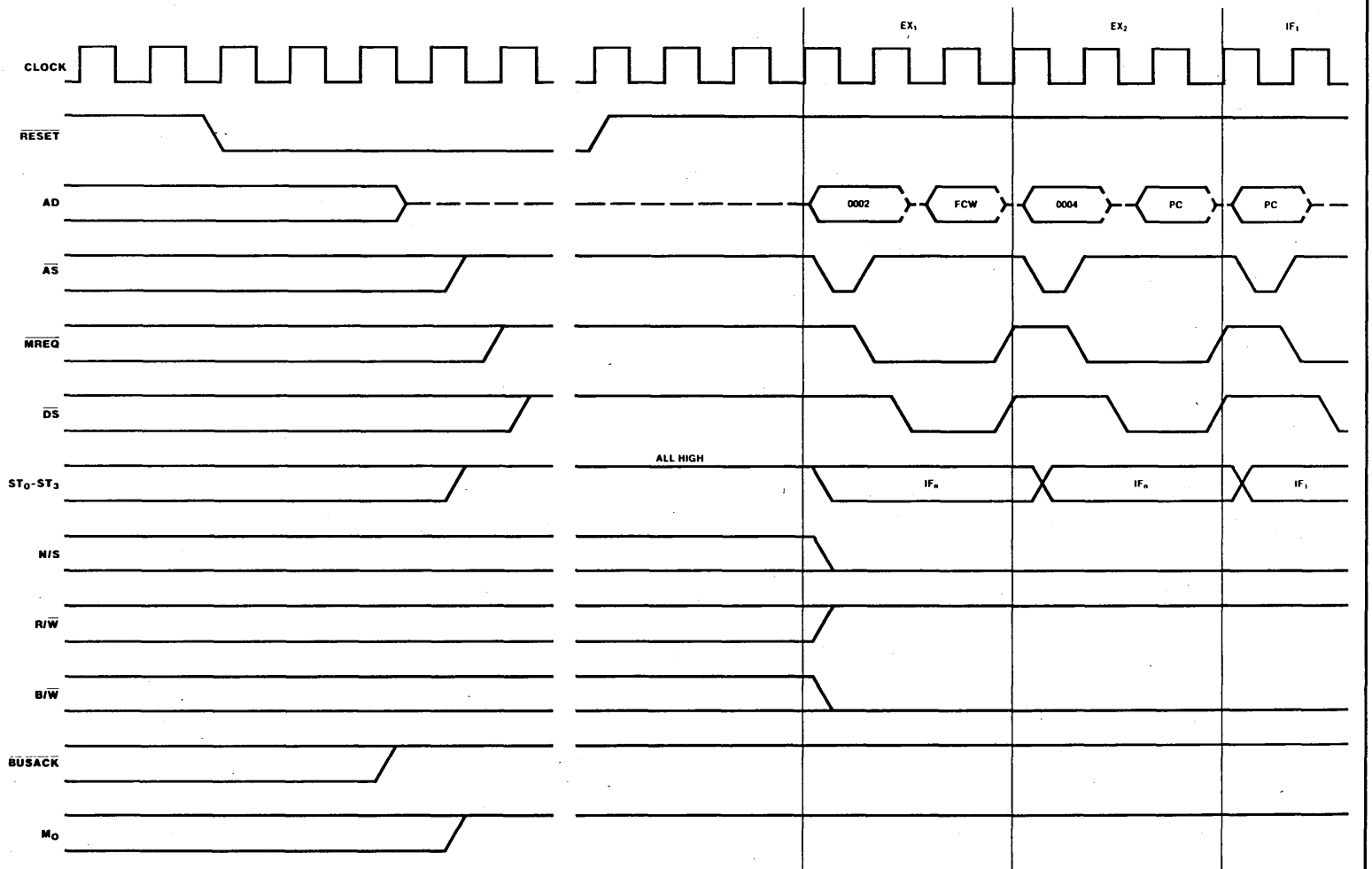
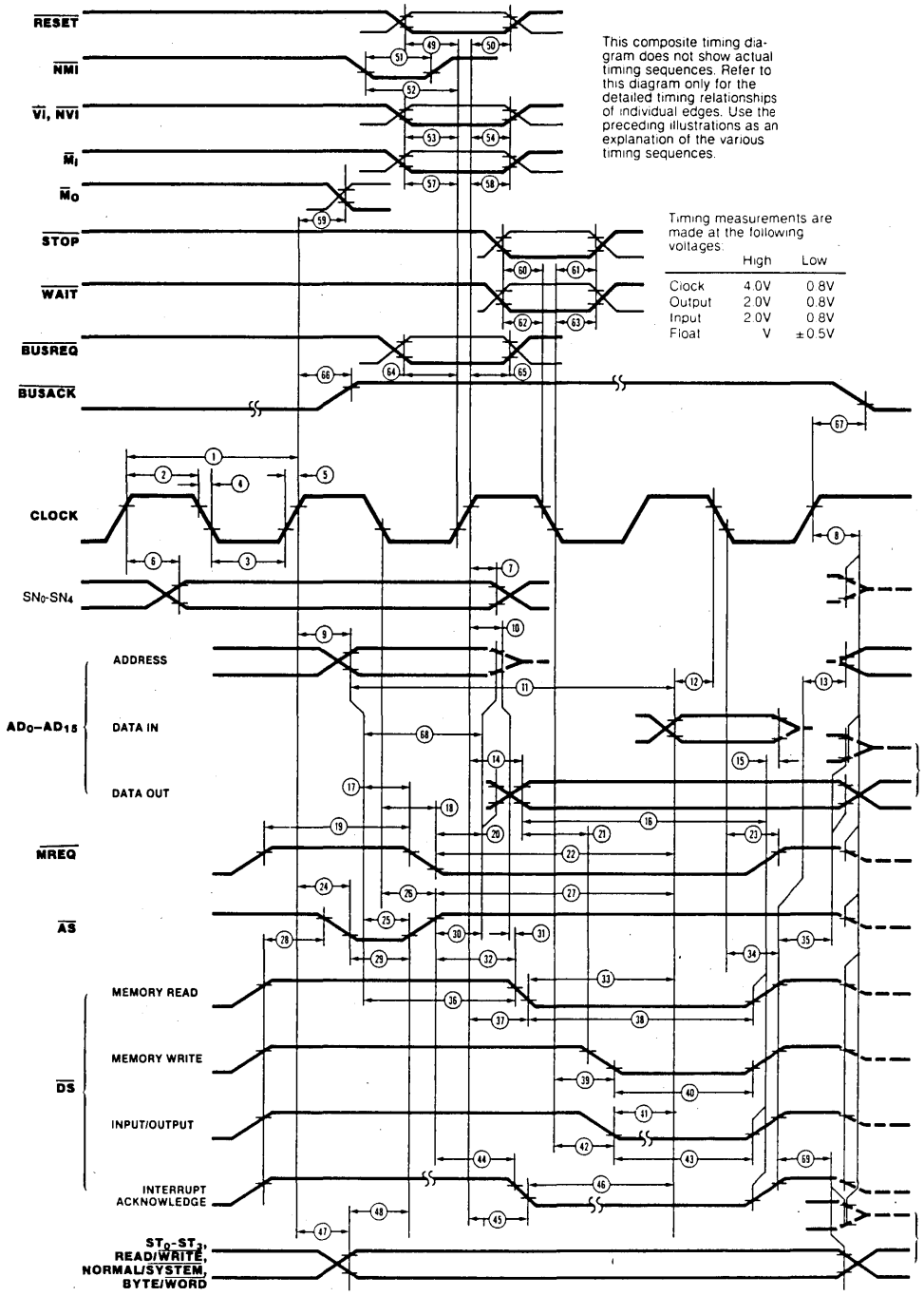


Figure 19. Reset Timing

COMPOSITE AC TIMING DIAGRAM



This composite timing diagram does not show actual timing sequences. Refer to this diagram only for the detailed timing relationships of individual edges. Use the preceding illustrations as an explanation of the various timing sequences.

Timing measurements are made at the following voltages:

	High	Low
Clock	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	V	±0.5V

AC CHARACTERISTICS†

Number	Symbol	Parameter	Z160 6 MHz Min	Z160 6 MHz Max	Z160 10 MHz Min	Z160 10 MHz Max
1	TcC	Clock Cycle Time	165	2000	100	2000
2	TwCh	Clock Width (High)	70	1930	40	1960
3	TwCl	Clock Width (Low)	70	1930	40	1960
4	TfC	Clock Fall Time		10		10
5	TrC	Clock Rise Time		15		10
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pf load)		110		90
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	10		0	
8	TdC(Bz)	Clock ↑ to Bus Float		55		50
9	TdC(A)	Clock ↑ to Address Valid		75		55
10	TdC(Az)	Clock ↑ to Address Float		55		50
11	TdA(DR)	Address Valid to Read Data Required Valid		305*		180*
12	TsDR(C)	Read Data to Clock ↓ Setup time	20		10	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	45*		20*	
14	TdC(DW)	Clock ↑ to Write Data Valid		75		60
15	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0	
16	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	195*		110*	
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	35*		20*	
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		70		50
19	TwMRh	$\overline{MREQ}$ Width (High)	135*		80*	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	35*		20*	
21	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	35*		15*	
22	TdMR(DR)	$\overline{MREQ}$ ↓ to Read Data Required Valid		230*		140*
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		60		50
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		60		45
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	35*		20*	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		80		45
27	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		220*		140*
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	35*		15*	
29	TwAS	$\overline{AS}$ Width (Low)	55*		30*	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	45*		20*	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0	
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	55*		30*	
33	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid		130*		70*
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		65		50
35	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid	45*		25*	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	110*		65*	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		85		65
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	185*		110*	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		80		65
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	110*		75*	

*Clock-cycle time-dependent characteristics. See Footnotes to AC Characteristics.
†Units in nanoseconds (ns).

AC CHARACTERISTICS† (Continued)

Number	Symbol	Parameter	Z160 6 MHz Min	Z160 6 MHz Max	Z160 10 MHz Min	Z160 10 MHz Max
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid		210*		120*
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		90		65
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	255*		160*	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	690*		410*	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		85		70
46	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Required Delay		295*		165*
47	TdC(S)	Clock ↑ to Status Valid Delay		85		65
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	30*		20*	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	70		50	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		0	
51	TwNMI	$\overline{NMI}$ Width (Low)	70		50	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	70		50	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	50		40	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	20		10	
55						
56						
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	140		80	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		0	
59	TdC(MO)	Clock ↑ to $\overline{MO}$ Delay		85		80
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	100		50	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		0	
62	TsW(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	30		20	
63	ThW(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		5	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	80		60	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		5	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		75		65
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		75		65
68	TwA	Address Valid Width	95*		50*	
69	TdDS(S)	$\overline{DS}$ ↑ to STATUS Not Valid	55*		30*	

*Clock-cycle time-dependent characteristics. See Footnotes to AC Characteristics.
†Units in nanoseconds (ns).

FOOTNOTES TO AC CHARACTERISTICS

Number	Symbol	Z160 6 MHz Equation	Z160 10 MHz Equation
11	TdA(DR)	2TcC + TwCh – 95 ns	2TcC + TwCh – 60 ns
13	TdDS(A)	TwCl – 25 ns	TwCl – 20 ns
16	TdDW(DS)	TcC + TwCh – 40 ns	TcC + TwCh – 30 ns
17	TdA(MR)	TwCh – 35 ns	TwCh – 20 ns
19	TwMRh	TcC – 30 ns	TcC – 20 ns
20	TdMR(A)	TwCl – 35 ns	TwCl – 20 ns
21	TdDW(DSW)	TwCh – 35 ns	TwCh – 25 ns
22	TdMR(DR)	2TcC – 100 ns	2TcC – 60 ns
25	TdA(AS)	TwCh – 35 ns	TwCh – 20 ns
27	TdAS(DR)	2TcC – 110 ns	2TcC – 60 ns
28	TdDS(AS)	TwCl – 35 ns	TwCl – 25 ns
29	TwAS	TwCh – 15 ns	TwCh – 10 ns
30	TdAS(A)	TwCl – 25 ns	TwCl – 20 ns
32	TdAS(DSR)	TwCl – 15 ns	TwCl – 10 ns
33	TdDSR(DR)	TcC + TwCh – 105 ns	TcC + TwCh – 70 ns
35	TdDS(DW)	TwCl – 25 ns	TwCl – 15 ns
36	TdA(DSR)	TcC – 55 ns	TcC – 35 ns
38	TwDSR	TcC + TwCh – 50 ns	TcC + TwCh – 30 ns
40	TwDSW	TcC – 55 ns	TcC – 25 ns
41	TdDSl(DR)	2TcC – 120 ns	2TcC – 80 ns
43	TwDS	2TcC – 75 ns	2TcC – 40 ns
44	TdAS(DSA)	4TcC + TwCl – 40 ns	4TcC + TwCl – 30 ns
46	TdDSA(DR)	2TcC + TwCh – 105 ns	2TcC + TwCh – 75 ns
48	TdS(AS)	TwCh – 40 ns	TwCh – 30 ns
68	TwA	TcC – 70 ns	TcC – 50 ns
69	TdDS(s)	TwCl – 15 ns	TwCl – 10 ns

AC Timing Test Conditions

V_{OL} = 0.8V
V_{OH} = 2.0V
V_{IL} = 0.8V
V_{IH} = 2.4V
V_{ILC} = 0.45V
V_{IHC} = V_{CC} – 0.4V

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

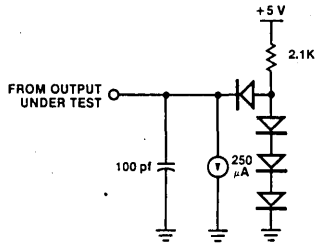
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

- S = 0°C to +70°C, $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- **E = -40°C to +100°C, $+4.75V \leq V_{CC} \leq +5.25V$**



All ac parameters assume a total load capacitance (including parasitic capacitances) or 100 pf max, except for parameter 6 (50 pf max). Timing references between two output signals assume a load difference of 50 pf max.

The Ordering Information section lists package temperature ranges and product numbers.

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	$V_{CC}-0.4$	$V_{CC}+0.3$	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
$V_{IH\ RESET}$	Input High Voltage on $\overline{RESET}$ pin	2.4	$V_{CC}+0.3$	V	
$V_{IH\ NMI}$	Input High Voltage on NMI pin	2.4	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\ mA$
I_{IL}	Input Leakage		±10	µA	$0.4 \leqslant V_{IN} \leqslant +2.4V$
$I_{IL\ SEGT}$	Input Leakage on $\overline{SEGT}$ pin	−100	100	µA	
I_{OL}	Output Leakage		±10	µA	$0.4 \leqslant V_{IN} \leqslant +2.4V$
I_{CC}	V_{CC} Power Supply Current		300	mA	4 MHz and 6 MHz commercial
			400	mA	Extended temperature range
			400	mA	10 MHz speed range

October 1988

Z320™ CPU

FEATURES

- Full 32-bit architecture and implementation
- 4G (billion) bytes of directly addressable memory in each of four address spaces
- Linear or segmented address space
- Virtual memory management integrated with CPU
- On-chip cache memory
- General-purpose register file with sixteen 32-bit registers
- Nine general addressing modes
- Numerous data types include bit, bit field, logical value, signed integer, and string
- Regular use of operations, addressing modes, and data types in instruction set
- System and normal modes of operation with separate stacks
- Sophisticated interrupt and trap handling
- Software is a binary-compatible extension of Z8000® software, totally compatible with the Z80,000®
- Small, low cost 68-pin plastic leaded chip carrier package for surface mount applications
- Hardware is compatible with other Z-BUS® bus components with multiplexed address and data
- Mainframe performance at a micro price

GENERAL DESCRIPTION

The Z320 CPU is an advanced, high-end 32-bit microprocessor that integrates the architecture of a mainframe computer into a single chip. While maintaining full compatibility with Z8000 family software and hardware, the Z320 CPU offers greater power and flexibility in both its architecture and interface capability. Operating systems and compilers are easily developed in the Z320 CPU's high-quality environment, and the hardware interface provides for connection to a wide variety of system configurations.

Adddresses in the Z320 CPU are 32 bits. This allows direct addressing of 4G bytes in each of four address spaces: system-mode data, system-mode instruction, normal-mode data, and normal-mode instruction. The CPU supports three modes of address representation. The 16-bit compact addresses are compatible with Z8000 nonsegmented mode. The 32-bit segmented addresses include both 16-bit offset, which is compatible with Z8000 segmented mode, and 24-bit offset. In addition, a full 32-bit linear address space is provided.

The CPU features a general-purpose register file with sixteen 32-bit registers and nine operand addressing choices for compact representation or for full 32-bit addressing. The instruction set can operate on bit, bit field, logical value, signed integer, unsigned interger, address, string, stack, and packed decimal byte data types. Logical and arithmetic instructions operated on bytes (8 bits), words (16 bits) and longwords (32 bits). The Extended Processing Architecture (EPA) supports highly regular in combining operations, data types, and addressing modes. High-level language compilation is supported with instructions for procedure linkage, array index calculation, and bounds checking. Other instructions provide operating system functions such as system call and control of memory management.

There are two main operating modes, system and normal, supported by separate stacks. User programs operate in normal mode, while sensitive operating system functions are performed in system mode. This protects critical parts of the operating system from user access. In addition, some instructions are priveleged, and execute only in system mode. Memory management functions protect both system memory from user programs, and user memory from other users. Vectored, nonvectored, and nonmaskable interrupts support realtime operating systems.

Memory managmement is fully integrated with the CPU; no external support circuitry is necessary. A paging address translation mechanism is implemented. Registers in the CPU point to address translation tables located in memory; the most recently used table entries are kept in a Translation Lookaside Buffer (TLB) in the CPU. The CPU performs logical to physical address translation and access protection for each memory reference. When a logical memory reference causes a translation or protection violation, the state of the CPU is automatically restored to restart the instruction. I/O ports can be referenced either by dedicated instructions or by the memory management mechanism mapping logical memory addresses to I/O port addresses.

Extensive trapping facilities, such as integer overflow, subrange out of bounds, and subscript out of bounds, catch common run-time errors. Software debuggers can use trace and breakpoint traps. Privileged instruction traps and memory protection violation traps secure the operating system from user programming errors or mischief. The overflow stack allows recovery from otherwise fatal errors.

The CPU has full 32-bit internal address and data paths. Externally, 32 pins time-multiplex the address and data. The interface is compatible with the complete line of Z-BUS peripherals. The hardware interface features 16-bit or 32-bit memory data path and programmable wait states. Burst transfers and an on-chip cache for instructions and data help develop high-performance systems. The interface supports multiprocessing configurations with interlocked memory references and two types of bus request protocols. The system designer can tailor the Z320 based system to cost and performance needs.

In summary, the Z320 CPU meets and surpasses the requirements of medium and high-end microprocessor systems for the 1980s. Software program development is easily accomplished with the CPU's sophisticated architecture. The highly pipelined design, on-chip cache, and external interface support systems ranging from dedicated controllers to mainframe computers. While Zilog continues to develop support for the Z320 CPU, Z8000 peripherals and development software are fully compatible with this latest in Zilog's line of high-performance microprocessors.

REGISTERS

The Z320 CPU is a register-oriented processor offering sixteen 32-bit general-purpose registers, a 32-bit Program Counter (PC), a 16-bit Flag and Control Word (FCW), and nine other special-purpose registers.

The general-purpose register file (Figure 1) contains 64 bytes of storage. The first 16 bytes (RL0,RH0,...,RL7,RH7) can be used as accumulators for byte data. The first 16 words (R0,R1,...,R15) can be used as accumulators for word data, as index registers (except R0), or for memory addresses in compact mode (except R0). Any longword register (RR0,RR2,...,RR30) can be used as an accumulator for longword data, an index register in linear or segmented mode (except RR0), or for memory addresses in linear or segmented mode (except RR0). Quadword registers (RQ0,RQ4,...,RQ28) can be used as accumulators for Multiply, Divide, and Extend Sign instructions. This unique register organization allows bytes and words of data to be manipulated conveniently while leaving most of the register file free to hold addresses, counters, and any other data.

Two registers are dedicated to the Stack Pointer (SP) and Frame Pointer (FP) used by Call, Enter, Exit, and Return and R14 the Frame Pointer. In linear or segmented mode, RR14 is the Stack Pointer and RR12 is the Frame Pointer.

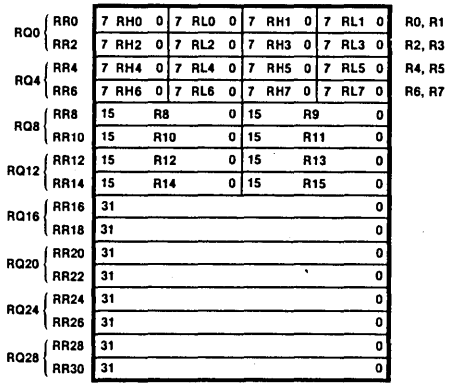


Figure 1. General-Purpose Register File

CACHE

The CPU implements a cache mechanism to keep on-chip copies of the most recently referenced memory locations (Figure 12). The CPU examines the cache on memory fetches to determine if the addressed data are located in the cache. If the information is in the cache (a hit), then the CPU fetches from the cache, and no transaction is necessary on the external interface. If the information is not in the cache (a miss), then the CPU performs a memory read transaction to fetch the missing information.

The cache stores data in blocks of 16 bytes. Each data word in the cache has an associated validity bit to indicate whether or not the word is a valid copy of the corresponding main memory location. The cache contains 16 blocks, providing 256 bytes of storage.

The cache is fully associative, so that a block currently needed and missing in the cache can replace any block in the cache. Moreover, when a block miss occurs, the least recently used (LRU) block in the cache is replaced. When a cache miss occurs on an instruction fetch, the CPU fetches the missing instruction from memory and prefetches the following words in the block using a burst transaction. When a cache miss occurs on an operand fetch, the CPU fetches the missing data from memory. (The CPU uses burst transactions only for fetching operands when more than one data transfer is necessary: longword operands on a 16-bit bus, unaligned operands, string instructions, Load Multiple instructions, and loading Program Status.)

On store references, the data is written to memory (store through), and if the reference hits in the cache, the data is also written to the cache. If the store reference misses in the cache, the cache is unaffected.

Software has some control over the cache. The cache can be selectively enabled for instruction and data references by bits CI and CD in the SCCL control register. The memory management mechanism allows cacheing to be inhibited for individual pages. The Pcache instruction can be used to invalidate all information in the cache.

The cache has an option, controlled by bit CR in SCCL, to inhibit block replacement on a miss. This option can be used to lock fixed locations into the cache for fast, onchip access. To do this, the cache is first enabled for block replacement of data references only. Selected blocks are read into the cache. The block replacement algorithm is then disabled, while the cache is enabled for instruction and data references.

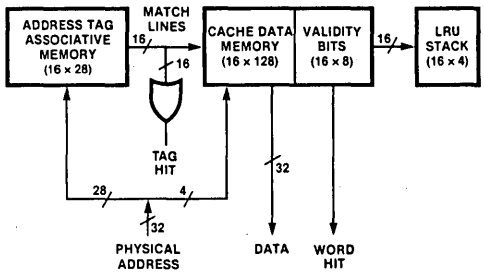


Figure 2. Cache Organization

PIN DESCRIPTIONS

The CPU has 58 signal lines and additional power supply connections. Pin functions are shown in Figure 3a, b, c.

AD_0-AD_{31}. *Address/Data (Bidirectional, active High, 3-state).* These 32 lines are time-multiplexed to transfer address and data. At the beginning of each transaction the lines are driven with the 32-bit address. After the address has been driven, the lines are used to transfer one or more bytes, words, or longwords of data.

$\overline{AS}$. *Address Strobe (Output, active Low, 3-state).* The falling edge of $\overline{AS}$ indicates the beginning of a transaction and shows that the address and ST_0-ST_3, are valid. R/$\overline{W}$, BL/$\overline{W}$, BW/$\overline{L}$, and $\overline{BRST}$ are valid on the rising edge of $\overline{AS}$.

BL/$\overline{W}$; BW/$\overline{L}$. *Byte, Longword/Word; Byte, Word/Longword (Output, 3-state).* These two lines specify the data transfer size.

BL/$\overline{W}$	BW/$\overline{L}$	Size
High	High	Byte
Low	High	Word
High	Low	Longword
Low	Low	Reserved

$\overline{BRST}$. *Burst (Output, active Low, 3-state).* A Low on this line indicates that the CPU is performing a burst transfer; i.e, multiple Data Strobes following a single Address Strobe.

$\overline{BRSTA}$. *Burst Acknowledge (Input, active Low).* A Low on this line indicates that the responding device can support burst transfers.

$\overline{BUSACK}$. *Bus Acknowledge (Output, active Low).* A Low on this line indicates that the CPU has relinquished control of the local bus in response to a bus request.

$\overline{BUSREQ}$. *Bus Request (Input, active Low).* A Low on this line indicates that a bus requestor has obtained or is trying to obtain control of the local bus.

CLK. *Clock (Input).* This is the clock used to generate all CPU timing.

$\overline{DS}$. *Data Strobe (Output, active Low, 3-state).* DS is used for timing data transfers.

EPUABORT. ***EPU Abort (Output, active High).*** **A High on this line indicates that the CPU is aborting execution of an EPA instruction, typically because an Address Translation trap has occurred.**

$\overline{EPUBSY}$. *EPU Busy (Input, active Low).* A Low on this line indicates that an EPU is busy. This line is used to synchronize the operation of the CPU with an EPU during execution of an EPA instruction.

$\overline{GACK}$. *Global Acknowledge (Input, active Low).* A Low on this line indicates the CPU has been granted control of a global bus.

$\overline{GREQ}$. *Global Request (Output, active Low, 3-state).* A Low on this line indicates the CPU has obtained or is trying to obtain control of a global bus.

$\overline{IE}$. *Input Enable (Output, active Low, 3-state).* A Low on this line can be used to enable buffers on the AD lines to drive toward the CPU.

$\overline{NMI}$. *Non-Maskable Interrupt (Input, Edge activated).* A High-to-Low transition on this line requests a nonmaskable interrupt.

$\overline{NVI}$. *Non-Vectored Interrupt (Input, active Low).* A Low on this line requests a non-vectored interrupt.

$\overline{OE}$. *Output Enable (Output, active Low, 3-state).* A Low on this line can be used to enable buffers on the AD lines to drive away from the CPU.

$\overline{RESET}$. *Reset (Input, active Low).* A Low on this line resets the CPU.

RSP_0-RSP_1. *Response (Input).* These lines encode the response to transactions initiated by the CPU. Note that RSP_0 and RSP_1 can be connected together for Z-BUS $\overline{WAIT}$ timing.

RSP_0	RSP_1	Response
High	High	Ready
Low	High	Bus Error
High	Low	Bus Retry
Low	Low	Wait

R/$\overline{W}$. *Read/Write (Output, Low = Write, 3-state).* This signal indicates the direction of data transfer.

ST_0-ST_3. *Status (Output, active High, 3-state).* These lines specify the kind of transaction occurring on the bus. (See Table 5.)

VBB. *Substrate Bias Generator* (Output, for internal biasing only).

$\overline{VI}$. *Vectored Interrupt (Input, active Low).* A Low on this line requests a vectored interrupt.

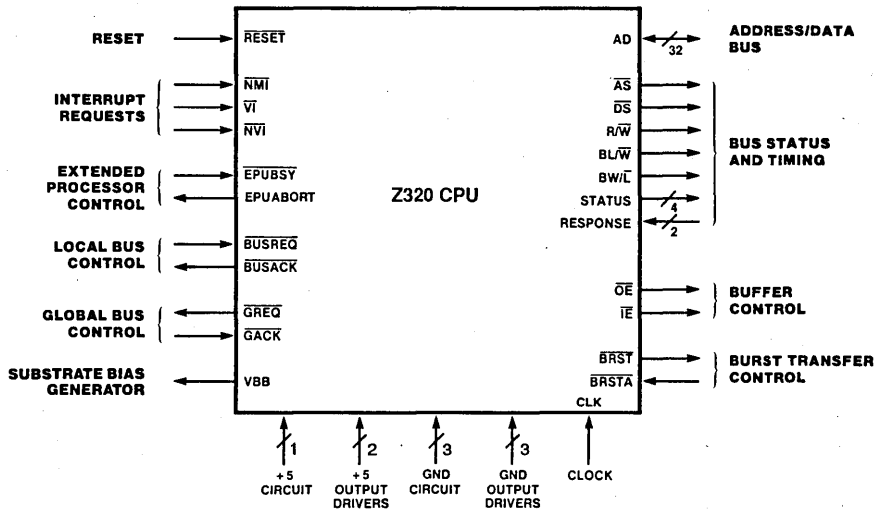


Figure 3a. Pin Functions

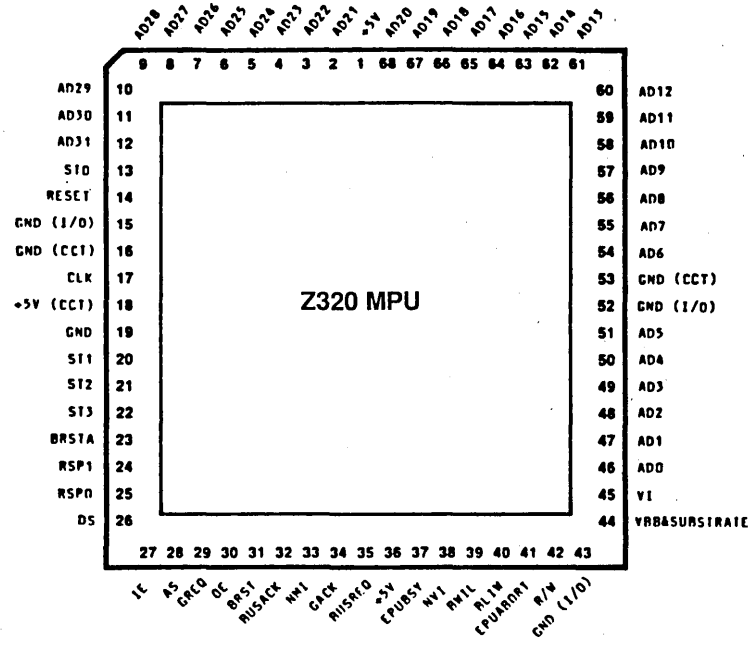


Figure 3b.

PIN ASSIGNMENT FOR Z320 IN 68-PIN PLCC PACKAGE

PIN #	FUNCTION	PIN #	FUNCTION	PIN #	FUNCTION	PIN #	FUNCTION
10	AD29	27	IE	44	VBB&SUBSTRATE	61	AD13
11	AD30	28	AS	45	VI	62	AD14
12	AD31	29	GREQ	46	AD0	63	AD15
13	SIO	30	OE	47	AD1	64	AD16
14	RESET	31	BRST	48	AD2	65	AD17
15	GND (I/O)	32	BUSACK	49	AD3	66	AD18
16	GND (CCT)	33	NMI	50	AD4	67	AD19
17	CLK	34	GACK	51	AD5	68	AD20
18	+5V (CCT)	35	BUSREQ	52	GND (I/O)	1	+5V
19	GND	36	+5V	53	GND (CCT)	2	AD21
20	SI1	37	EPUBSY	54	AD6	3	AD22
21	SI2	38	NVI	55	AD7	4	AD23
22	SI3	39	BWIL	56	AD8	5	AD24
23	BRSTA	40	BLIW	57	AD9	6	AD25
24	RSP1	41	EPUABORT	58	AD10	7	AD26
25	RSP0	42	R/W	59	AD11	8	AD27
26	DS	43	GND (I/O)	60	AD12	9	AD28

0-87

Figure 3c.

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs
with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient
Temperature. . . . . .See ordering information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only: operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

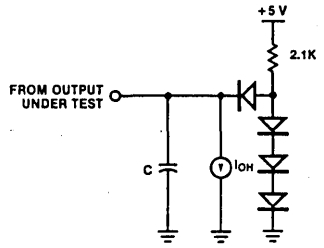
STANDARD TEST CONDITIONS

The DC characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

■ S = 0°C to +70°C, $+4.75V \leqslant V_{CC} \leqslant +5.25V$

All ac parameters assume a total load capacitance (C), including parasitic capacitances, of 100 pf max.



DC CHARACTERISTICS

Symbol	Parameter	Min	Typ	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.0		V_{CC} + 0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3		0.6	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0		V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	−0.3		0.8	V	
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −500 μA
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = +4.0 mA
I_{IL}	Input Leakage			±10	μA	$0.4 \leqslant V_{IN} \leqslant +2.4V$
I_{CC}	V_{CC} Supply Current			700	mA	8 MHz Clock Frequency
	VCC Supply Current			800	mA	10 MHz Clock Frequency

AC CHARACTERISTICS*

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max
1	TcC	PClock Cycle Time	125	500	100	500
2	TwCh	Width (High)	52		40	
3	TwCl	Width (Low)	52		40	
4	TfC	PClock Fall Time		10		10
5	TrC	PClock Rise Time		10		10
6	TdC(Bz)	PClock ↑ to Bus Float		47		47
7	TsA(C)	Address Valid to PClock ↑ Setup Time	0		0	
8	TdC(Az)	PClock ↑ to Address Float		40		40
9	TdA(DR)	Address Valid to Read Data Required Valid (single memory read timing with one wait, without wait is 280)	605†		480†	
10	TsDR(C)	Read Data to PClock ↑ Setup Time	20			
11	TdDS(A)	$\overline{DS}$ ↑ to Address Active	95†		70†	
12	TdC(DW)	PClock ↑ to Write Data Valid		65		65
13	ThDR(C)	Read Data Valid to PClock ↑ Hold Time	5		5	
14	ThDR(DS)	Read Data Valid to $\overline{DS}$ ↑ Hold Time	0		0	
15	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	330†		255†	
16	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	80†		55†	
17	TdC(ASf)	PClock ↑ to $\overline{AS}$ ↓ Delay		50		50
18	TsS(C)	Status Valid to PClock ↑ Setup Time	0		0	
19	TdC(ASr)	PClock ↑ to $\overline{AS}$ ↑ Delay		50		50
20	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid (single memory read timing with one wait; without wait is 120)	420†		320†	
21	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	335†		260†	
22	TwAS	$\overline{AS}$ Width (Low)	110†		85†	
23	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	95†		70†	
24	TdAz(DSR)	Address Float to $\overline{DS}$ ↓ (Read) Delay	0		0	
25	TdBz(BUS)	Bus Float to $\overline{BUSACK}$ ↓ Delay	9		9	
26	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Read) Delay	95†		70†	
27	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid (single memory read timing with one wait, without wait is 20)	295†		220†	
28	TdC(DS)	PClock ↑ to $\overline{DS}$ ↑ Delay		50		50
29	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid	95†		70†	
31	TdC(DSR)	PClock ↑ to $\overline{DS}$ (Read) ↓ Delay		50		50
32	TwDSR	$\overline{DS}$ (Read) Width (Low) (single memory read timing with one wait, without wait is 85)	360†		285†	
33	TdC(DSW)	PClock ↑ to $\overline{DS}$ (Write) ↓ Delay		50		50
34	TwDSW	$\overline{DS}$ (Write) Width (Low)	235†		185†	
35	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid	95†		120†	
36	TdC(DSI)	PClock ↑ to $\overline{DS}$ (I/O) ↓ Delay		50		50

*Units in nanoseconds. See Footnotes to AC Characteristics.
†Clock-cycle time-dependent characteristics.

AC CHARACTERISTICS* (Continued)

			8 MHz		10 MHz	
Number	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**
37	TwDSI	$\overline{DS}$ (I/O) Width (Low)	235†		185†	
38	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay	95†		70†	
39	TdC(DSA)	PClock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		50		50
40	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Required Valid	170†		120†	
41	TdC(S)	PClock ↑ to Status Valid Delay		60		60
42	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	85†		60†	
43	TsR(C)	$\overline{RESET}$ to PClock ↓ Setup Time	20		20	
44	ThR(C)	$\overline{RESET}$ to PClock ↓ Hold Time	25		25	
45	TwNMIl	$\overline{NMI}$ Width (Low)	365†		290†	
46	TwNMIh	$\overline{NMI}$ Width (High)	240†		190†	
47	TsNMI(C)	$\overline{NMI}$ ↓ to PClock ↑ Setup Time	40		40	
48	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to PClock ↑ Setup Time	40		40	
49	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to PClock ↑ Hold Time	20		20	
50	TwVI	$\overline{VI}$, $\overline{NVI}$ Width (Low)	365†		290†	
51	TsBREQ(C)	$\overline{BUSREQ}$ Change to PClock ↑ Setup Time	40		40	
52	TwBREQ	$\overline{BUSREQ}$ Width (Low)	365†		290†	
53	ThBREQ(C)	$\overline{BUSREQ}$ to PClock ↑ Hold Time	20		20	
54	TdC(BACKr)	PClock ↑ to $\overline{BUSACK}$ ↑ Delay		65		65
55	TdC(BACKf)	PClock ↑ to $\overline{BUSACK}$ ↓ Delay		65		65
57	TdC(IEr)	PClock ↑ to $\overline{IE}$ ↑ Delay		65		65
58	TdC(IEf)	PClock ↑ to $\overline{IE}$ ↓ Delay		65		65
59	TsBRSTA(C)	$\overline{BRSTA}$ to PClock ↑ Setup Time	25		25	
60	TsEPUBSY(C)	$\overline{EPUBSY}$ to PClock ↑ Setup Time	20		20	
61	ThBRSTA(C)	$\overline{BRSTA}$ to PClock ↑ Hold Time	5		5	
62	ThEPUBSY(C)	$\overline{EPUBSY}$ to PClock ↑ Hold Time	5		5	
63	TsRSP(C)	RSP Change to PClock ↑ Setup Time	20		20	
64	ThRSP(C)	RSP to PClock ↑ Hold Time	5		5	
65	TdIE(OE)	$\overline{OE}$ Change to $\overline{IE}$ Change Delay	295†		270†	
66	TwGACK	$\overline{GACK}$ Width (Low)	365†		290†	
67	TsGACK(C)	$\overline{GACK}$ Change to PClock ↑ Setup time	40		40	
68	ThGACK(C)	$\overline{GACK}$ to PClock ↑ Hold Time	20		20	
69	TdC(OEf)	PClock ↑ to $\overline{OE}$ ↓ Delay		50		50
70	TdC(OEr)	PClock ↑ to $\overline{OE}$ ↑ Delay		50		50
71	TdC(BRSTf)	PClock ↑ to $\overline{BRST}$ ↓ Delay		65		65
72	TdC(BRSTr)	PClock ↑ to $\overline{BRST}$ ↑ Delay		65		65
73	TdC(GREQf)	PClock ↑ to $\overline{GREQ}$ ↓ Delay		50		50
74	TdC(GREQr)	PClock ↑ to $\overline{GREQ}$ ↑ Delay		50		50

*Units in nanoseconds. See Footnotes to AC Characteristics.
†Clock-cycle time-dependent characteristics.

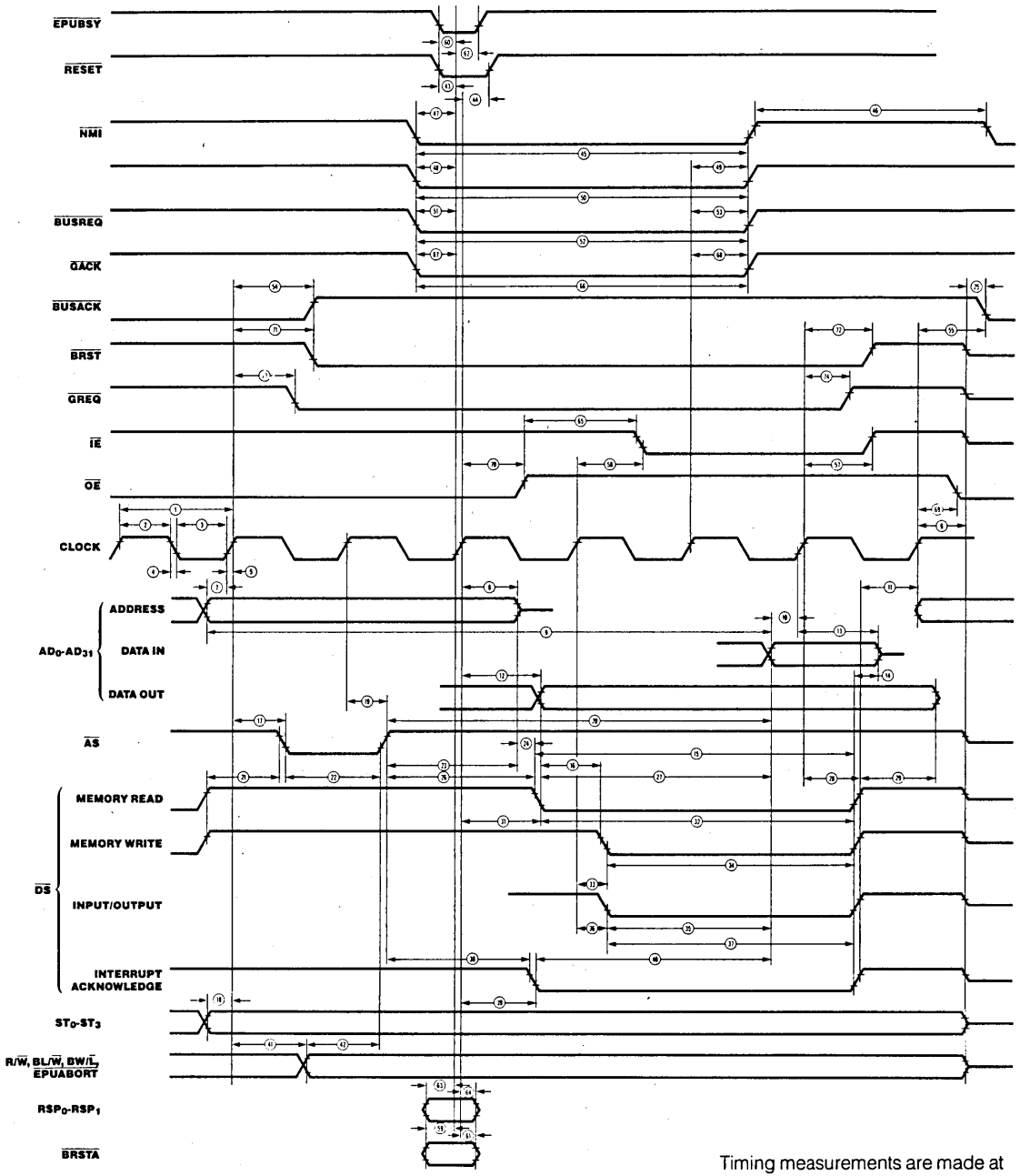
FOOTNOTES TO AC CHARACTERISTICS

No.	Symbol	Equation
9	TdA(DR)	5TcC – TsDR(C)
11	TdDS(A)	TcC – TdC(DS) + TdC(Az) – 20ns
15	TdDW(DS)	3TcC – TdC(DW) + TdC(DS) – 30ns
16	TdDW(DSW)	TcC – TdC(DW) + TdC(DSW) – 30ns
20	TdAS(DR)	4TcC – TdC(ASr) – TsDR(C) – TrC
21	TdDS(AS)	3TcC – TdC(DS) + TdC(ASf) – 40ns
22	TwAS	TcC – 15ns
23	TdAS(A)	TcC – TdC(ASr) + TdC(Az) – 20ns
26	TdAS(DSR)	TcC – TdC(ASr) + TdC(DSR) – 30ns
27	TdDSR(DR)	3TcC – TdC(DSR) – TsDR(C) – TrC
29	TdDS(DW)	TcC – TdC(DS) + TdC(Az) – 20ns
32	TwDSR	3TcC – 15ns
34	TwDSW	2TcC – 15ns
35	TdDSI(DR)	2TcC – TdC(DSI) – TsDR(C) – TrC
37	TwDSI	2TcC – 15ns
38	TdAS(DSA)	TcC – TdC(ASr) + TdC(DSA) – 30ns
40	TdDSA(DR)	2TcC – TdC(DSA) – TsDR(C) – TrC
42	TdS(AS)	TcC – TdC(S) + TdC(ASr) – 30ns
45	TwNMIl	3TcC – 10ns
46	TwNMIh	2TcC – 10ns
50	TwVI	3TcC – 10ns
52	TwBREQ	3TcC – 10ns
65	TdIE(OE)	TcC – TdC(OEr) + TdC(IEf) – 45ns
66	TwGACK	3TcC – 10ns

AC Timing Test Conditions

V_{OL} = 0.8V
V_{OH} = 2.0V
V_{IL} = 0.8V
V_{IH} = 2.4V
V_{ILC} = 0.6V
V_{IHC} = 3.0V

AC TIMING



Timing measurements are made at the following voltages.

	High	Low
Clock	3.0V	0.6V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	ΔV	±0.5V

October 1988

Z328 In Circuit Emulator

FEATURES:

- Z320 Support Chip
- Full Z320 instruction Set
- Real Time Execution
- Single Step Operation
- Break Request and Break Acknowledge Signals
- Easy Solution to System Debugging
- 84 Pin Ceramic Grid Array Package

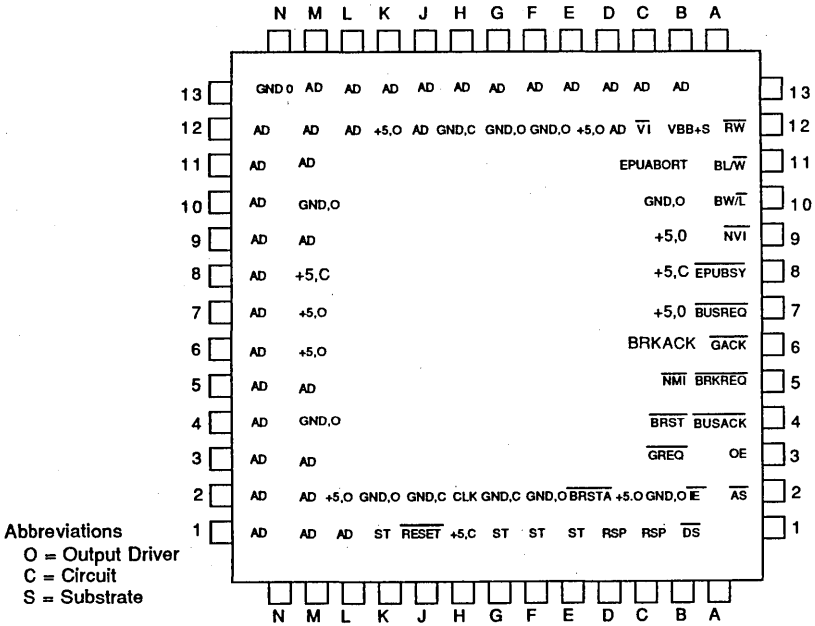
GENERAL DESCRIPTION:

Emulator chip is support chip that gives users the hardware control of the internal instruction execution microcycles performed by the pipelined structure.

Access to breakpoint request for breakpoint acknowledge signals of the Z80328 chip allows real time execution with external control of the breakpoint trap feature.

Continual exertion of a breakpoint request will result in single-step operation where access to read or modify CPU registers for memory in the CPU and the target offers the ability to trace.

The Z328 is an easy solution to system debugging. The Z328 is available in an 84-pin ceramic grid-array package.



Abbreviations
O = Output Driver
C = Circuit
S = Substrate

84-pin Pin Grid Array (PGA), Pin Assignments, Preliminary View of Pin Side

October 1988

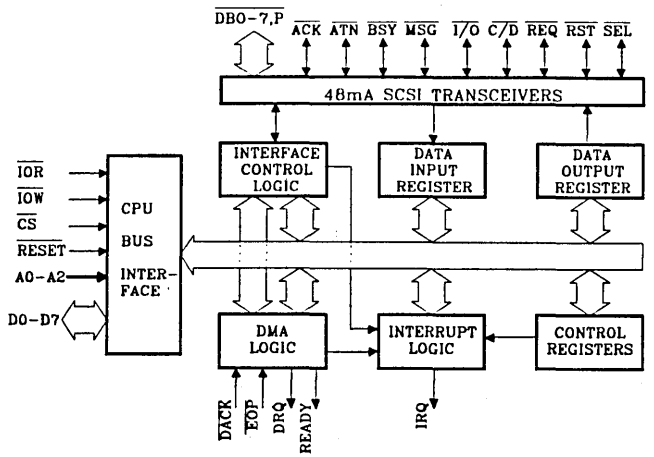
Z5380 SCSI
Small Computer System

FEATURES:

- Compatible 5380 Pinout
- CMOS - Typical Icc 2.5 mA
- Asynchronous Interface, Supports 1.5 MB/s
- Direct SCSI Bus Interface with On-Board 48 mA Drivers
- Supports Target and Initiator Roles
- Arbitration Support
- DMA or Programmed I/O Data Transfers
- Supports Normal or Block Mode DMA
- Memory or I/O Mapped CPU Interface

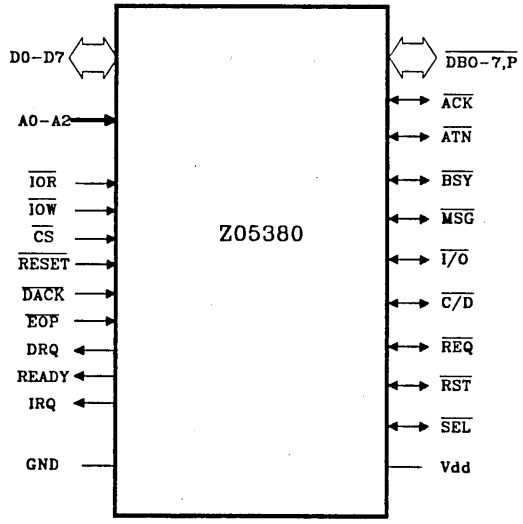
GENERAL DESCRIPTION:

Zilog's Z5380 SCSI (Small Computer System Interface) controller, is a 40 pin DIP or 44 pin PLCC CMOS device. It was designed to implement the SCSI protocol as defined by the ANSI X3T9.2 Committee, and is a plug-in replacement of the industry standard - the NMOS 5380. The Z5380 is capable of operating both as a Target and an Initiator. This enables the Z5380 to find its use in Bus Host Adaptors, Formatters, and Host Port designs. Special high-current open-drain outputs enable it to directly interface to, and drive, the SCSI bus. These drivers are capable of sinking 48 mA at 0.5V. The Z5380 has the necessary interface hook-ups so the system CPU can communicate with it like with any other peripheral device. The CPU can read from, or write to the SCSI registers which may be addressed as standard or memory-mapped I/O's. The Z5380 increases the system performance by minimizing the CPU intervention in DMA operations which the Z5380 controls. The CPU will be interrupted by the Z5380 when it detects a bus condition that requires that attention. It also supports arbitration and reselection. The Z5380 has the proper hand-shake signals to support normal and block mode DMA operations with most of the popular DMA controllers available.

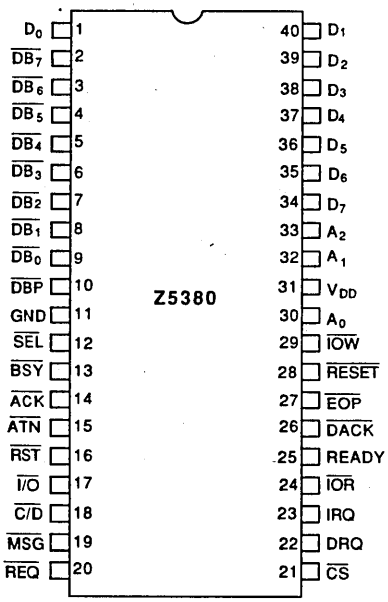


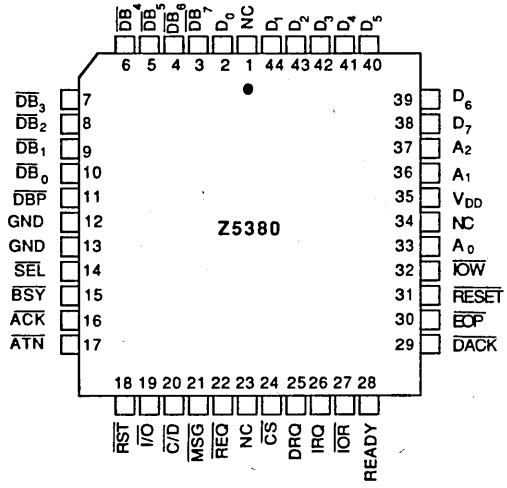
BLOCK DIAGRAM

LOGIC SYMBOL



CONNECTION DIAGRAMS





TOP VIEW

PIN DESCRIPTION

Microprocessor Bus

A0-A2. *Address Lines* (Input). Address lines are used with CS, IOR or IOW to address all internal registers.

$\overline{CS}$. *$\overline{Chip\ Select}$* (Input, Active LOW). CS in conjunction with $\overline{IOR}$ or $\overline{IOW}$, enables the internal register selected by A_0-A_2, to be read from or written to.

$\overline{DACK}$. *$\overline{DMA\ Acknowledge}$* (Input, Active LOW). DACK resets DRQ and selects the data register for input or output data transfers. $\overline{DACK}$ is used by DMA controller instead of $\overline{CS}$.

DRQ. *DMA Request* (Output Active High). DRQ indicates that the data register is ready to be read or written. DRQ is asserted only if DMA mode is set in the Command Register. DRQ is cleared by $\overline{DACK}$.

D_0-D_7. *Data Lines* (Bidirectional; Three-State, Active HIGH). Bidirectional microprocessor data bus lines.

$\overline{EOP}$. $\overline{End\ of\ Process}$ (Input, Active LOW). EOP is used to terminate a DMA transfer. If asserted during a DMA cycle, the current byte will be transferred, but no additional bytes will be requested.

$\overline{IOR}$. *$\overline{I/O\ Read}$* (Input, Active LOW). $\overline{IOR}$ is used in conjunction with $\overline{CS}$ and A_0-A_2. to write an internal register It also selects the input Data Register when used with $\overline{DACK}$.

$\overline{IOW}$. $\overline{I/O\ Write}$ (Input, Active LOW). $\overline{IOW}$ is used in conjunction with $\overline{CS}$ and A0-A2, to write an internal register. It also selects the Output Data Register when used with $\overline{DACK}$.

IRQ. *Interrupt Request* (Output, Active HIGH). IRQ alerts a microprocessor of an error condition or an event completion.

READY. *Ready* (Output Active HIGH). READY is used to control the speed of Block mode DMA transfers. This signal goes active to indicate the chip is ready to send/receive data and remains FALSE after a transfer until the last byte is sent or until the DMA MODE bit is reset.

$\overline{RESET}$. *$\overline{Reset}$* (Input, Active LOW). $\overline{RESET}$ clears all registers. It has no effect upon the SCSI $\overline{RST}$ signal.

Power Signals

VDD. + 5-Volt Power Supply

GND. Ground

SCSI Bus

The following signals are all bidirectional, active-LOW, open-drain, with 48-mA sink capability. All pins interface directly with the SCSI Bus.

$\overline{ACK}$. *$\overline{Acknowledge}$* (Bidirectional; Open Drain, Active LOW). Driven by an Initiator, $\overline{ACK}$ indicates an acknowledgement for a $\overline{REQ}$/$\overline{ACK}$ data-transfer handshake. In the Target role, $\overline{ACK}$ is received as a response to the $\overline{REQ}$ signal.

$\overline{ATN}$. *$\overline{Attention}$* (Bidirectional; Open Drain, Active LOW). Driven by an Initiator, received by the target. $\overline{ATN}$ indicates an Attention condition.

$\overline{BSY}$. *$\overline{Busy}$* (Bidirectional; Open Drain, Active LOW). This signal indicates that the SCSI Bus is being used and can be driven by both the Initiator and the Target device.

$\overline{C/D}$. *$\overline{Control/Data}$* (Bidirectional; Open Drain, Active LOW). Driven by the Target and received by the Initiator. $\overline{C/D}$ indicates whether Control or Data information is on the Data Bus.

$\overline{I/O}$. *$\overline{Input/Output}$* (Bidirectional; Open Drain, Active LOW). $\overline{I/O}$ is a signal driven by a Target which controls the direction of data movement on the SCSI Bus. TRUE indicates input to the Initiator. This signal is also used to distinguish between Selection and Reselection phases.

$\overline{MSG}$. *$\overline{Message}$* (Bidirectional; Open Drain, Active LOW). $\overline{MSG}$ is a signal driven by the Target during the Message phase. This signal is received by the Initiator.

$\overline{REQ}$. *$\overline{Request}$* (Bidirectional; Open Drain, Active LOW). Driven by a Target and received by the initator, $\overline{REQ}$ indicates a request for a $\overline{REQ}$/$\overline{ACK}$ data-transfer handshake.

$\overline{RST}$. *SCSI Bus RESET* (Bidirectional; Open Drain, Active LOW). The $\overline{RST}$ signal indicates a SCSI Bus RESET condition.

$\overline{DB_0}$-$\overline{DB_7}$,$\overline{DBP}$. *$\overline{Data\ Bits\ Parity\ Bit}$* (Bidirectional; Open Drain, Active LOW). These eight data bits ($\overline{DB0}$-$\overline{DB7}$), plus a parity bit (DBP) form the Data Bus. $\overline{DB7}$ is the most significant bit (MSB) and has the highest priority during the Arbitration phase. Data parity is odd. Parity is always generated and optionally checked. Parity is not valid during Arbitration.

$\overline{SEL}$. *$\overline{Select}$* (Bidirectional; Open Drain, Active LOW). $\overline{SEL}$ is used by an Initiator to select a Target, or by a Target to reselect an Initiator.

FUNCTIONAL DESCRIPTION

General: The Z5380 Small Computer System Interface (SCSI) device has a set of eight registers that are controlled by the CPU. By reading and writing the appropriate registers, the CPU may initiate any SCSI Bus activity or may sample and assert any signal on the SCSI Bus. This allows the user to implement all or any of the SCSI protocol in software. These registers are read (written) by activating $\overline{CS}$ with an address on A_0-A_2 and then issuing an $\overline{IOR}$ ($\overline{IOW}$) pulse. This section describes the operation of the internal registers (Reference Table 1)

REGISTER SUMMARY

Address				
A2	A1	A0	R/W	Register Name
0	0	0	R	Current SCSI Data
0	0	0	W	Output Data
0	0	1	R/W	Initiator Command
0	1	0	R/W	Mode
0	1	1	R/W	Target Command
1	0	0	R	Current SCSI Bus Status
1	0	0	W	Select Enable
1	0	1	R	Bus and Status
1	0	1	W	Start DMA Send
1	1	0	R	Input Data
1	1	0	W	Start DMA Target Receive
1	1	1	R	Reset Parity/Interrupts
1	1	1	W	Start DMA Initiator Receive

TABLE 1. REGISTER SUMMARY

Data Registers: The data registers are used to transfer SCSI commands, data, data, status, and message bytes between the microprocessor Data Bus and the SCSI Bus. The Z5380 does not interpret any information that passes through the data registers. The data registers consist of the transparent Current SCSI Data Register, the Output Data Register, and the Input Data Register.

Current SCSI Data Register - Address 0 (Read Only): The Current SCSI Data Register (Reference Figure 1) is a read-only register which allows the microprocessor to read the active SCSI Data Bus. This is accomplished by activating $\overline{CS}$ with an address on A_2-A_0 of 000 and issuing an $\overline{IOR}$ pulse. If parity checking is enabled, the SCSI Bus parity is checked at the beginning of the read cycle. This register is used during a programmed I/O data read or during Arbitration to check for higher priority arbitrating devices. Parity is not guaranteed valid during Arbitration.

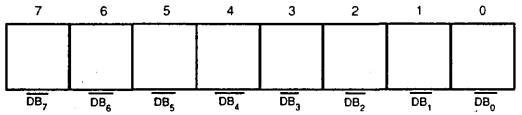


Figure 2.0: Current SCSI Data Register

Output Data Register - Address 0 (Write Only): The Output Data Register (Reference Figure 2) is a write-only register that is used to send data to the SCSI Bus. This is accomplished by either using a normal CPU write, or under DMA control, by using IOW and DACK. This register also asserts the proper ID bits on the SCSI Bus during the Arbitration and Selection phases.

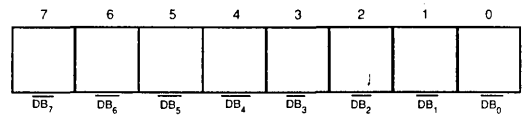


Figure3.0: Output Data Register

Input Data Register - Address 6 (Read Only): The Input Data Register (Reference Figure 3) is a read-only register that is used to read latched data from the SCSI Bus. Data is latched either during a DMA Target receive operation when $\overline{ACK}$ goes active or during a DMA Initiator receive when $\overline{REQ}$ goes active. The DMA MODE bit (port 2, bit 1) must be set before data can be latched in the Input Data Register. This register is read under DMA control using $\overline{IOR}$ and $\overline{DACK}$. Parity is optionally checked when the Input Data Register is loaded.

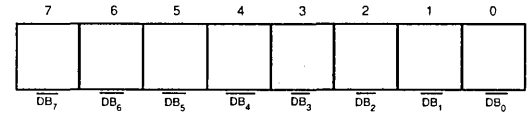


Figure 4.0: Input Data Register

Initiator Command Register - Address 1 (read/write): The Initiator Command Register (Reference Figures 5.0, 5.1) is a read/write register which asserts certain SCSI Bus signals, monitors those signals, and monitors the progress of bus arbitration. Many of these bits are significant only when being used as an Initiator; however, most can be used during Target role operation.

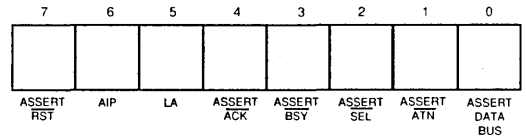


Figure 5.0: Initiator Command Register - Register Read

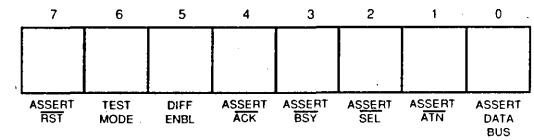


Figure 5.1: Initiator Command Register - Register Write

The following describes the operation of all bits in the Initiator Command Register.

Bit 0 -ASSERT DATA BUS: The ASSERT DATA BUS bit, when set, allows the contents of the Output Data Register to be enabled as chip outputs on the signals DB0-DB7. Parity is also generated and asserted on DBP.

When connected as an Initiator, the outputs are only enabled if the TARGETMODE bit (port 2, bit 6) is FALSE, the received signal $\overline{I/O}$ is FALSE, and the phase signals ($\overline{C/D}$, $\overline{I/O}$, and $\overline{MSG}$) match the contents of the ASSERT $\overline{C/D}$, ASSERT $\overline{I/O}$, and ASSERT $\overline{MSG}$ in the Target Command Register.

This bit should also be set during DMA send operations.

Bit 1 - ASSERT ATN ATN may be asserted on the SCSI Bus by setting this bit to a one (1) if the TARGETMODE bit (port 2, bit 6) is FALSE. $\overline{ATN}$ is normally asserted by the initiator to request a Message Out bus phase. Note that since ASSERT $\overline{SEL}$ and ASSERT $\overline{ATN}$ are in the same register, a select with $\overline{ATN}$ may be implemented with one CPU write. $\overline{ATN}$ may be deasserted by resetting this bit to zero. A read of this register simply reflects the status of this bit.

Bit 2 - ASSERT SEL Writing a one (1) into this bit position asserts SEL onto the SCSI Bus. SEL is normally asserted after Arbitration has been successfully completed. $\overline{SEL}$ may be disabled by resetting bit 2 to a zero. A read of this register reflects the status of this bit.

Bit 3 - ASSERT BSY Writing a one (1) into this bit position asserts BSY onto the SCSI Bus. Conversely, a zero resets the $\overline{BSY}$ signal. Asserting $\overline{BSY}$ indicates a successful selection or reselection. Resetting this bit creates a Bus-Disconnect condition. Reading this register reflects bit status.

Bit 4 - ASSERT ACK Bit 4 is used by the bus initiator to assert ACK on the SCSI Bus. In order to assert ACK, the TARGETMODE bit (port 2, bit 6) must be FALSE. Writing a zero to this bit deasserts $\overline{ACK}$. Reading this register reflects bit status.

Bit 5 - DIFF ENBL (Differential Enable) (WRITE Bit) Bit 5 should be written with a zero for proper operation.

Bit 5 - LA (Lost Arbitration) (Read Bit) Bit 5, when active, indicates that the Z5380 detected a Bus-Free condition, arbitrated for use of the bus by asserting BSY and its ID on the Data Bus, and lost Arbitration due to SEL being asserted by another bus device. This bit is active only when the ARBITRATE bit (port 2, bit 0) is active.

Bit 6 - TEST MODE (Write Bit) Bit 6 is written during a test environment to disable all output drivers, effectively removing the Z5380 from the circuit. Resetting this bit returns the part to normal operation.

Bit 6 - AIP (Arbitration in Progress) (Read Bit) Bit 6 is used to determine if Arbitration is in progress. for this bit to be active, the ARBITRATE bit (port 2, bit 0) must have been set previously. It indicates that a Bus-Free condition has been detected and that the chip has asserted BSY and put the contents of the Output Data Register (port 0) onto the SCSI Bus. AIP will remain active until the ARBITRATE bit is reset.

Bit 7 - ASSERT RST Whenever a one is written to bit 7 of the Initiator Command Register, the RST signal is asserted on the SCSI Bus. The RST signal will remain asserted until this bit is reset or until an external $\overline{RESET}$ occurs. After this bit is set (1), IRQ goes active and all internal logic and control registers are reset (except for the interrupt latch and the ASSERT $\overline{RST}$ bit). Writing a zero to bit 7 of the Initiator Command Register deasserts the $\overline{RST}$ signal. The status of this bit is monitored by reading the Initiator Command Register.

Mode Register - Address 2 (Read/Write): The Mode Register controls the operation of the chip. This register determines whether the Z5380 operates as an Initiator or a Target, whether DMA transfers are being used, whether parity is checked, and whether interrupts are generated on various external conditions. This register is read to check the value of these internal control bits (Reference Figure 6.0)

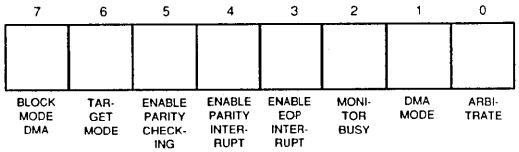


Figure 6.0: Mode Register

Bit 0 - ARBITRATE The ARBITRATE bis is set (1) to start the Arbitration process. Prior to setting this bit, the Output Data Register should contain the proper SCSI device ID value. Only one data bit should be active for SCSI Bus Arbitration. The Z5380 waits for a Bus-Free condition before entering the Arbitration phase. The results of the Arbitration phase is determined by reading the status bits LA and AIP (port 1, bits 5 and 6, respectively).

Bit 1 - DMA MODE The DMA MODE bit is normally used to enable a DMA transfer and must be set (1) prior to writing ports 5 through 7. Ports 5 through 7 are used

to start DMA transfers. The TARGETMODE bit (port 2, bit 6) must be consistent with writes to port 6 and 7 [i.e., set (1) for a write to port 6 and reset (0) for a write to port 7]. The control bit ASSERT DATA BUS (port 1, bit 0) must be TRUE (1) for all DMA send operations. In the DMA mode, REQ and ACK are automatically controlled.

The DMA MODE bit is not reset upon the receipt of an $\overline{EOP}$ signal. Any DMA transfer is stopped by writing a zero into this bit location; however, care must be taken not to cause CS and DACK to be active simultaneously.

Bit 2 - MONITOR BUSY The MONITOR BUSY bit, when TRUE (1), causes an interrupt to be generated for an unexpected loss of BSY. When the interrupt is generated due

to loss of $\overline{BSY}$, the lower six bits of the Initiator Command Register are reset (0) and all signals are removed from the SCSI Bus.

Bit 3 - ENABLE EOP INTERRUPT The enable EOP interrupt, when set (1), causes an interrupt to occur when the EOP (End of Process) signal is received from the DMA controller logic.

Bit 4 - ENABLE PARITY INTERRUPT The ENABLE PARITY INTERRUPT bit, when set (1), will cause an interrupt (IRQ) to occur if a parity error is detected. A parity interrupt will only be generated if the ENABLE PARITY CHECKING bit (bit 5) is also enabled (1).

Bit 5 - ENABLE PARITY CHECKING The ENABLE PARITY CHECKING bit determines whether parity errors are ignored or saved in the parity error latch. If this bit is reset (0), parity is ignored. Conversely, if this bit is set (1), parity errors are saved.

Bit 6 - TARGETMODE The TARGETMODE bit allows the Z5380 to operate as either a SCSI Bus Initiator, bit reset (0), or as a SCSI Bus Target device, bit set (1). If the signals ATN and ACK are to be asserted on the SCSI Bus, the TARGETMODE bit must be reset (0). If the signals C/D, I/O, $\overline{MSG}$, and $\overline{REQ}$ are to be asserted on the SCSI Bus, the TARGETMODE bit must be set (1).

Bit 7 - BLOCK MODE DMA The BLOCK MODE DMA bit controls the characteristics of the DMA DRQ-DACK handshake. When this bit is reset (0) and the DMA MODE bit is active (1), the DMA handshake uses the normal interlocked handshake, and the rising edge of $\overline{DACK}$ indicates the end of each byte being transferred. In block mode operations, BLOCK MODE DMA bit set (1) and DMA MODE bit set (1), the end of $\overline{IOR}$ or $\overline{IOW}$ signifies the end of each byte transferred and DACK is allowed to remain active throughout the DMA operation. READY can then be used to request the next transfer.

Target Command Register - Address 3 (Read/Write) When connected as a target device, the Target Command Register (Reference Figure 7) allows the CPU to control the SCSI Bus Information Transfer phase and/or to assert REQ by writing this register. The TARGETMODE bit (port 2, bit 6) must be TRUE (1) for bus assertion to occur. The SCSI Bus phases are described in Table 2

SCSI INFORMATION TRANSFER PHASES

Bus Phase	ASSERT $\overline{I/O}$	ASSERT $\overline{C/D}$	ASSERT $\overline{MSG}$
Data Out	0	0	0
Unspecified	0	0	1
Command	0	1	0
Message Out	0	1	1
Data In	1	0	0
Unspecified	1	0	1
Status	1	1	0
Message In	1	1	1

TABLE 2. SCSI INFORMATION TRANSFER PHASES

When connected as an Initiator with DMA Mode TRUE, if the phase lines ($\overline{I/O}$, $\overline{C/D}$, and $\overline{MSG}$) do not match the phase bits in the Target Command Register, a phase-mismatch interrupt is generated when $\overline{REQ}$ goes active. To send data as an Initiator, the ASSERT $\overline{I/O}$, ASSERT $\overline{C/D}$, and ASSERT $\overline{MSG}$ bits must match the corresponding bits in the Current SCSI Bus Status Register (port 4). The ASSERT $\overline{REQ}$ bit (bit 3) has no meaning when operating as an Initiator.

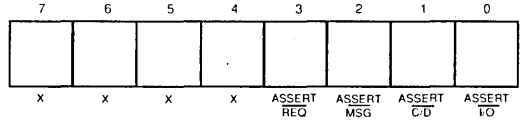


Figure 7.0: Target Command Register

Current SCSI Bus Status Register - Address 4 (Read Only): The Current SCSI Bus Register is a read-only register which is used to monitor seven SCSI Bus control signals, plus the Data Bus parity bit. For example, an Initiator device can use this register to determine the current bus phase and to poll REQ for pending data transfers. This register may also be used to determine why a particular interrupt occurred. Figure 8.0 describes the Current SCSI Bus Status Register.

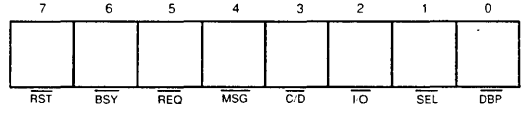


Figure 8.0: Current SCSI Bus Status Register

Select Enable Register - Address 4 (Write Only) The Select Enable Register (Reference Figure 8) is a write-only register which is used as a mask to monitor a signal ID during a selection attempt. The simultaneous occurrence of the correct ID bit, BSY FALSE, and SEL TRUE will cause an interrupt. This interrupt can be disabled by resetting all bits in this register. If the ENABLE PARITY CHECKING bit (port 2, bit 5) is active (1), parity is checked during selection.

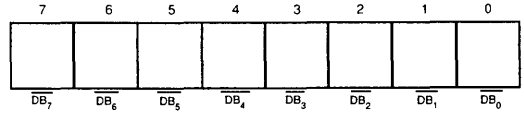


Figure 9.0: Select Enable Register

Bus and Status Register - Address 5 (Read Only) The Bus and Status Register (Reference Figure 10.0) is a read-only register which can be used to monitor the remaining SCSI control signals not found in the Current SCSI Bus Status Register (ATN and ACK), as well as six other status bits. The following describes each bit of the Bus Status Register individually.

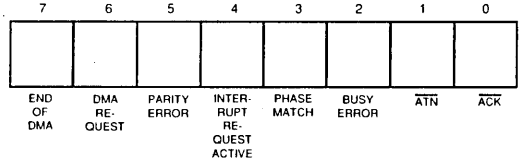


Figure 10.0: Bus and Status Register

Bit 0 - ACK Bit 0 reflects the condition of the SCSI Bus control signal ACK. This signal is normally monitored by the Target device.

Bit 1 - ATN Bit 1 reflects the condition of the SCSI Bus control signal ATN. This signal is normally monitored by the Target device.

Bit 2 - BUSY ERROR The BUSY ERROR bit is active if an unexpected loss of the BSY signal has occurred. This latch is set whenever the MONITOR BUSY bit (port 2, bit 2) is TRUE and $\overline{BSY}$ is FALSE. An unexpected loss of $\overline{BSY}$ disables any SCSI outputs and resets the DMA MODE bit (port 2, bit 1).

Bit 3 - PHASE MATCH The SCSI signals, $\overline{MSG}$, $\overline{C/D}$, and $\overline{I/O}$, represent the current information Transfer phase. The PHASE MATCH bit indicates whether the current SCSI Bus phase matches the lower 3 bits of the Target Command Register. PHASE MATCH is continuously updated and is only significant when operating as a Bus Initiator. A phase match is required for data transfers to occur on the SCSI Bus.

Bit 4 - INTERRUPT REQUEST ACTIVE Bit 4 is set if an enabled interrupt condition occurs. It reflects the current state of the IRQ output and can be cleared by reading the Reset Parity/Interrupt Register (port 7).

Bit 5 - PARITY ERROR Bit 5 is set if a parity error occurs during a data receive or a device selection. The PARITY ERROR bit can only be set (1) if the ENABLE PARITY CHECK bit (port 2, bit 5) is active (1). This bit may be cleared by reading the Reset Parity/Interrupt Register (port 7).

Bit 6 - DMA REQUEST The DMA REQUEST bit allows the CPU to sample the output pin DRQ. DRQ can be cleared by asserting $\overline{DACK}$ or by resetting the DMA MODE bit (bit 1) in the Mode Register (port 2). The DRQ signal does not reset when a phase-mismatch interrupt occurs.

Bit 7 - END OF DMA TRANSFER The END OF DMA TRANSFER bit is set if EOP, DACK, and either IOR or IOW are simultaneously active for at least 100ns. Since the $\overline{EOP}$ signal can occur during the last byte sent to the Output Data Register (Port 0), the $\overline{REQ}$ and $\overline{ACK}$ signals should be monitored to ensure that the last byte has been transferred. This bit is reset when the DMA MODE bit is reset (0) in the Mode Register (port 2).

DMA Registers Three write-only registers are used to initiate all DMA activity. They are: Start DMA Send (port 5), Start DMA Target Receive (port 6), and Start DMA Initiator Receive (port 7). Performing a write operation into one of these registers starts the desired type of DMA transfer. Data presented to the Z5380 on signals D0-D7 during the register write is meaningless and has no effect on the operation. Prior to writing these registers; the BLOCK MODE DMA bit (bit 7), the DMA MODE bit (bit 1), and the TARGETMODE bit (bit 6) in the Mode Register (port 2) must be appropriately set. The individual registers are briefly described as follows:

Start DMA Send - Address 5 (Write Only) This register is written to initiate a DMA send, from the DMA to the SCSI Bus, for either Initiator or Target role operations. The DMA MODE bit (port 2, bit 1) is set prior to writing this register.

Start DMA Target Receive - Address 6 (Write Only) This register is written to initiate a DMA receive - from the SCSI Bus to the DMA, for Target operation only. The DMA MODE bit (bit 1) and the TARGETMODE bit (bit 6) in the Mode Register (port 2) must both be set (1) prior to writing this register.

Start DMA Initiator Receive - Address 7 (Write Only) This register is written to initiate a DMA receive - from the SCSI Bus to the DMA, for Initiator operation only. The DMA MODE bit (bit 6) must be FALSE (0) in the Mode Register (port 2) prior to writing this register.

Reset Parity/Interrupt - Address 7 (Read Only): Reading this register resets the PARITY ERROR bit (bit 5), the INTERRUPT REQUEST bit (bit4), and the BUSY ERROR bit (bit 2) in the Bus and Status Register (port 5).

On-Chip SCSI Hardware Support: The Z5380 is easy to use because of its simple architecture. The chip allows direct control and monitoring of the SCSI Bus by providing a latch for each signal. However, portions of the protocol define timings which are much too quick for traditional microprocessors to control. Therefore, hardware suppport has been provided for DMA transfers, bus arbitration, phasechange monitoring, bus disconnection, bus reset, parity generation, parity checking, and device selection/reselection.

Arbitration is accomplished using a Bus-Free filter to continuously monitor $\overline{BSY}$. If $\overline{BSY}$ remains inactive for at least 400ns, the SCSI Bus is considered free and Arbitration may begin. Arbitration will begin if the bus is free, $\overline{SEL}$ is inactive, and the ARBITRATION bit (port 2, bit 0) is active. Once arbitration has begun ($\overline{BSY}$ asserted), an arbitration

may begin. Arbitration will begin if the bus is free, $\overline{SEL}$ is inactive, and the ARBITRATION bit (port 2, bit 0) is active. Once arbitration has begun ($\overline{BSY}$ asserted), an arbitration delay of 2.2 us must elapse before the Data Bus can be examined to determine if Arbitration is enabled. This delay is implemented in the controlling software driver.

The Z5380 is a clockwise device. Delays such as bus-free delay, bus-set delay, and bus-settle delay are implemented using gate delays. These delays may differ between devices because of inherent process variations, but are well within the proposed ANSI X3T9.2 specification.

INTERRUPTS The Z5380 provides an interrupt output (IRQ) to indicate a task completion or an abnormal bus occurrence. The use of interrupts is optional and may be disabled by resetting the appropriate bits in the Mode Register (port 2) or the Select Enable Register (port 4).

When an interrupt occurs, the Bus and Status Register and the Current SCSI Bus Status Register (Reference Figures 10 & 11) must be read to determine which condition created the interrupt. IRQ can be reset simply by reading the Reset Parity/Interrupt Register (port 7) or by an external chip reset ($\overline{RESET}$ active for 200 ns).

Assuming the Z5380 has been properly initialized, an interrupt will be generated; if the chip is selected or reselected, if an $\overline{EOP}$ signal occurs during a DMA transfer, if a SCSI Bus reset occurs, if a parity error occurs during a data transfer, if a bus phase mismatch occurs, or if a SCSI Bus disconnection occurs.

Selection/Reselection The Z5380 generates a select interrupt if SEL is TRUE (1), its device ID is TRUE (1), and $\overline{BSY}$ is FALSE for at least a bus-settle delay (400 ns). If $\overline{I/O}$ is active, this is considered a reselect interrupt. The correct ID bit is determined by a match in the Select Enable Register (port 4). Only a single bit match is required to generate an interrupt. This interrupt may be disabled by writing zeros into all bits of the Select Enable Register.

If parity is supported, parity should be good during the selection phase. Therefore, if the ENABLE PARITY bit (port 2, bit 5) is active, the PARITY ERROR bit is checked to ensure that a proper selection has occurred. The ENABLE PARITY INTERRUPT bit need not be set for this interrupt to be generated.

The proposed SCSI specification also requires that no more than two device ID's be active during the selection process. To ensure this, the Current SCSI Data Register (port 0) is read.

The proper values forthe Bus and Status Register (port 5) and the Current SCSI Bus Status Register (port 4) are displayed in Figures 11.0 and 12.0, respectively.

7	6	5	4	3	2	1	0
0	0	0	1	x	0	x	0
END OF DMA	DMA REQUEST	PARITY ERROR	INTERRUPT REQUEST ACTIVE	PHASE MATCH	BUSY ERROR	$\overline{ATN}$	$\overline{ACK}$

Figure 11.0: Bus and Status Register

7	6	5	4	3	2	1	0
0	0	0	x	x	x	1	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/D	$\overline{I/O}$	$\overline{SEL}$	$\overline{DBP}$

Figure 12.0: Current SCSI Bus Status Register

End of Process (EOP) Interrupt: An End of Process signal (EOP) which occurs during a DMA transfer (DMAMODE TRUE) will set the END OF DMA Status bit (port 5, bit 7) and will optionally generate an interrupt if ENABLE EOP INTERRUPT bit (port 2, bit 3) is TRUE. The $\overline{EOP}$ pulse will not be recognized (END OF DMA bit set) unless $\overline{EOP}$, $\overline{DACK}$, and either $\overline{IOR}$ or $\overline{IOW}$ are concurrently active for at least 100 ns. DMA transfers can still occur if $\overline{EOP}$ was not asserted at the correct time. This interrupt is disabled by resetting the ENABLE EOP INTERRUPT bit.

The proper values for the Bus and Status Register (port 5) and the Current SCSI Bus Status Register (port 4) for this interrupt are shown in Figures 13.0 and 14.0.

7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	x
END OF DMA	DMA REQUEST	PARITY ERROR	INTERRUPT REQUEST ACTIVE	PHASE MATCH	BUSY ERROR	$\overline{ATN}$	$\overline{ACK}$

Figure 13.0: Bus and Status Register

7	6	5	4	3	2	1	0
0	1	1	x	x	x	0	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/D	$\overline{I/O}$	$\overline{SEL}$	$\overline{DBP}$

Figure 14.0: Current SCSI Bus Status Register

The END OF DMA bit is used to determine when a block transfer is complete. Receive operations are complete when there is no data left in the chip and no additional handshakes occurring. The only exception to this is receiving data as an Initiator and the Target opts to send additional data for the same phase. In this case, $\overline{REQ}$ goes active and the new data is present in the Input Data Register. Since a phase-mismatch interrupt will not occur, $\overline{REQ}$ and $\overline{ACK}$ need to be sampled to determine that the Target is attempting to send more data.

For send operations, the END OF DMA bit is set when the DMA finishes its transfer, but the SCSI transfer may still be in progress. If connected as a Target, $\overline{REQ}$ and $\overline{ACK}$ should be sampled until both are FALSE. If connected as an Initiator, a phase change interrupt is used to signal the

completion of the previous phase. It is possible for the Target to request additional data for the same phase. In this case, a phase change will not occur and both $\overline{REQ}$ and $\overline{ACK}$ are sampled to determine when the last byte was transferred.

SCSI Bus Reset: The Z5380 generates an interrupt when the $\overline{RST}$ signal transitions to TRUE. The device releases all bus signals within a bus-clear delay (800 ns) of this transition. This interrupt also occurs after setting the ASSERT $\overline{RST}$ bit (port 1, bit 7). This interrupt cannot be disabled. (note: $\overline{RST}$ is not latched in bit 7 of the Current SCSI Bus Status Register and is not active when this port is read. For this case, the Bus Reset interrupt is determined by default).

The proper values for the Bus and Status Register (port 5) and the Current SCSI Bus Status Register (port 4) are displayed in Figures 15.0 and 16.0, respectively.

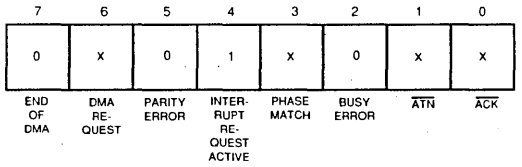


Figure 15.0: Bus and Status Register

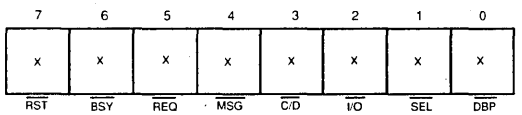


Figure 16.0: Current SCSI Bus Status Register

Parity Error: An interrupt is generated for a received parity error if the ENABLE PARITY CHECK (bit 5) and the ENABLE PARITY INTERRUPT (bit 4) bits are set (1); in the Mode Register (port 2). Parity is checked during a read of the Current SCSI Data Register (port 0) and during a DMA receive operation. A parity error can be detected without generating an interrupt by disabling the ENABLE PARITY INTERRUPT bit and checking the PARITY ERROR flag (port 5, bit 5).

The proper values for the Bus and Status Register (port 5) and the Current SCSI Bus Status Register (port 4) are displayed in Figures 17.0 and 18.0, respectively.

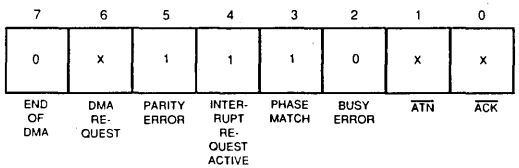


Figure 17.0: Bus and Status Register

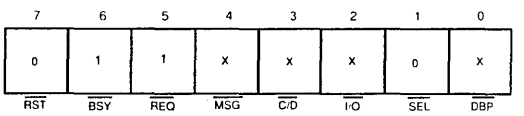


Figure 18.0: Current SCSI Bus Status Register

Bus Phase Mismatch: The SCSI phase lines are comprised of the signals I/O, C/D, and MSG. These signals are compared with the corresponding bits in the Target Command Register: ASSERT $\overline{I/O}$ (bit 0), ASSERT $\overline{C/D}$ (bit 1), and ASSERT $\overline{MSG}$ (bit 2). The comparison occurs continually and is reflected in the PHASE MATCH bit (bit 3) of the Bus and Status Register (port 5). If the DMA MODE bit (port 2, bit 1) is active and a phase mismatch occurs when $\overline{REQ}$ transitions from FALSE to TRUE, an interrupt (IRQ) is generated.

A phase mismatch prevents the recognition of $\overline{REQ}$ and removes the chip from the bus during an Initiator send operation (DB_0-DB_7 and $\overline{DBP}$ will not be driven even though the ASSERT DATA BUS bit (port 1, bit 0) is active). This may be disabled by resetting the DMA MODE bit (Note: it is possible for this interrupt to occur when connected as a Target if another device is driving the phase lines to a different state).

The proper values for the Bus and Status Register (port 5) and the Current SCSI Bus Status Register (port 4) are displayed in Figures 19.0 and 20.0, respectively.

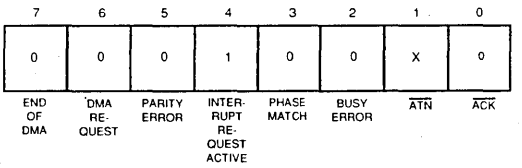


Figure 19.0: Bus and Status Register

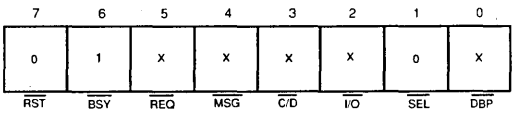


Figure 20.0: Current SCSI Bus Status Register

Loss of BSY: If the MONITOR BUSY bit (bit 2) in the Mode Register (port 2) is active, an interrupt is generated if the BSY signal goes FALSE for at least a bus-settle delay (400 ns). This interrupt is disabled by resetting the MONITOR BUSY bit. Register values are displayed in Figures 21.0 and 22.0.

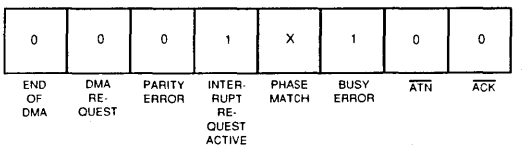


Figure 21.0: Bus and Status Register

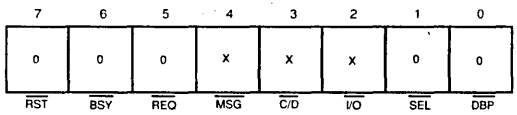


Figure 22.0: Current SCSI Bus Status Register

Reset Conditions: Three possible reset situations exist with the Z5380, as follows:

Hardware Chip Reset When the signal RST is active for at least 200 ns, the Z5380 device is re-initialized and all internal logic and control registers are cleared. This is a chip reset only and does not create a SCSI Bus-Reset condition.

SCSI Bus Reset (RST) Received: When a SCSI $\overline{RST}$ signal is received, an IRQ interrupt is generated and a chip reset is performed. All internal logic and registers are cleared, except for the IRQ interrupt latch and the ASSERT $\overline{RST}$ bit (bit 7) in the Initiator Command Register (port 1). (Note: the $\overline{RST}$ signal may be sampled by reading he Current SCSI Bus Status Register (port 4); however, this signal is not latched and may not be present when this port is read.)

SCSI Bus Reset (RST) Issued : If the CPU sets the ASSERT $\overline{RST}$ bit (bit 7) in the Initiator Command Register (port 1), the $\overline{RST}$ signal goes active on the SCSI Bus and an internal reset is performed. Again, all internal logic and registers are cleared except for the IRQ interrupt latch and the ASSERT $\overline{RST}$ bit (bit 7) in the Initiator Command Register (port 1). The $\overline{RST}$ signal will continue to be active until the ASSERT $\overline{RST}$ bit is reset or until a hardware reset occurs.

Data Transfers: Data is transferred between SCSI Bus devices in one of four modes: 1) Programmed I/O, 2) Normal DMA, 3) Block Mode DMA, or 4) Pseudo DMA. The following sections describe these modes in detail (Note: for all data transfer operations DACK and CS should never be active simultaneously).

Programmed I/O Transfers: Programmed I/O is the most primitive form of data transfer. The REQ and ACK handshake signals are individually monitored and asserted by reading and writing the appropriate register bits. This type of transfer is normally used when transferring small blocks of data such as command blocks or message and status bytes. An Initiator send operation would begin by setting the $\overline{C/D}$, $\overline{I/O}$, and $\overline{MSG}$ bits in the Target Command Register to the correct state so that a phase match exists. In addition to the phase match condition, it is necessary for the ASSERT DATA BUS bit (port 1, bit 0) to be TRUE and the received I/O signal to be FALSE for the Z5380 to send data. For each transfer, the data is loaded into the Output Data Register (port 0). The CPU then waits for the $\overline{REQ}$ bit (port 4, bit 5) to become active. Once $\overline{REQ}$ goes active, the PHASE Match bit (port 5, bit 3) is checked and the ASSERT $\overline{ACK}$ bit (port 1, bit 4) is set. The $\overline{REQ}$ bit is sampled until it becomes FALSE and the CPU resets the ASSERT $\overline{ACK}$ bit to complete the transfer.

Normal DMA Mode: DMA transfers are normally used for large block transfers. The SCSI chip outputs a DMA request (DRQ) whenever it is ready for a byte transfer. External DMA logic uses this DRQ signal to generate DACK and an IOR or an IOW pulse to the Z5380. DRQ goes inactive when $\overline{DACK}$ is asserted and $\overline{DACK}$ goes inactive some time after the minimum read or write pulse width. This process is repeated for every byte. For this mode, $\overline{DACK}$ should not be allowed to cycle unless a transfer is taking place.

Block Mode DMA: Some popular DMA controllers, such as the 9517A, provide a Block Mode DMA transfer. This type of transfer allows the DMA controller to transfer blocks of data without relinquishing the use of the Data Bus to the CPU after each byte is transferred; thus, faster transfer rates are achieved by eliminating the repetitive access and release of the CPU Bus. If the BLOCK MODE DMA bit (port 2, bit 7) is active, the Z5380 begins the transfer by asserting DRQ. The DMA controller then asserts $\overline{DACK}$ for the remainder of the block transfer. DRQ goes inactive for the duration of the transfer. The READY output is used to control the transfer rate. Non-Block Mode DMA transfers end when $\overline{DACK}$ goes FALSE, whereas Block Mode transfers end when $\overline{IOR}$ or $\overline{IOW}$ becomes inactive. Since this is the case, DMA transfers may be started sooner in a Block Mode transfer. To obtain optimum performance in Block Mode operation, the DMA logic optionally uses the normal DMA mode interlocking handshake. READY is still available to throttle the DMA transfer, but DRQ is 30 to 40 ns faster than READY and is used to start the cycle sooner. The methods described under "Halting a DMA Operation" apply for all DMA operations.

Pseudo DMA Mode: To avoid the tedium of monitoring and asserting the request/acknowledge handshake signals for programmed I/O transfers, the system may be designed to implement a pseudo DMA mode. This mode is implemented by programming the Z5380 to operate in the DMA mode, but using the CPU to emulate the DMA handshake. DRQ may be detected by polling the DMA REQUEST bit (bit 6) in the Bus and Status Register (port 5), by sampling the signal through an external port, or by using it to generate a CPU interrupt. Once DRQ is detected, the CPU can perform a read or write data transfer. This CPU read/write is externally decoded to generate the appropriate $\overline{DACK}$ and $\overline{IOR}$ or $\overline{IOW}$ signals.

Often, external decoding logic is necessary to generate the Z5380 $\overline{CS}$ signal. This same logic may be used to generate $\overline{DACK}$ at no extra system cost and provide an increased performance in programmed I/O transfers.

Halting a DMA Operation: The EOP signal is not the only way to halt a DMA transfer. A bus phase mismatch or a reset of the DMA MODE bit (port 2, bit 1) can also terminate a DMA cycle for the current bus phase.

Using the EOP Signal If EOP is used, it should be asserted for at least 100 ns while DACK and IOR or IOW are simultaneously active. Note, however, that if IOR or IOW is not active, an interrupt is generated, but the DMA activity continues. The EOP signal does not reset the DMA MODE bit. Since the EOP signal can occur during the last byte sent to the Output Data Register (port 0), the REQ and ACK signals are monitored to ensure that the last byte has transferred.

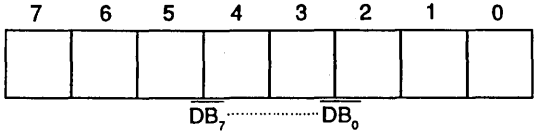
Bus Phase Mismatch Interrupt: A bus phase mismatch interrupt is used to halt the transfer if operating as an Initiator. Using this method frees the host from maintaining a data length counter and frees the DMA logic from providing the EOP signal. If performing an Initiator send operation, the **Z5380** requires DACK to cycle before ACK goes inactive. Since phase changes cannot occur if ACK is active, either DACK must be cycled after the last byte is sent or the DMA MODE bit must be reset in order to receive the phase mismatch interrupt.

Resetting the DMA MODE Bit: A DMA operation may be halted at any time simply by resetting the DMA MODE bit. It is recommended that the DMA MODE bit be reset after receiving an EOP or bus phase-mismatch interrupt. The DMA MODE bit must then be set before writing any of the start DMA registers for subsequent bus phases.

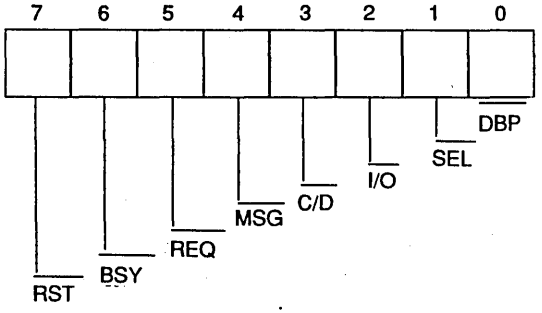
If resetting the DMA MODE bit is used instead of EOP for Target role operation, then care must be taken to reset this bit at the proper time. If receiving data as a Target device, the DMA MODE bit must be reset once the last DRQ is received and before DACK is asserted to prevent an additional REQ from occurring. Resetting this bit causes DRQ to go inactive. However, the last byte received remains in the Input Data Register and may be obtained either by performing a normal CPU read or by cycling DACK and IOR. In most cases, EOP is easier to use when operating as a Target device.

READ

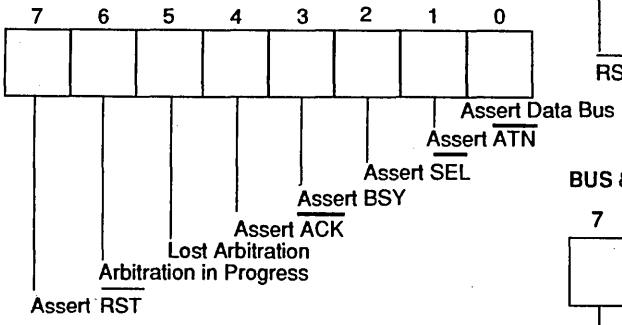
CURRENT SCSI DATA (00)



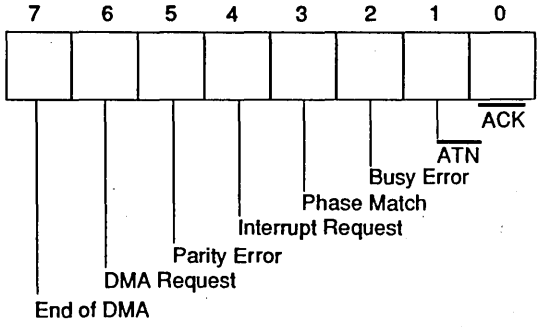
CURRENT SCSI BUS STATUS (04)



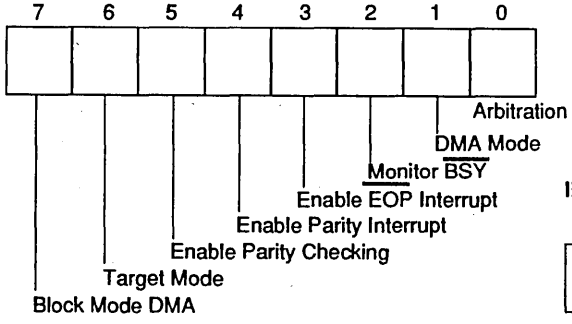
INITIATOR COMMAND REGISTER (01)



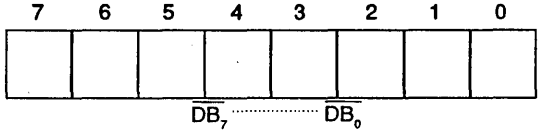
BUS & STATUS REGISTER (05)



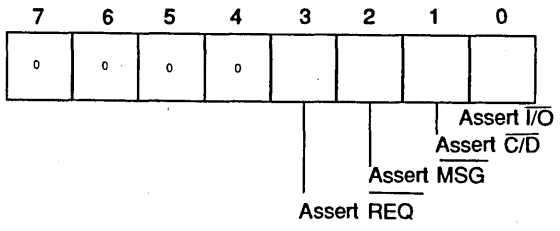
MODE REGISTER (02)



INPUT DATA REGISTER (06)



TARGET COMMAND REGISTER (03)



RESET PARITY/INTERRUPT (07)

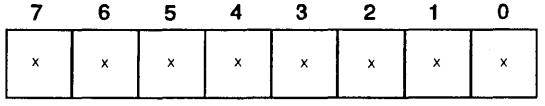
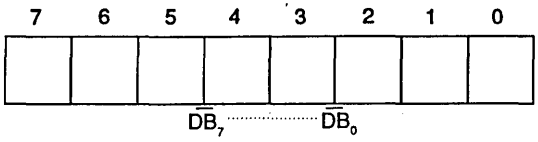
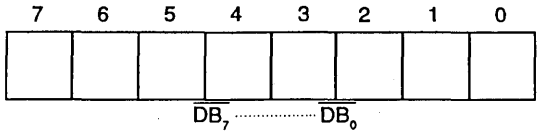


Figure 27.0 Register Reference Chart

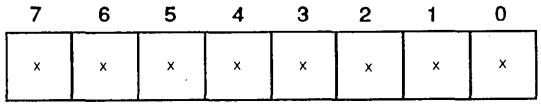
OUTPUT DATA REGISTER (00)



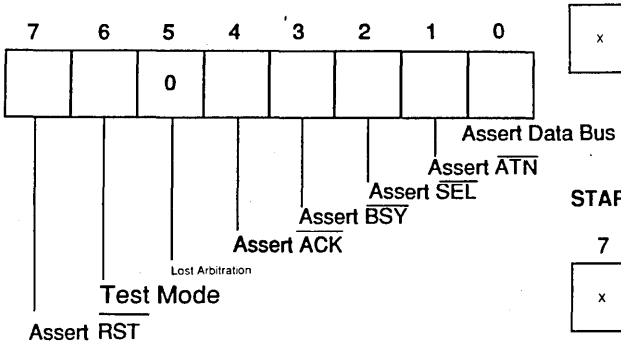
SELECT ENABLE REGISTER (04)



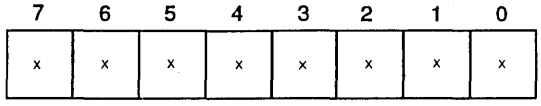
START DMA SEND (05)



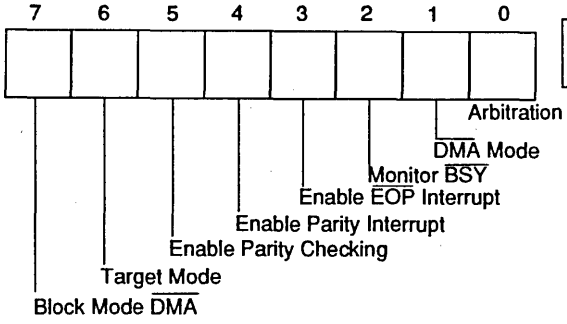
INITIATOR COMMAND REGISTER (01)



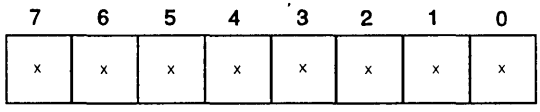
START DMA TARGET RECEIVE (06)



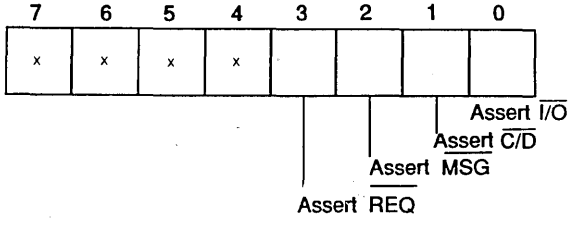
MODE REGISTER (02)



START DMA INITIATOR RECEIVE (07)



TARGET COMMAND REGISTER (03)



NOTE: X = DONT CARE

Figure 27.0 Register Reference Chart

ABSOLUTE MAXIMUM RATINGS

Store Temperature	-65 to +150 deg C
Supply Voltage on Any Pin with Respect to Ground	-0.5 to + 7.0 V
Power Dissipation	0.2 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGE

Commercial (C) Devices

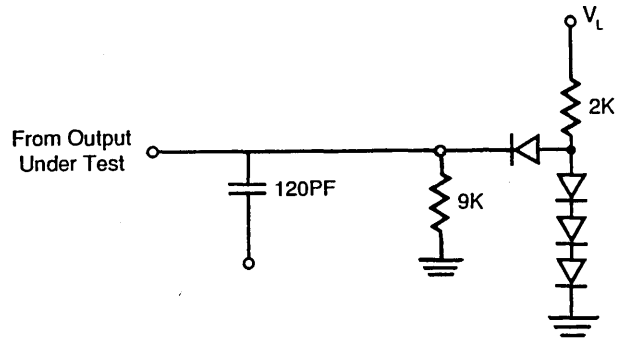
Temperature (T_A)	0 to + 70 C
Supply Voltage (VCC)	+ 4.74 to + 5.25 V

Operating ranges define those limits between which the functionality of the device is gauranteed.

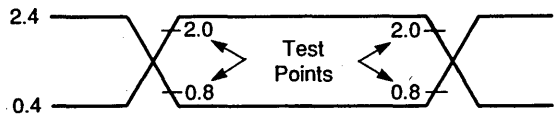
DC CHARACTERISTICS over operating range unless otherwise specified

Parameter Description	Test Conditions	Min.	Max.	
Input Signal Requirements				
HIGH-Level, Input VI_H		2.0	5.25	V
LOW-Level, Input V_{IL}		-0.3	0.8	V
HIGH-Level, Input Current, I_{IH}, on: SCSI Bus Pins	V_{IH} = 5.25 V. V_{IL} = 0		50	µA
All Other Pins			10	
LOW-Level Input Current, I_{IL} on: SCSI Bus Pins	V_{IH} = 5.25 V. V_{IL} = 0		-50	µA
All Other Pins			-10	
Output Signal Requirements				
HIGH-Level Output on All Pins	V_{DD}=4.75V. I_{OH} = 3.0 mA	2.4		V
LOW-Level Output on: SCSI Bus Pins	V_{DD}=4.75V. I_{OL} = 48.0 mA		0.5	V
All Other Pins	V_{DD} =4.75V. I_{OL} = 7.0 mA		0.5	V

SWITCHING TEST CIRCUIT



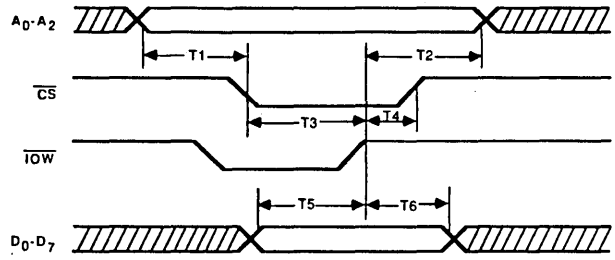
SWITCHING TEST WAVEFORM



CPU Write Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	Address Setup to Write Enable*	20				ns
T2	Address Hold from End Write Enable*	20				ns
T3	Write Enable Width*	70				ns
T4	Chip Select Hold from End of $\overline{IOW}$	0				ns
T5	Data Setup to end of Write Enable*	50				ns
T6	Data Hold Time from End of $\overline{IOW}$	30				ns

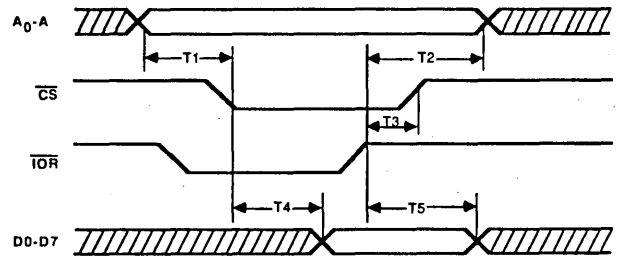
*Write Enable is the occurrence of $\overline{IOW}$ and $\overline{DACK}$



CPU Read Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	Address Setup to Read Enable*	20				ns
T2	Address Hold from End Read Enable*	20				ns
T3	Chip Select Hold from End of $\overline{IOR}$	0				ns
T4	Data Access Time from Read Enable*		130			ns
T5	Data Hold Time from End of $\overline{IOR}$	20				ns

*Read Enable is the occurrence of IOR and CS

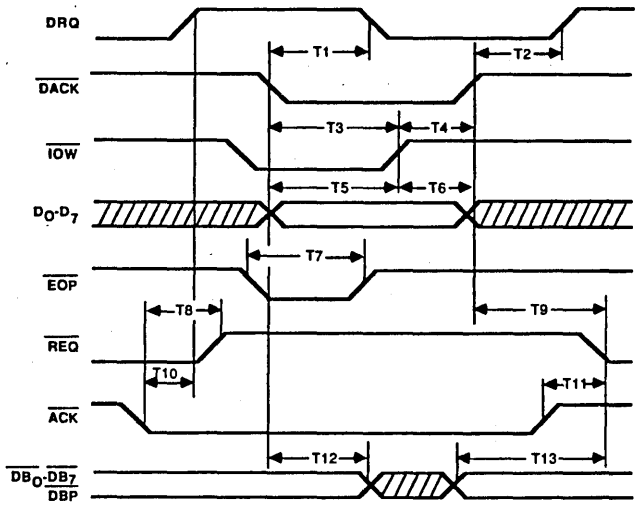


DMA Write (Non-Block Mode) Target Send Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	DRQ FALSE from $\overline{DACK}$ TRUE		130			ns
T2	$\overline{DACK}$ FALSE to DRQ TRUE	30				ns
T3	Write Enable Width*	100				ns
T4	DACK Hold from End of $\overline{IOW}$	0				ns
T5	Data Setup to End of Write Enable*	50				ns
T6	Data Hold Time from End of $\overline{IOW}$	40				ns
T7	Width of $\overline{EOP}$ Pulse (Note 1)	100				ns
T8	$\overline{ACK}$ TRUE to REQ FALSE	25	125			ns
T9	$\overline{REQ}$ from End of $\overline{DACK}$ ($\overline{ACK}$ FALSE)	30	150			ns
T10	$\overline{ACK}$ TRUE to DRQ TRUE (Target)	15	110			ns
T11	$\overline{REQ}$ from End of $\overline{ACK}$ ($\overline{DACK}$ FALSE)	20	150			ns
T12	Data Hold from Write Enable	15				ns
T13	Data Setup to $\overline{REQ}$ TRUE (Target)	60				ns

*Write Enable is the occurrence of $\overline{IOW}$ and $\overline{DACK}$

Notes: 1. $\overline{EOP}$, $\overline{IOW}$, and $\overline{DACK}$ must be concurrently TRUE for at least T7 for proper recognition of the $\overline{EOP}$ pulse.

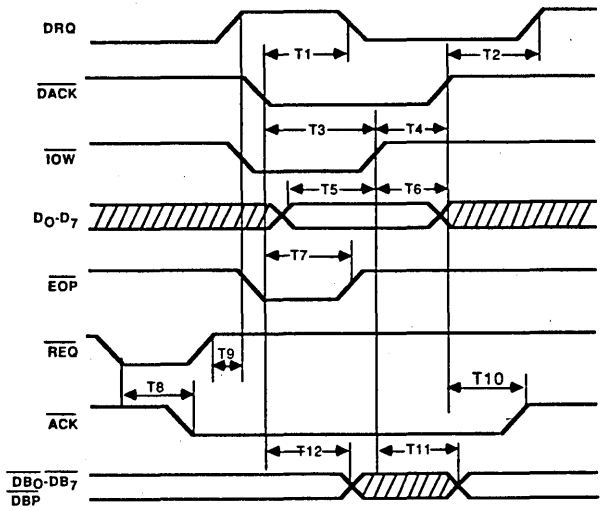


DMA Write (Non-Block Mode) Initiator Send Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	DRQ FALSE from $\overline{DACK}$ TRUE		130			ns
T2	$\overline{DACK}$ FALSE to DRQ TRUE	30				ns
T3	Write Enable Width*	100				ns
T4	$\overline{DACK}$ Hold from End of $\overline{IOW}$	0				ns
T5	Data Setup to End of Write Enable*	50				ns
T6	Data Hold Time from End of IOW	40				ns
T7	Width of $\overline{EOP}$ Pulse (Note 1)	100				ns
T8	$\overline{REQ}$ TRUE to $\overline{ACK}$ TRUE	20	160			ns
T9	$\overline{REQ}$ FALSE to DRQ TRUE	20	110			ns
T10	$\overline{DACK}$ FALSE to $\overline{ACK}$ FALSE	25	150			ns
T11	$\overline{IOW}$ FALSE to Valid SCSI Data		100			ns
T12	Data Hold from Write Enable	15				ns

*Write Enable is the occurrence of $\overline{IOW}$ and $\overline{DACK}$

Notes: 1. $\overline{EOP}$, $\overline{IOW}$, and $\overline{DACK}$ must be concurrently TRUE for at least T7 for proper recognition of the $\overline{EOP}$ pulse.

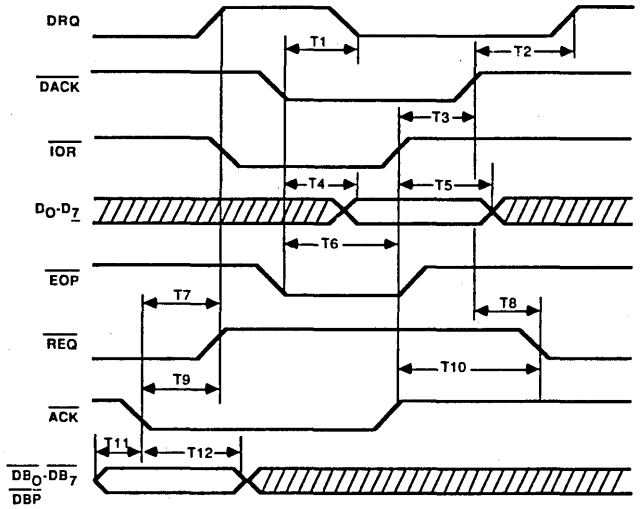


DMA Read (Non-Block Mode) Target Receive Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	DRQ FALSE from $\overline{DACK}$ TRUE		130			ns
T2	$\overline{DACK}$ FALSE to DRQ TRUE	30				ns
T3	$\overline{DACK}$ Hold Time from End of IOR	0				ns
T4	Data Access Time from Read Enable*		115			ns
T5	Data Hold Time from End of $\overline{IOR}$	20				ns
T6	Width of $\overline{EOP}$ Pulse (Note 1)	100				ns
T7	$\overline{ACK}$ TRUE to DRQ TRUE	15	110			ns
T8	$\overline{DACK}$ FALSE to $\overline{REQ}$ TRUE ($\overline{ACK}$ FALSE)	30	150			ns
T9	$\overline{ACK}$ TRUE to $\overline{REQ}$ FALSE	25	125			ns
T10	$\overline{ACK}$ FALSE to $\overline{REQ}$ TRUE ($\overline{DACK}$ FALSE)	20	150			ns
T11	Data Setup Time to $\overline{ACK}$	20				ns
T12	Data Hold Time from $\overline{ACK}$	50				ns

*Read Enable is the occurrence of $\overline{IOR}$ and $\overline{DACK}$

Notes: 1. $\overline{EOP}$, $\overline{IOR}$, and $\overline{DACK}$ must be concurrently TRUE for at least T6 for proper recognition of the $\overline{EOP}$ pulse.

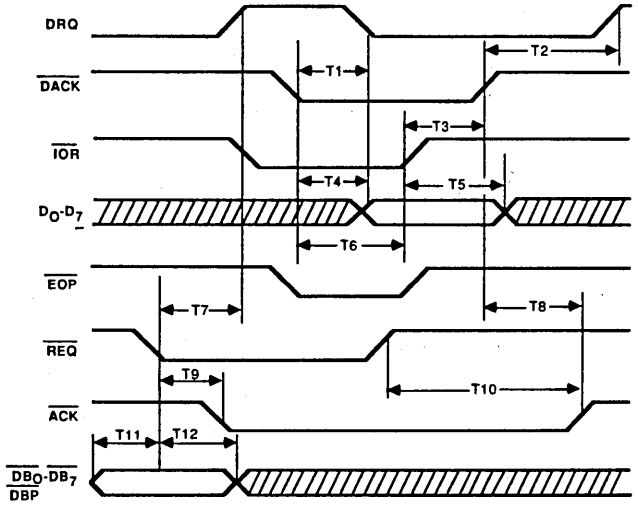


DMA Read (Non-Block Mode) Initiator Receive Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	DRQ FALSE from $\overline{DACK}$ TRUE		130			ns
T2	$\overline{DACK}$ FALSE to DRQ TRUE	30				ns
T3	$\overline{DACK}$ Hold Time from End of $\overline{IOR}$	0				ns
T4	Data Access Time from Read Enable*		115			ns
T5	Data Hold Time from End of $\overline{IOR}$	20				ns
T6	Width of $\overline{EOP}$ Pulse (Note 1)	100				ns
T7	$\overline{REQ}$ TRUE to DRQ TRUE	20	150			ns
T8	$\overline{DACK}$ FALSE to $\overline{ACK}$ FALSE ($\overline{REQ}$ FALSE)	25	160			ns
T9	$\overline{REQ}$ TRUE to $\overline{ACK}$ TRUE	20	160			ns
T10	$\overline{REQ}$ FALSE to $\overline{ACK}$ FALSE ($\overline{DACK}$ FALSE)	15	140			ns
T11	Data Setup Time to $\overline{REQ}$	20				ns
T12	Data Hold Time from $\overline{REQ}$	50				ns

*Read Enable is the occurrence of $\overline{IOR}$ and $\overline{DACK}$

Notes: 1. $\overline{EOP}$, $\overline{IOR}$, and $\overline{DACK}$ must be concurrently TRUE for at least T6 for proper recognition of the $\overline{EOP}$ pulse.

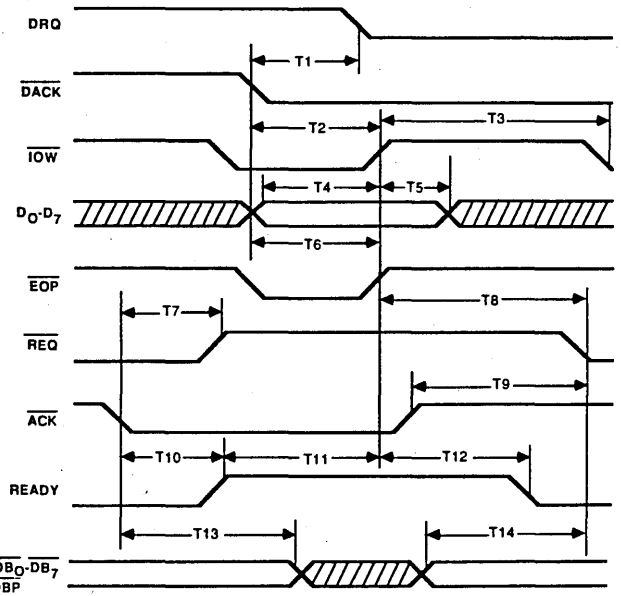


DMA Write (Block Mode) Target Send Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	DRQ FALSE from $\overline{DACK}$ TRUE		130			ns
T2	Write Enable Width*	100				ns
T3	Write Recovery Time	120				ns
T4	Data Setup to End of Write Enable*	50				ns
T5	Data Hold Time from End of $\overline{IOW}$	40				ns
T6	Width of $\overline{EOP}$ Pulse (Note 1)	100				ns
T7	$\overline{ACK}$ TRUE to $\overline{REQ}$ FALSE	25	125			ns
T8	$\overline{REQ}$ from End of $\overline{IOW}$ ($\overline{ACK}$ FALSE)	40	180			ns
T9	$\overline{REQ}$ from End of $\overline{ACK}$ ($\overline{IOW}$ FALSE)	20	170			ns
T10	$\overline{ACK}$ TRUE to READY TRUE	20	140			ns
T11	READY TRUE to $\overline{IOW}$ FALSE	70				ns
T12	$\overline{IOW}$ FALSE to READY FALSE	20	140			ns
T13	Data Hold from $\overline{ACK}$ TRUE	40				ns
T14	Data Setup to $\overline{REQ}$ TRUE	60				ns

*Write Enable is the occurrence of $\overline{IOW}$ and $\overline{DACK}$

Notes: 1. $\overline{EOP}$, $\overline{IOW}$, and $\overline{DACK}$ must be concurrently TRUE for at least T6 for proper recognition of the $\overline{EOP}$ pulse.

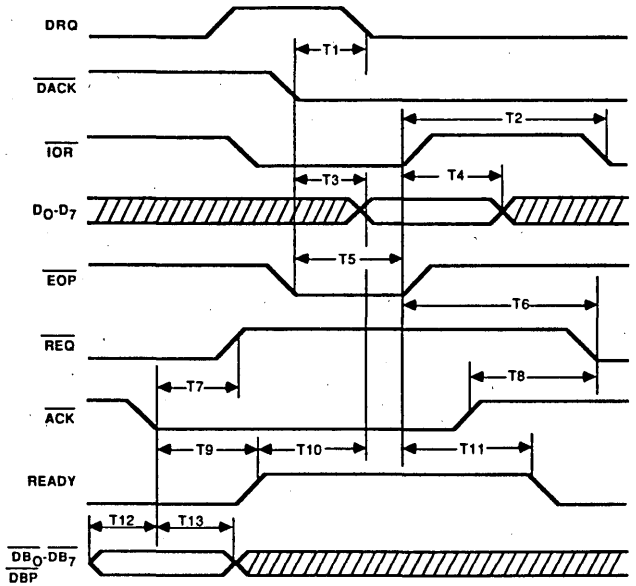


DMA Read (Block Mode) Target Receive Cycle

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	$\overline{DRQ}$ FALSE from $\overline{DACK}$ TRUE		130			ns
T2	$\overline{IOR}$ Recovery Time	120				ns
T3	Data Access Time from Read Enable*		110			ns
T4	Data hold Time from End of $\overline{IOR}$	20				ns
T5	Width of $\overline{EOP}$ Pulse (Note 1)	100				ns
T6	$\overline{IOR}$ FALSE to $\overline{REQ}$ TRUE ($\overline{ACK}$ FALSE)	30	190			ns
T7	$\overline{ACK}$ TRUE to $\overline{REQ}$ FALSE	25	125			ns
T8	$\overline{ACK}$ FALSE to $\overline{REQ}$ TRUE ($\overline{IOR}$ FALSE)	20	170			ns
T9	$\overline{ACK}$ TRUE to READY TRUE	20	140			ns
T10	READY TRUE to Valid Data		50			ns
T11	$\overline{IOR}$ FALSE to READY FALSE	20	140			ns
T12	Data Setup Time to $\overline{ACK}$	20				ns
T13	Data Hold Time from $\overline{ACK}$	50				ns

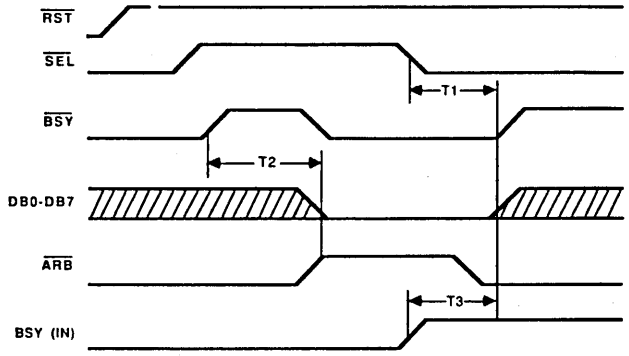
*Read Enable is the occurrence of $\overline{IOR}$ and $\overline{DACK}$

Notes: 1. $\overline{EOP}$, IOR, and $\overline{DACK}$ must be concurrently TRUE for at least T5 for proper recognition of the $\overline{EOP}$ pulse.



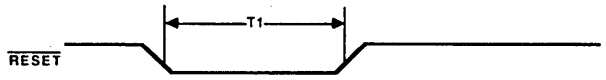
Arbitration

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	Bus Clear from $\overline{SEL}$ TRUE		600			ns
T2	Arbitrate Start drom $\overline{BSY}$ FALSE	1200	2200			ns
T3	Bus Clear from BSY FALSE		1100			ns



Reset

		Z0538010		Z0538015		
Name	Description	Min.	Max.	Min.	Max.	Units
T1	Minimum Width of Reset	200				ns



Z5380 NOTES

1) Edge triggered $\overline{RST}$ Interrupt - If the SCSI Bus is not terminated, the RST interrupt is continually generates.

2) TRUE End of DMA Interrup - The Am5380 generated an interrupt when it receives the last byte from the DMA, not when the last byte is transferred to the SCSI Bus.

3) Return to READY after $\overline{EOP}$ Interrupt When operating in Block mode DMA, the Z5380 does nor return the READY signal to a Ready condition. This locks up the bus and prevents the CPU from executing.

4) SCSI handshake after $\overline{EOP}$ occurs - If an EOP occurs when receiving data, a subsequent REQ wil cause $\overline{ACK}$ to be asserted even though no DRQ is issued.

5) During Reselection, if the Target Command Register does not reflect the current bus phase (most likely Data Out), the Reselection interrupt may get reset.

6) A phase-mismatch interrupt is not gauranteed after a Reselection for the following reasons:

-DMA MODE bit must be set in order to receive a phase mismatch interrupt

-DMA MODE bit cannot be set unless $\overline{BSY}$ is active

-$\overline{BSY}$ cannot be asserted until after the Reselection has occured

-Once $\overline{BSY}$ is asserted, the Target may assert $\overline{REQ}$ in less than 500 ns

-The phase-mismatch interrupt is generated on the active edge of $\overline{REQ}$. If the DMA MODE bit is not set before the $\overline{REQ}$ goes active, the phase-mismatch interrupt will not occur.

October 1988

Z7220A
High-Performance Graphics Display Controller

Description

The Z7220A High-performance Graphics Display Controller (HGDC) is an intelligent microprocessor peripheral designed to be the heart of a high-performance raster-scan computer graphics and character display system. Positioned between the video display memory and the microprocessor bus, the HGDC performs the tasks needed to generate the raster display and manage the display memory. Processor software overhead is minimized by the HGDC's sophisticated instruction set, graphics figure drawing, and DMA transfer capabilities. The display memory supported by the HGDC can be configured in any number of formats and sizes up to 256K 16-bit words. The display can be zoomed and panned, while partitioned screen areas can be independently scrolled. With its light pen input and multiple controller capability, the HGDC is ideal for advanced computer graphics applications.

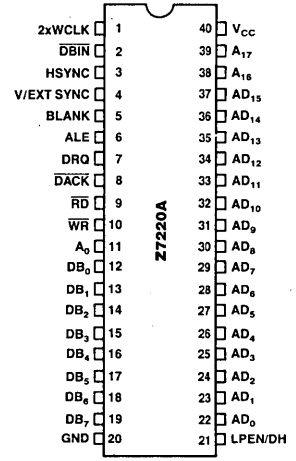
System Considerations

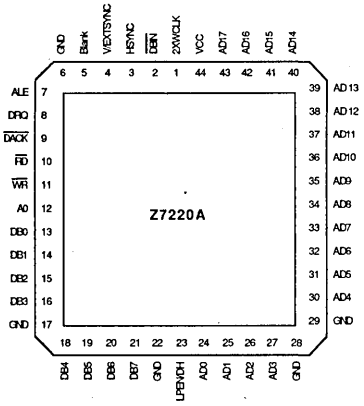
The HGDC is designed to work with a general purpose microprocessor to implement a high-performance computer graphics system. Through the division of labor established by the HGDC's design, each of the system components is used to the maximum extent through a six-level hierarchy of simultaneous tasks. At the lowest level, the HGDC generates the basic video raster timing, including sync and blanking signals. Partitioned areas on the screen and zooming are also accomplished at this level. At the next level, video display memory is modified during the figure drawing operations and data moves. Third, display memory addresses are calculated pixel by pixel as drawing progresses. Outside the HGDC at the next level, preliminary calculations are done to prepare drawing parameters. At the fifth level, the picture must be represented as a list of graphics figures drawable by the HGDC. Finally, this representation must be manipulated, stored, and communicated. By handling the first three levels, the HGDC takes care of the high-speed and repetitive tasks required to implement a graphics system.

Features

- ☐ Microprocessor Interface
  - DMA transfers
  - FIFO Command Buffering
- ☐ Display Memory Interface
  - Up to 256K words of 16 bits
  - Read-Modify-Write (RMW) Display Memory cycles as fast as 500ns
  - Dynamic RAM refresh cycles for nonaccessed memory
- ☐ Light Pen Input
- ☐ Drawing Hold Input
- ☐ External video synchronization mode
- ☐ Graphics Mode
  - Four megabit, bit-mapped display memory
- ☐ Character Mode
  - 8K character code and attributes display memory
- ☐ Mixed Graphics and Character Mode
  - 64K if all characters
  - 1 megapixel if all graphics
- ☐ Graphics Capabilities
  - Figure drawing of lines, arc/circles, rectangles, and graphics characters in 500ns per pixel
  - Display 1024-by-1024 pixels with 4 planes of color or grayscale
  - Two independently scrollable areas
- ☐ Character Capabilities
  - Auto cursor advance
  - Four independently scrollable areas
  - Programmable cursor height
  - Characters per row: up to 256
  - Character rows per screen: up to 100
- ☐ Video Display Format
  - Zoom magnification factors of 1 to 16
  - Panning
  - Command-settable video raster parameters
- ☐ Technology
  - Single +5V, NMOS, 40-pin DIP
- ☐ DMA Capability
  - Byte or word transfers
  - 4 clock periods per byte transferred
- ☐ On-chip pull-up resistor for VSYNC/EXT, HSYNC and $\overline{DACK}$, and a pull-down resistor for LPEN/DH

Pin Configuration





44-Pin Plastic Chip Carrier (PCC)
Pin Assignments

Pin Identification

Pin No.	Symbol	Direction	Function
1	2xWCLK	In	Clock Input
2	$\overline{DBIN}$	Out	Display Memory Read Input Flag
3	HSYNC	Out	Horizontal Video Sync Output
4	V/EXT SYNC	In/Out	Vertical Video Sync Output or External VSYNC Input
5	BLANK	Out	CRT Blanking Output
6	ALE ($\overline{RAS}$)	Out	Address Latch Enable Output
7	DRQ	Out	DMA Request Output
8	$\overline{DACK}$	In	DMA Acknowledge Input
9	$\overline{RD}$	In	Read Strobe Input for Microprocessor Interface
10	$\overline{WR}$	In	Write Strobe Input for Microprocessor Interface
11	A_0	In	Address Select Input for Microprocessor Interface
12–19	DB_0–DB_7	In/Out	Bidirectional Data Bus to Host Microprocessor
20	GND	—	Ground
21	LPEN/DH	In	Light Pen Detect Input/Drawing Hold Input
22–34	AD_0–AD_{12}	In/Out	Address and Data Lines to Display Memory
35–37	AD_{13}–AD_{15}	In/Out	Utilization Varies with Mode of Operation
38	A_{16}	Out	Utilization Varies with Mode of Operation
39	A_{17}	Out	Utilization Varies with Mode of Operation
40	V_{CC}	—	+5V ± 10% Power Supply

Character Mode Pin Utilization

Pin No.	Symbol	Direction	Function
35–37	AD_{13}–AD_{15}	Out	Line Counter Bits 0 to 2 Outputs
38	A_{16}	Out	Line Counter Bit 3 Output
39	A_{17}	Out	Cursor Output and Line Counter Bit 4

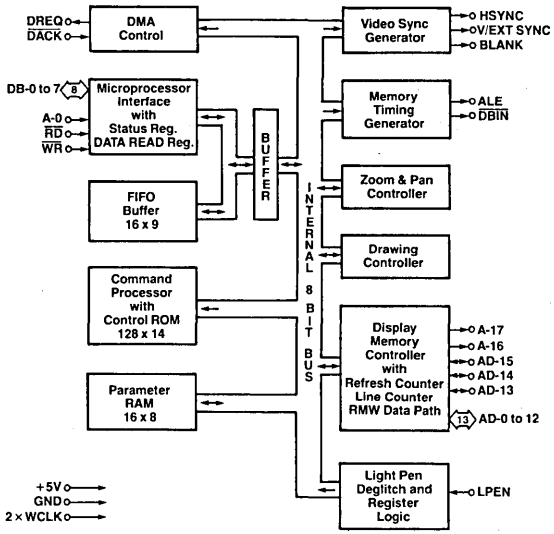
Mixed Mode Pin Utilization

Pin No.	Symbol	Direction	Function
35–37	AD_{13}–AD_{15}	In/Out	Address and Data Bits 13 to 15
38	A_{16}	Out	Attribute Blink and Clear Line Counter Output
39	A_{17}	Out	Cursor and Bit-map Area Flag Output

Graphics Mode Pin Utilization

Pin No.	Symbol	Direction	Function
35–37	AD_{13}–AD_{15}	In/Out	Address and Data Bits 13 to 15
38	A_{16}	Out	Address Bit 16 Output
39	A_{17}	Out	Address Bit 17 Output

Block Diagram



HGDC Components

Microprocessor Bus Interface

Control of the HGDC by the system microprocessor is achieved through an 8-bit bidirectional interface. The status register is readable at any time. Access to the FIFO buffer is coordinated through flags in the status register and operates independently of the various internal HGDC operations, due to the separate data bus connecting the interface and the FIFO buffer.

Command Processor

The contents of the FIFO are interpreted by the command processor. The command bytes are decoded, and the succeeding parameters are distributed to their proper destinations within the HGDC. The command processor yields to the bus interface when both access the FIFO simultaneously.

DMA Control

The DMA control circuitry in the HGDC coordinates transfers over the microprocessor interface when using an external DMA controller. The DMA Request and Acknowledge handshake lines directly interface with a DMA controller, so that display data can be moved between the microprocessor memory and the display memory.

Parameter RAM

The 16-byte RAM stores parameters that are used repetitively during the display and drawing processes. In character mode, this RAM holds four sets of partitioned display area parameters; in graphics mode, the drawing pattern and graphics character take the place of two of the sets of parameters.

Video Sync Generator

Based on the clock input, the sync logic generates the raster timing signals for almost any interlaced, non-interlaced, or "repeat field" interlaced video format. The generator is programmed during the idle period following a reset. In video sync slave mode, it coordinates timing between multiple HGDCs.

Memory Timing Generator

The memory timing circuitry provides two memory cycle types: a two-clock period refresh cycle and the read-modify-write (RMW) cycle which takes four clock periods. The memory control signals needed to drive the display memory devices are easily generated from the HGDC's ALE and DBIN outputs.

Zoom and Pan Controller

Based on the programmable zoom display factor and the display area entries in the parameter RAM, the zoom and pan controller determines when to advance to the next memory address for display refresh and when to go on to the next display area. A horizontal zoom is produced by slowing down the display refresh rate while maintaining the video sync rates. Vertical zoom is accomplished by repeatedly accessing each line a number of times equal to the horizontal repeat. Once the line count for a display area is exhausted, the controller accesses the starting address and line count of the next display area from the parameter RAM. The system microprocessor, by modifying a display area starting address, can pan in any direction, independently of the other display areas.

Drawing Controller

The drawing processor contains the logic necessary to calculate the addresses and positions of the pixels of the various graphics figures. Given a starting point and the appropriate drawing parameters, the drawing controller needs no further assistance to complete the figure drawing.

Display Memory Controller

The display memory controller's tasks are numerous. Its primary purpose is to multiplex the address and data information in and out of the display memory. It also contains the 16-bit logic unit used to modify the display memory contents during RMW cycles, the character mode line counter, and the refresh counter for dynamic RAMS. The memory controller apportions the video field time between the various types of cycles.

Light Pen Deglitcher/Drawing Hold

Only if two rising edges on the light pen input occur at the same point during successive video fields are the pulses accepted as a valid light pen detection. A status bit indicates to the system microprocessor that the light pen register contains a valid address. If this input is held high for a period greater than four 2xWCLK cycles, drawing execution is halted.

Programmer's View of HGDC

The HGDC occupies two addresses on the system microprocessor bus through which the HGDC's status register and FIFO are accessed. Commands and parameters are written into the HGDC's FIFO and are differentiated based on address bit A_0. The status register or the FIFO can be read as selected by the address line.

A0	READ	WRITE
0	Status Register	Parameter Into FIFO
1	FIFO Read	Command Into FIFO

HGDC Microprocessor Bus Interface Registers

Commands to the HGDC take the form of a command byte followed by a series of parameter bytes as needed for specifying the details of the command. The command processor decodes the commands, unpacks the parameters, loads them into the appropriate registers within the HGDC, and initiates the required operations.

The commands available in the HGDC can be organized into five categories as described in the following section.

HGDC Commands Summary

Video Control Commands

1. RESET1 — Resets the HGDC to its idle state. Resynchronizes video timing. Blanks the display.
2. RESET2 — Resets the HGDC to its idle state. Does not resynchronize video timing. Blanks the display.
3. RESET3 — Resets the HGDC to its idle state. Does not resynchronize video timing. Does not blank the display.
4. SYNC — Specifies the video display format.
5. VSYNC — Selects master or slave video synchronization mode.
6. CCHAR — Specifies the cursor and character row heights.

Display Control Commands

1. START — Ends Idle mode and unblanks the display.
2. BLANK1 — Controls the blanking and unblanking of the display, along with video resynchronization.
3. BLANK2 — Controls the blanking and unblanking of the display. Does not blank the display.
4. ZOOM — Specifies zoom factors for the display and graphics characters writing.
5. CURS — Sets the position of the cursor in display memory.
6. PRAM — Defines starting addresses and lengths of the display areas and specifies the eight bytes for the graphics character.
7. PITCH — Specifies the width of the X dimension of display memory.

Drawing Control Commands

1. WDAT — Writes data words or bytes into display memory.
2. MASK — Sets the mask register contents.
3. FIGS — Specifies the parameters for the drawing controller.
4. FIGD — Draws the figure as specified above.
5. GCHRD — Draws the graphics character into display memory.

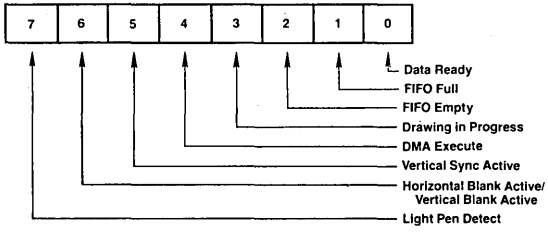
Data Read Commands

1. RDAT — Reads data words or bytes from display memory.
2. CURD — Reads the cursor position.
3. LPRD — Reads the light pen address.

DMA Control Commands

1. DMAR — Requests a DMA read transfer.
2. DMAW — Requests a DMA write transfer.

Status Register Flags



Status Register (SR)

SR-7: Light Pen Detect

When this bit is set to 1, the light pen address (LAD) register contains a deglitched value that the system microprocessor may read. This flag is reset after the 3-byte LAD is moved into the FIFO in response to the light pen read command.

SR-6: Horizontal Blank Active/Vertical Blank Active

A 1 value for this flag signifies that horizontal retrace blanking or vertical retrace blanking is currently underway dependent on the status of the VH bit in SYNC or the RESETx parameter 6.

SR-5: Vertical Sync

Vertical retrace sync occurs while this flag is a 1. The vertical sync flag coordinates display format modifying commands to the blanked interval surrounding vertical sync. This eliminates display disturbances.

SR-4: DMA Execute

This bit is a 1 during DMA data transfers.

SR-3: Drawing in Progress

While the HGDC is drawing a graphics figure, this status bit is a 1.

SR-2: FIFO Empty

This bit and the FIFO-full flag coordinate system microprocessor accesses with the HGDC FIFO. When it is 1, the Empty flag ensures that all the commands and parameters previously sent to the HGDC have been interpreted.

SR-1: FIFO Full

A 1 at this flag indicates a full FIFO in the HGDC. A 0 ensures that there is room for at least one byte. This flag needs to be checked before each write into the HGDC.

SR-0: Data Ready

When this flag is a 1, it indicates that a byte is available to be read by the system microprocessor. This bit must be tested before each read operation. It drops to a 0 while the data is transferred from the FIFO into the microprocessor interface data register.

FIFO Operation and Command Protocol

The first-in, first-out buffer (FIFO) in the HGDC handles the command dialogue with the system microprocessor. This flow of information uses a half-duplex technique, in which the single 16-location FIFO is used for both directions of data movement, one direction at a time. The FIFO's direction is controlled by the system microprocessor through the HGDC's command set. The host microprocessor coordinates these transfers by checking the appropriate status register bits.

The command protocol used by the HGDC requires differentiation of the first byte of a command sequence from the succeeding bytes. The first byte contains the operation code and the remaining bytes carry parameters. Writing into the HGDC causes the FIFO to store a flag value alongside the data byte to signify whether the byte was written into the command or the parameter address. The command processor in the HGDC tests this bit as it interprets the entries in the FIFO.

The receipt of a command byte by the command processor marks the end of any previous operation. The number of parameter bytes supplied with a command is cut short by the receipt of the next command byte. A read operation from the HGDC to the microprocessor can be terminated at any time by the next command.

The FIFO changes direction under the control of the system microprocessor. Commands written into the HGDC always put the FIFO into write mode if it was not in it already. If it was in read mode, any read data in the FIFO at the time of the turnaround is lost. Commands which require an HGDC response, such as RDAT, CURD and LPRD, put the FIFO into read mode after the command is interpreted by the HGDC's command processor. Any commands and parameters behind the read-evoking command are discarded when the FIFO direction is reversed.

Read-Modify-Write Cycle

Data transfers between the HGDC and the display memory are accomplished using a read-modify-write (RMW) memory cycle. The four-clock period timing of the RMW cycle is used to: 1) output the address, 2) read data from the memory, 3) modify the data, and 4) write the modified data back into the initially selected memory address. This type of memory cycle is used for all interactions with display memory including DMA transfers, except for the two-clock period display and RAM refresh cycles.

The operations performed during the modify portion of the RMW cycle merit additional explanation. The circuitry in the HGDC uses three main elements: the Pattern register, the Mask register, and the 16-bit Logic unit. The Pattern register holds the data pattern to be moved into memory. It is loaded by the WDAT parameters or, during drawing, from the parameter RAM. The Mask register contents determine which bits of the read data will be modified. Based on the contents of these registers, the Logic unit performs the selected operations of REPLACE, COMPLEMENT, SET, or CLEAR on the data read from display memory.

The Pattern register contents are ANDed with the Mask register contents to enable the actual modification of the memory read data, on a bit-by-bit basis. For graphics drawing, one bit at a time from the Pattern register is combined with the Mask. When ANDed with the bit set to a 1 in the Mask register, the proper single pixel is modified by the Logic unit. For the next pixel in the figure, the next bit in the Pattern register is selected and the Mask register bit is moved to identify the pixel's location within the word. The Execution word address pointer register, EAD, is also adjusted as required to address the word containing the next pixel.

In character mode, all of the bits in the Pattern register are used in parallel to form the respective bits of the modify data word. Since the bits of the character code word are used in parallel, unlike the one-bit-at-a-time graphics drawing process, this facility allows any or all of the bits in a memory word to be modified in one RMW memory cycle. The Mask register must be loaded with ones in the positions where modification is to be permitted.

The Mask register can be loaded in either of two ways. In graphics mode, the CURS command contains a 4-bit dAD field to specify the dot address. The command processor converts this parameter into the 1-of-16 format used in the Mask register for figure drawing. A full 16 bits can be loaded into the Mask register using the MASK command. In addition to the character mode use mentioned above, the 16-bit MASK load is convenient in graphics mode when all of the pixels of a word are to be set to the same value.

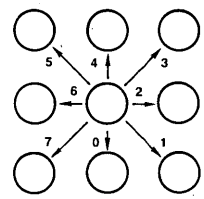
The Logic unit combines the data read from display memory, the Pattern register, and the Mask register to generate the data to be written back into display memory. Any one of four operations can be selected: REPLACE, COMPLEMENT, CLEAR or SET. In each case, if the respective Mask bit is 0, that particular bit of the read data is returned to memory unmodified. If the Mask bit is 1, the modification is enabled. With the REPLACE operation, the Pattern register data simply takes the place of the read data for modification enabled bits. For the other three operations, a 0 in the modify data allows the read data bit to be returned to memory. A 1 value causes the specified operation to be performed in the bit positions with set Mask bits.

Figure Drawing

The HGDC draws graphics figures at the rate of one pixel per read-modify-write (RMW) display memory cycle. These cycles take four clock periods to complete. At a clock frequency of 8MHz, this is equal to 500ns. During the RMW cycle the HGDC simultaneously calculates the address and position of the next pixel to be drawn.

The graphics figure drawing process depends on the display memory addressing structure. Groups of 16 horizontally adjacent pixels form the 16-bit words which are handled by the HGDC. Display memory is organized as a linearly addressed space of these words. Addressing of individual pixels is handled by the HGDC's internal RMW logic.

During the drawing process, the HGDC finds the next pixel of the figure which is one of the eight nearest neighbors of the last pixel drawn. The HGDC assigns each of these eight directions a number from 0 to 7, starting with straight down and proceeding counterclockwise.



Drawing Directions

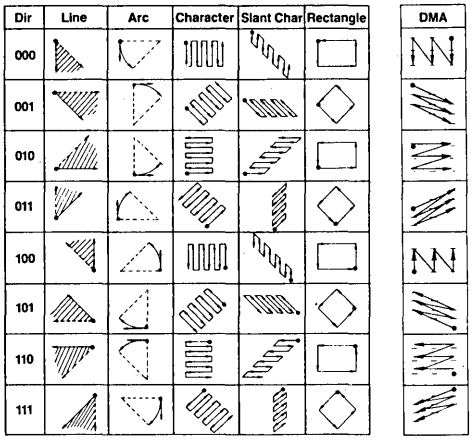
Figure drawing requires the proper manipulation of the address and the pixel bit position according to the drawing direction to determine the next pixel of the figure. To move to the word above or below the current one, it is necessary to subtract or add the number of words per line in display memory. This parameter is called the pitch. To move to the word to either side, the Execute word address cursor, EAD, must be incremented or decremented as the dot address pointer bit reaches the LSB or the MSB of the Mask register. To move to a pixel within the same word, it is necessary to rotate the dot address pointer register to the right or left. The table below summarizes these operations for each direction.

Dir	Operations to Address the Next Pixel
000	EAD − P→EAD
001	EAD − P→EAD dAD (MSB) = 1:EAD − 1→EAD dAD→LR
010	dAD (MSB) = 1:EAD − 1→EAD dAD→LR
011	EAD − P→EAD dAD (MSB) = 1:EAD − 1→EAD dAD→LR
100	EAD − P→EAD
101	EAD − P→EAD dAD (LSB) = 1:EAD − 1→EAD dAD→RR
110	dAD (LSB) = 1:EAD − 1→EAD dAD→RR
111	EAD − P→EAD dAD (LSB) = 1:EAD − 1→EAD dAD→RR

Note: P = Pitch, LR = Left Rotate, RR = Right Rotate, EAD = Execute Word Address, and dAD = Dot Address stored in the Mask register.

Whole word drawing is useful for filling areas in memory with a single value. By setting the Mask register to all 1s with the MASK command, both the LSB and MSB of the dAD will always be 1, so that the EAD value will be incremented or decremented for each cycle regardless of direction. One RMW cycle will be able to affect all 16 bits of the word for any drawing type. One bit in the Pattern register is used per RMW cycle to write all the bits of the word to the same value. The next Pattern bit is used for the word, etc.

For the various figures, the effect of the initial direction upon the resulting drawing is shown below:



Note that during line drawing, the angle of the line may be anywhere within the shaded octant defined by the DIR value. Arc drawing starts in the direction initially specified by the DIR value and veers into an arc as drawing proceeds. An arc may be up to 45° in length. DMA transfers are done on word boundaries only, and follow the arrows indicated in the table to find successive word addresses. The slanted paths for DMA transfers indicate the HGDC changing both the X and Y components of the word address when moving to the next word. It does not follow a 45° diagonal path by pixels.

Drawing Parameters

In preparation for graphics figure drawing, the HGDC's Drawing processor needs the figure type, direction and drawing parameters, the starting pixel address, and the pattern from the microprocessor. Once these are in place within the HGDC, the Figure Draw command, FIGD, initiates the drawing operation. From that point on, the system microprocessor is not involved in the drawing process. The HGDC Drawing controller coordinates the RMW circuitry and address registers to draw the specified figure pixel by pixel.

The algorithms used by the processor for figure drawing are designed to optimize its drawing speed. To this end, the specific details about the figure to be drawn are reduced by the microprocessor to a form conducive to high-speed address calculations within the HGDC. In this way the repetitive, pixel-by-pixel calculations can be done quickly, thereby minimizing the overall figure drawing time. The table below summarizes the parameters.

Drawing Type	DC	D	D2	D1	DM
Initial Value ①	0	8	8	−1	−1
Line	$\lvert\Delta I\rvert$	$2\lvert\Delta D\rvert - \lvert\Delta I\rvert$	$2(\lvert\Delta D\rvert - \lvert\Delta I\rvert)$	$2\lvert\Delta D\rvert$	–
Arc ②	$r\sin\phi$	r−1	2(r−1)	−1	$r\sin\theta\downarrow$
Rectangle	3	A−1	B−1	−1	A−1
Area Fill	B−1	A	A	–	–
Graphic Character ③	B−1	A	A	–	–
Read & Write Data	W−1	–	–	–	–
DMAW	D−1	C−1	–	–	–
DMAR	D−1	C−1	(C−1)/2†	–	–

Notes: All numbers are shown in base 10 for convenience. The HGDC accepts base 2 numbers (2s complement notation) where appropriate.

① Initial values for the various parameters remain as each drawing process ends.

② Circles are drawn with 8 arcs, each of which span 45°, so that $\sin\phi = 1/\sqrt{2}$ and $\sin\theta = 0$.

③ Graphic characters are a special case of bit-map area filling in which B and A ≤ 8. If A = 8 there is no need to load D and D2.

Symbol Definitions

−1 = All ONES value.
– = No parameter bytes sent to HGDC for this parameter.
ΔI = The larger at Δx or Δy.
ΔD = The smaller at Δx or Δy.
r = Radius of curvature, in pixels.
ϕ = Angle from major axis to end of the arc. $\phi \leq 45°$.
θ = Angle from major axis to start of the arc. $\theta \leq 45°$.
↑ = Round up to the next higher integer.
↓ = Round down to the next lower integer.
A = Number of pixels in the initially specified direction.
B = Number of pixels in the direction at right angles to the initially specified direction.
W = Number of words to be accessed.
C = Number of bytes to be transferred in the initially specified direction. (Two bytes per word if word transfer mode is selected.)
D = Number of words to be accessed in the direction at right angles to the initially specified direction.
DC = Drawing count parameter which is one less than the number of RMW cycles to be executed.
DM = Dots masked from drawing during arc drawing.
† = Needed only for word reads.

Graphics Character Drawing

Graphics characters can be drawn into display memory pixel by pixel. The up to 8-by-8 character display is loaded into the HGDC's parameter RAM by the system microprocessor. Consequently, there are no limitations on the character set used. By varying the drawing parameters and drawing direction, numerous drawing options are available. In area fill applications, a character can be written into display memory as many times as desired without reloading the parameter RAM.

Once the parameter RAM has been loaded with up to eight graphics character bytes by the appropriate PRAM command, the GCHRD command can be used to draw the bytes into display memory starting at the cursor. The zoom magnification factor for writing, set by the ZOOM command, controls the size of the character written into the display memory in integer multiples of 1 through 16. The bit values in the PRAM are repeated horizontally and vertically the number of times specified by the zoom factor.

The movement of these PRAM bytes to the display memory is controlled by the parameters of the FIGS command.

Based on the specified height and width of the area to be drawn, the parameter RAM is scanned to fill the required area.

For an 8-by-8 graphics character, the first pixel drawn uses the LSB of RA-15, the second pixel uses bit 1 of RA-15, and so on, until the MSB of RA-15 is reached.

The HGDC jumps to the corresponding bit in RA-14 to continue the drawing. The progression then advances toward the LSB of RA-14. This snaking sequence is continued for the other 6 PRAM bytes. This progression matches the sequence of display memory addresses calculated by the drawing processor as shown above. If the area is narrower than 8 pixels wide, the snaking will advance to the next PRAM byte before the MSB is reached. If the area is less than 8 lines high, fewer bytes in the parameter RAM will be scanned. If the area is larger than 8 by 8, the HGDC will repeat the contents of the parameter RAM in two dimensions, as required to fill the area with the 8-by-8 mosaic. (Fractions of the 8-by-8 pattern will be used to fill areas which are not multiples of 8 by 8.)

Parameter RAM Contents: RAM Address RA-0 to RA-15

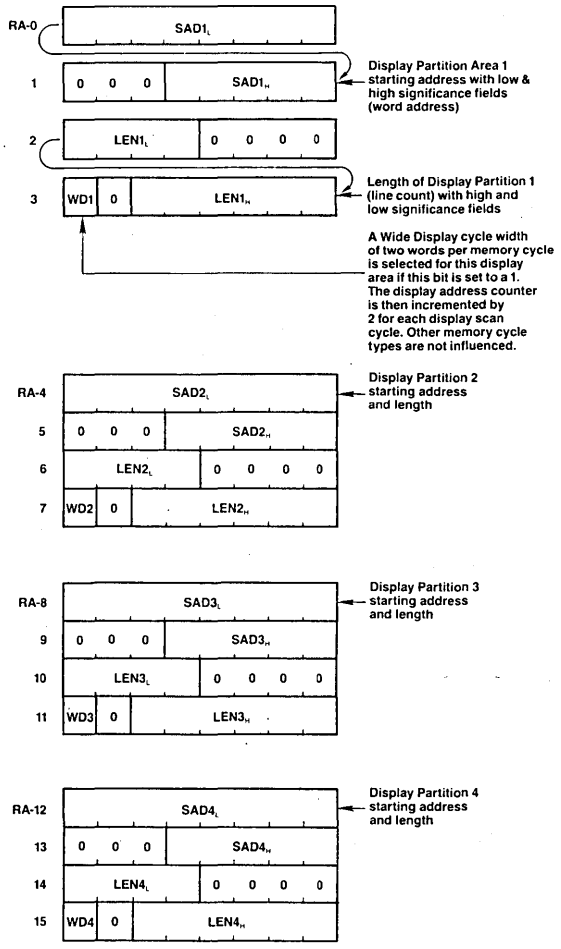
The parameters stored in the parameter RAM, PRAM, are available for the HGDC to refer to repeatedly during figure drawing and raster-scanning. In each mode of operation the values in the PRAM are interpreted by the HGDC in a predetermined fashion. The host microprocessor must load the appropriate parameters into the proper PRAM locations. PRAM loading command allows the host to write into any location of the PRAM and transfer as many bytes as desired. In this way any stored parameter byte or bytes may be changed without influencing the other bytes.

The PRAM stores two types of information. For specifying the details of the display area partitions, blocks of four bytes are used. The four parameters stored in each block include the starting address in display memory of each display area, and its length. In addition, there are two mode bits for each area which specify whether the area is a bit-mapped graphics area or a coded-character area, and whether a 16-bit or a 32-bit wide display cycle is to be used for that area.

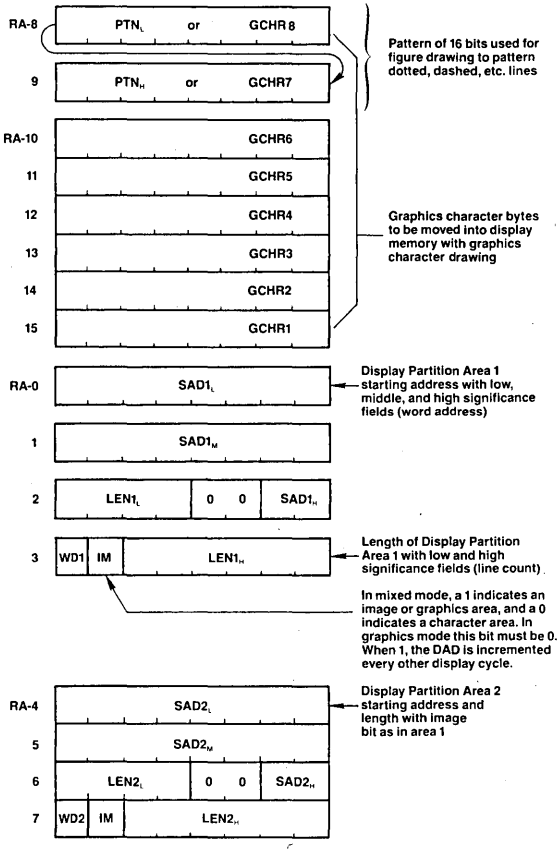
The other use for the PRAM contents is to supply the pattern for figure drawing when in a bit-mapped graphics area or mode. In these situations, PRAM bytes 8 through 16 are reserved for this patterning information. For line, arc, and rectangle drawing (linear figures) locations 8 and 9 are loaded into the Pattern register to allow the HGDC to draw dotted, dashed, etc. lines. For area filling and graphics bit-mapped character drawing locations 8 through 15 are referenced for the pattern or character to be drawn.

Details of the bit assignments are shown for the various modes of operation.

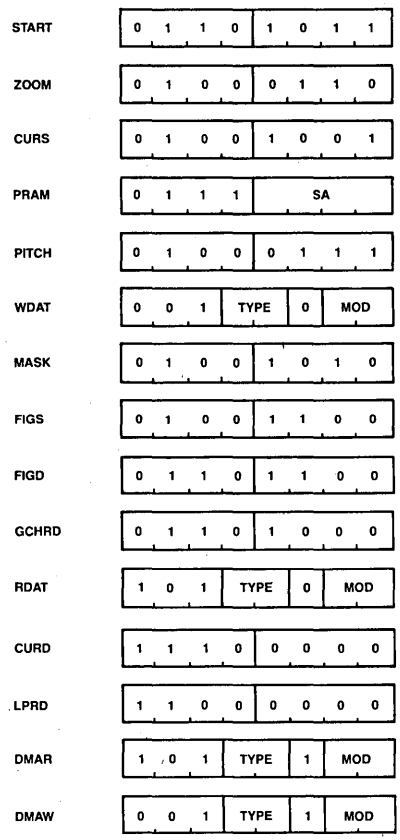
Character Mode



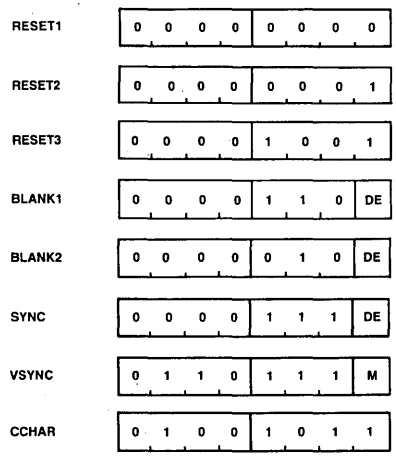
Graphics and Mixed Graphics and Character Modes



Command Bytes Summary

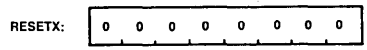


Command Bytes Summary



Video Control Commands

Reset

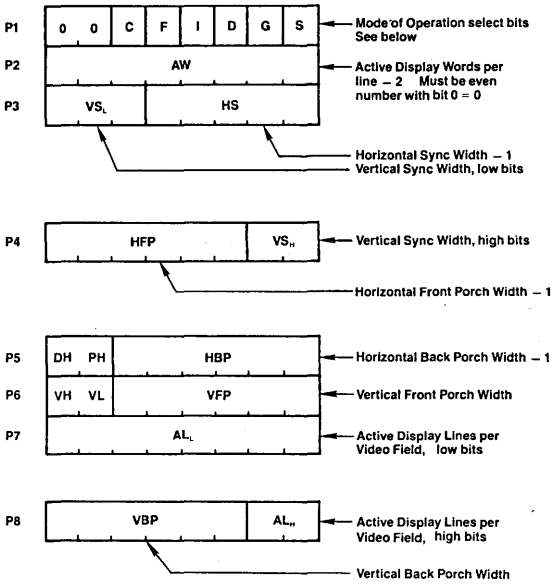


Blank the display, enter Idle mode, and initialize within the HGDC:
- FIFO
- Command Processor
- Internal Counters

This command can be executed at any time and does not modify any of the parameters already loaded into the HGDC.

If followed by parameter bytes, this command also sets the sync generator parameters as described below. Idle mode is exited with the START command.

RESET1: Resync video timing in slave mode.
RESET2: Blank the display and do not resync.
RESET3: Unblank the display and do not resync.



In graphics mode, a word is a group of 16 pixels. In character mode, a word is one character code and its attributes, if any. The number of active words per line must be an even number from 2 to 256. An all-zero parameter value selects a count equal to 2^n where n = number of bits in the parameter field for vertical parameters. All horizontal widths are counted in display words. All vertical intervals are counted in lines.

If the Drawing Hold (DH) is set to one, pin 21 (LPEN/DH) is used as the drawing hold control pin. When the input to LPEN/DH is held high for over four 2 x WCLK clocks, the drawing address output is temporarily held and the display address is output.

The HGDC allows an even or odd number of lines per frame. Selection is via the VL flag, the seventh bit of the sixth parameter byte following a RESET or SYNC command. When VL is 0, an odd number of display lines is generated.

VL	Number of lines in interlaced mode
0	Odd, as in 7220
1	Even

When VH = 0, status operation is as in the 7220.

VH	Blank Status Bit Definition
0	Status register bit 6 indicates Horizontal Blank
1	Status register bit 6 indicates Vertical Blank

PH is the most significant bit (9) of the display pitch parameter. Use the PITCH command to set the lower eight bits.

SYNC Generator Period Constraints

Horizontal Back Porch Constraints

1. In general:
   HBP ⩾ 3 Display Word Cycles (6 clock cycles).
2. If the Image bit or WD mode changes within one video field:
   HBP ⩾ 5 Display Word Cycles (10 clock cycles).
3. If interlaced, mixed mode, or split screen is used:
   HBP ⩾ 5 Display Word Cycles (10 clock cycles).

Horizontal Front Porch Constraints

1. In general:
   HFP ⩾ 2 Display Word Cycles (4 clock cycles).
2. If the HGDC is used in the video sync Slave mode:
   HFP ⩾ 4 Display Word Cycles (8 clock cycles).
3. If the Light Pen is used:
   HFP ⩾ 6 Display Word Cycles (12 clock cycles).
4. If interlaced mode, DMA, or ZOOM is used:
   HFP ⩾ 3 Display Word Cycles (6 clock cycles).

Horizontal SYNC Constraints

1. If Interlaced display mode is used:
   HS ⩾ 5 Display Word Cycles (6 clock cycles).
2. If DRAM Refresh is enabled:
   HS ⩾ 2 Display Word Cycles (4 clock cycles).

Modes of Operation Bits

C	G	Display Mode
0	0	Mixed Graphics & Character
0	1	Graphics Mode
1	0	Character Mode
1	1	Invalid

I	S	Video Framing
0	0	Non-interlaced
0	1	Invalid
1	0	Interlaced Repeat Field for Character Displays
1	1	Interlaced

Repeat Field Framing: 2 field sequence with 1/2 line offset between otherwise identical fields.

Interlaced Framing: 2 field sequence with 1/2 line offset. Each field displays alternate lines.

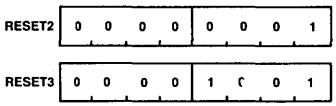
Non-interlaced Framing: 1 field brings all the information to the screen.

D	Dynamic RAM Refresh Cycles Enable
0	No Refresh — Static RAM
1	Refresh — Dynamic RAM

Dynamic RAM refresh is important when high display zoom factors or DMA are used in such a way that not all of the rows in the RAMs are regularly accessed during display raster generation and for otherwise inactive display memory.

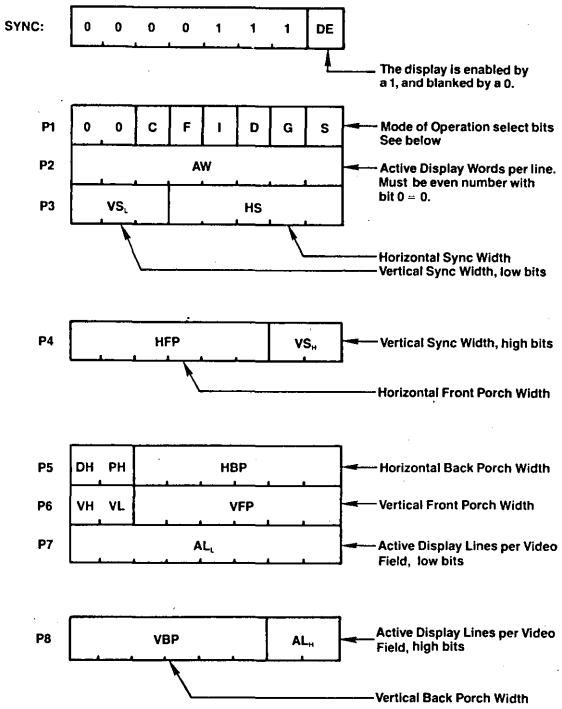
F	Drawing Time Window
0	Drawing during active display time and retrace blanking
1	Drawing only during retrace blanking

Access to display memory can be limited to retrace blanking intervals only, so that no disruptions of the image are seen on the screen.



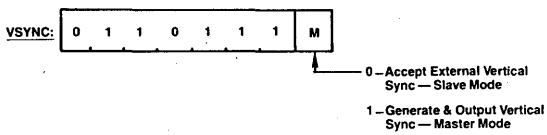
Both commands allow a reset while preventing reinitialization of the internal sync generator by an external sync source (slave mode).

SYNC Format Specify



This command also loads parameters into the sync generator. The various parameter fields and bits are identical to those at the RESET command. The HGDC is not reset nor does it enter idle mode.

Vertical Sync Mode



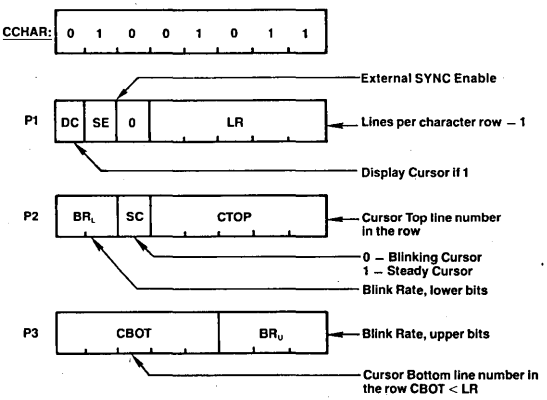
When using two or more HGDCs to contribute to one image, one HGDC is defined as the master sync generator, and the others operate as its slaves. The VSYNC pins of all HGDCs are connected together.

Slave Mode Operation

A few considerations should be observed when synchronizing two or more HGDCs to generate overlayed video via the V/EXT SYNC pin. As mentioned above, the Horizontal Front Porch (HFP) must be four or more display cycles wide. This is equivalent to eight or more clock cycles. This gives the slave HGDCs time to initialize their internal video sync generators to the proper point in the video field to match the incoming vertical sync pulse (VSYNC). This resetting of the generator occurs just after the end of the incoming VSYNC pulse, during the HFP interval. Enough time during HFP is required to allow the slave HGDC to complete the operation before the start of the HSYNC interval.

Once the HGDCs are initialized and set up as master and slaves, they must be given time to synchronize. It is a good idea to watch the VSYNC status bit of the master HGDC and wait until after one or more VSYNC pulses have been generated before the display process is started. The START command will begin the active display of data and will end the video synchronization process, so be sure there has been at least one VSYNC pulse generated to which the slaves can synchronize.

Cursor and Character Characteristics

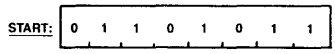


In graphics mode, LR should be set to 0. The blink rate parameter controls both the cursor and attribute blink rates. The cursor blink-on time = blink-off time = 2 x BR (video frames). The attribute blink rate is always one-half the cursor rate but with a $^3/_4$-on-$^1/_4$-off duty cycle. **All three parameter bytes must be output for interlaced displays, regardless of mode.** For interlaced displays in graphics mode, the parameter $BR_L = 3$.

When SE = 0, the HGDC, in slave mode, detects the falling edge of EX. SYNC on the first frame. When SE = 1, the HGDC, in slave mode, detects the falling edge of EX. SYNC on every frame.

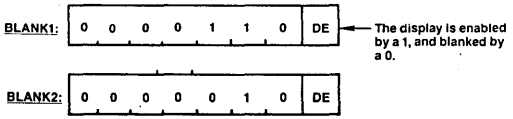
Display Control Commands

Start Display and End Idle Mode



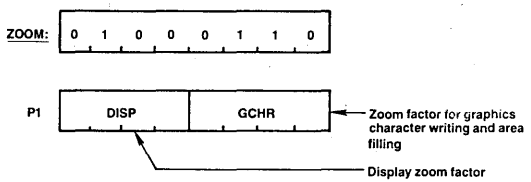
The START command generates the video signals as specified by the RESETX or SYNC command.

Display Blanking Control



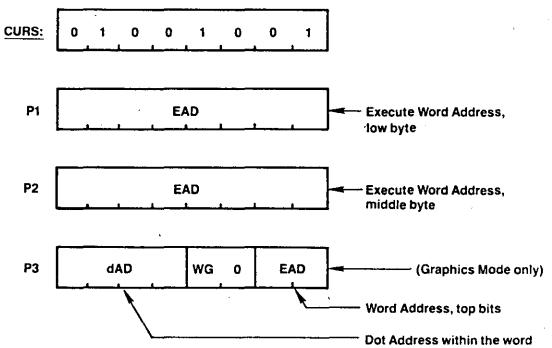
BLANK 2 does not cause the resyncing of an HGDC in slave mode. BLANK 1 does cause the resyncing of an HGDC in slave mode.

Zoom Factors Specify



Zoom magnification factors of 1 through 16 are available using codes 0 through 15, respectively.

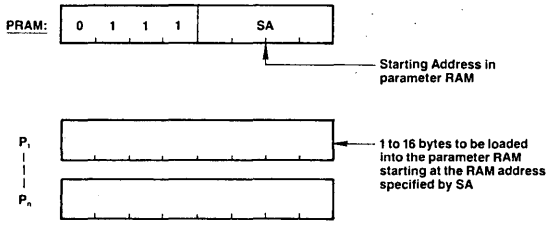
Cursor Position Specify



In character mode, the third parameter byte is not needed. The cursor is displayed for the word time in which the display scan address (DAD) equals the cursor address. In graphics mode, the cursor word address specifies the word containing the starting pixel of the drawing; the dot address value specifies the pixel within that word.

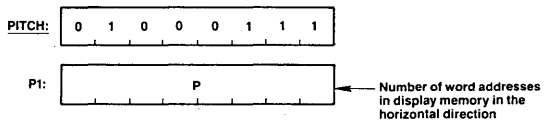
When the WG bit is set to one, any data following the WDAT command is written as is. When the WG bit is set to zero, the pattern written is determined by the least significant bit of each parameter byte following the WDAT command. This bit is expanded into 16 identical bits which form the pattern.

Parameter RAM Load



From the starting address SA, any number of bytes may be loaded into the parameter RAM at incrementing addresses, up to location 15. The sequence of parameter bytes is determined by the next command byte entered into the FIFO. The parameter RAM stores 16 bytes of information in predefined locations which differ for graphics and character modes. See the parameter RAM discussion for bit assignments.

Pitch Specification

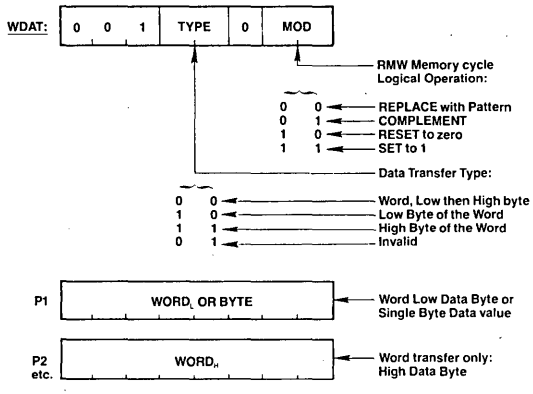


This value is used during drawing by the drawing processor to find the word directly above or below the current word, and during display to find the start of the next line.

The Pitch parameter (width of display memory) is set by two different commands. In addition to the PITCH command, the RESET (or SYNC) command also sets the pitch value. The "active-words-per-line" parameter, which specifies the width of the raster-scan display, also sets the pitch of the display memory. Note that the AW value is two less than the display window width. The PITCH command must be used to set the proper memory width larger than the window width.

Drawing Control Commands

Write Data into Display Memory



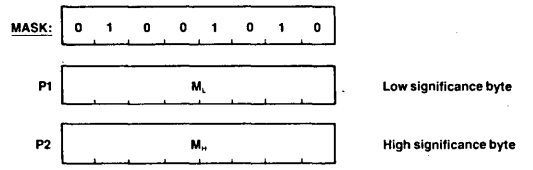
Upon receiving a set of parameters (two bytes for a word transfer, one for a byte transfer), one RMW cycle into video memory is done at the address pointed to by the cursor EAD. The EAD pointer is advanced to the next word, according to the previously specified direction. More parameters can then be accepted.

For byte writes, the unspecified byte is treated as all zeros during the RMW memory cycle.

In graphics bit-map situations, only the LSB of the WDAT parameter bytes is used as the pattern in the RMW operations. Therefore it is possible to have only an all ones or all zeros pattern. If the WG bit of the third parameter of the CURS command is set to one, any byte following the WDAT command is written as is. In coded character applications all the bits of the WDAT parameters are used to establish the drawing pattern.

The WDAT command operates differently from the other commands which initiate RMW cycle activity. It requires parameters to set up the Pattern register while the other commands use the stored values in the parameter RAM. Like all of these commands, the WDAT command must be preceded by a FIGS command and its parameters. Only the first three parameters need be given following the FIGS opcode to set up the type of drawing, the DIR direction, and the DC value. The DC parameter + 1 will be the number of RMW cycles done by the HGDC with the first set of WDAT parameters. Additional sets of WDAT parameters will see a DC value of 0 which will cause only one RMW cycle to be executed per set of parameters.

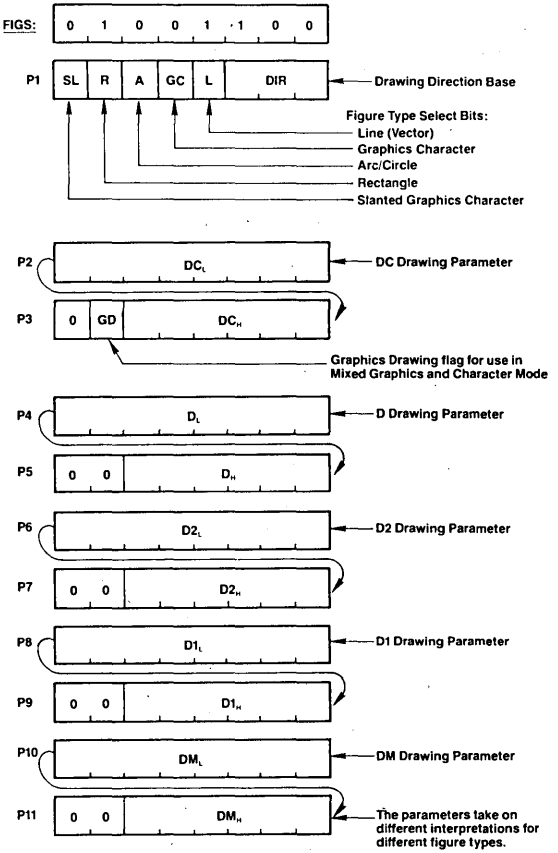
Mask Register Load



This command sets the value of the 16-bit Mask register of the figure drawing processor. The Mask register controls which bits can be modified in the display memory during a read-modify-write cycle.

The Mask register is loaded both by the MASK command and the third parameter byte of the CURS command. The MASK command accepts two parameter bytes to load a 16-bit value into the Mask register. All 16 bits can be individually one or zero, under program control. The CURS command, on the other hand, puts a 1-of-16 pattern into the Mask register based on the value of the Dot Address value, dAD. If normal single-pixel-at-a-time graphics figure drawing is desired, there is no need to do a MASK command at all since the CURS command will set up the proper pattern to address the proper pixels as drawing progresses. For coded character DMA, and screen setting and clearing operations using the WDAT command, the MASK command should be used after the CURS command if its third parameter byte has been output. The Mask register should be set to all ones for any "word-at-a-time" operation.

Figure Drawing Parameters Specify

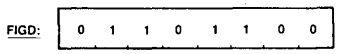


Valid Figure Type Select Combinations

SL	R	A	GC	L	Operation
0	0	0	0	0	Character Display Mode Drawing, Individual Dot Drawing, DMA, WDAT, and RDAT
0	0	0	0	1	Straight Line Drawing
0	0	0	1	0	Graphics Character Drawing and Area Filling with Graphics Character Pattern
0	0	1	0	0	Arc and Circle Drawing
0	1	0	0	0	Rectangle Drawing
1	0	0	1	0	Slanted Graphics Character Drawing and Slanted Area Filling

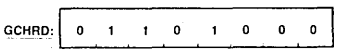
Only these bit combinations assure correct drawing operation.

Figure Draw Start



On execution of this instruction, the HGDC loads the parameters from the parameter RAM into the drawing processor and starts the drawing process at the pixel pointed to by the cursor, EAD, and the dot address, dAD.

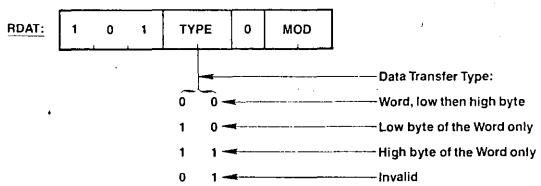
Graphics Character Draw and Area Filling Start



Based on parameters loaded with the FIGS command, this command initiates the drawing of the graphics character or area filling pattern stored in parameter RAM. Drawing begins at the address in display memory pointed to by the EAD and dAD values.

Data Read Commands

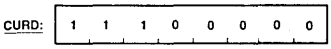
Read Data from Display Memory



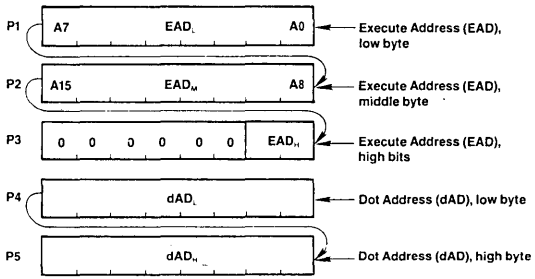
Using the DIR and DC parameters of the FIGS command to establish direction and transfer count, multiple RMW cycles can be executed without specification of the cursor address after the initial load (DC = number of words or bytes).

As this instruction begins to execute, the FIFO buffer direction is reversed so that the data read from display memory can pass to the microprocessor. Any commands or parameters in the FIFO at this time will be lost. A command byte sent to the HGDC will immediately reverse the buffer direction back to write mode, and all RDAT information not yet read from the FIFO will be lost. MOD should be set to 00 if no modification to video buffer is desired.

Cursor Address Read



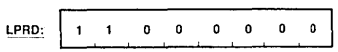
The following bytes are returned by the HGDC through the FIFO:



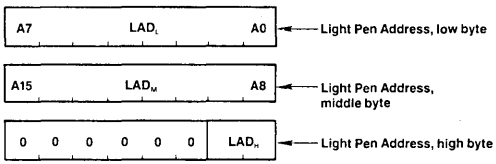
The execute address, EAD, points to the display memory word containing the pixel to be addressed.

The dot address, dAD, within the word is represented as a 1-of-16 code for graphics drawing operations.

Light Pen Address Read



The following bytes are returned by the HGDC through the FIFO:

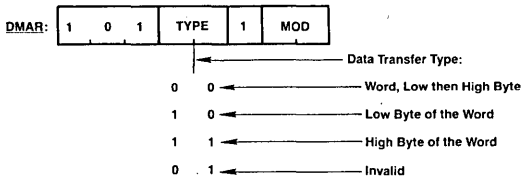


The light pen address, LAD, corresponds to the display word address, DAD, at which the light pen input signal is detected and deglitched.

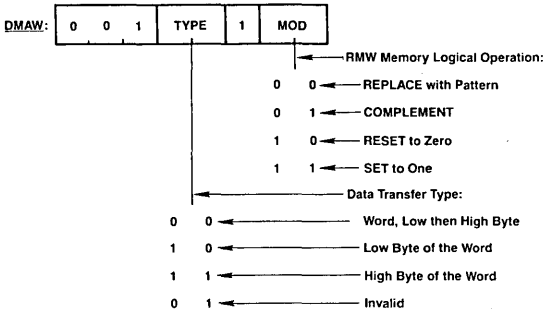
The light pen may be used in graphics, character, or mixed modes but only indicates the word address of light pen position.

DMA Control Commands

DMA Read Request



DMA Write Request



AC Characteristics

T_A = 0 to +70 °C; V_{CC} = 5.0 V ±10%; GND = 0 V

		6 MHz Limits		8 MHz Limits			
Parameter	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Test Conditions**
Read Cycle (GDC ⟷ CPU)							
Address setup to $\overline{RD}\downarrow$	t_{AR}	0		0		ns	
Address hold from $\overline{RD}\uparrow$	t_{RA}	0		0		ns	
$\overline{RD}$ pulse width	t_{RH1}	$t_{RD1}+20$	$t_{RCY}-1/2\,t_{CLK}$	$t_{RD1}+20$	$t_{RCY}-1/2\,t_{CLK}$	ns	
Data delay from $\overline{RD}\downarrow$	t_{RD1}		75		55	ns	C_L = 50 pF
Data floating from $\overline{RD}\uparrow$	t_{DF}	0	75	0	55	ns	
$\overline{RD}$ pulse cycle	t_{RCY}	4 t_{CLK}		4 t_{CLK}		ns	
Write Cycle (GDC ⟷ CPU)							
Address setup to $\overline{WR}\downarrow$	t_{AW}	0		0		ns	
Address hold from $\overline{WR}\uparrow$	t_{WA}	10		10		ns	
$\overline{WR}$ pulse width	t_{WW}	80	$t_{WCY}-t_{CLK}$	60	$t_{WCY}-t_{CLK}$	ns	
Data setup to $\overline{WR}\uparrow$	t_{DW}	65		45		ns	
Data hold from $\overline{WR}\uparrow$	t_{WD}	0		10		ns	
$\overline{WR}$ pulse cycle	t_{WCY}	4 t_{CLK}		4 t_{CLK}		ns	
DMA Read Cycle (GDC ⟷ CPU)							
$\overline{DACK}$ setup to $\overline{RD}\downarrow$	t_{KR}	0		0		ns	
$\overline{DACK}$ hold from $\overline{RD}\uparrow$	t_{RK}	0		0		ns	
$\overline{RD}$ pulse width	t_{RR2}	$t_{RD2}+20$		$t_{RD2}+20$		ns	
Data delay from $\overline{RD}\downarrow$	t_{RD2}		$1.5\,t_{CLK}+80$		$1.5\,t_{CLK}+60$	ns	C_L = 50 pF
DREQ delay from 2xWCLK↑	t_{REQ}		100		75	ns	C_L = 50 pF
DREQ setup to $\overline{DACK}\downarrow$	t_{QK}	0		0		ns	
$\overline{DACK}$ high-level width	t_{DK}	t_{CLK}		t_{CLK}		ns	
$\overline{DACK}$ pulse cycle	t_E	4 t_{CLK} (1)		4 t_{CLK} (1)		ns	
DREQ↓ delay from $\overline{DACK}\downarrow$	$t_{KQ(R)}$		$t_{CLK}+100$		$t_{CLK}+80$	ns	C_L = 50 pF
$\overline{DACK}$ low-level width	t_{LK}	2 t_{CLK}		2 t_{CLK}			

AC Characteristics (cont)

T_A = 0 to +70°C; V_{CC} = 5.0 V ±10%; GND = 0 V

		6 MHz Limits		8 MHz Limits			
Parameter	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Test Conditions**
DMA Write Cycle (GDC ⟷ CPU)							
$\overline{DACK}$ setup to $\overline{WR}$↓	t_{KW}	0		0		ns	
$\overline{DACK}$ hold from $\overline{WR}$↑	t_{WK}	0		0		ns	
RMW Cycle (GDC ⟷ Display Memory)							
Address/data display from 2xWCLK↑	t_{AD}	20	105	15	80	ns	C_L = 50 pF
Address/data floating from 2xWCLK↑	t_{OFF}	20	105	15	80	ns	C_L = 50 pF
Input data setup to 2xWCLK↓	t_{DIS}	0		0		ns	
Input data hold from 2xWCLK↓	t_{DIH}	t_{DE}		t_{DE}		ns	
$\overline{DBIN}$ delay from 2xWCLK↓	t_{DE}	20	80	15	60	ns	C_L = 50 pF
ALE↑ delay from 2xWCLK↑	t_{RR}	20	80	15	60	ns	C_L = 50 pF
ALE↓ delay from 2xWCLK↓	t_{RF}	20	65	15	50	ns	C_L = 50 pF
ALE high width	t_{RW}	1/3 t_{CLK}		1/3 t_{CLK}		ns	C_L = 50 pF
ALE low width	t_{RL}	1.5 t_{CLK} − 30		1.5 t_{CLK} − 30		ns	
Address setup to ALE↓	t_{AA}	30		30			
Display Cycle (GDC ⟷ Display Memory)							
Video signal display from 2xWCLK↑	t_{VD}		90		70	ns	C_L = 50 pF
Input Cycle (GDC ⟷ Display Memory)							
Input signal setup to 2xWCLK↑	t_{PS}	10		10		ns	
Input signal width	t_{PW}	t_{CLK}		t_{CLK}		ns	
Clock (2xWCLK)							
Clock rise time	t_{CR}		15		15	ns	
Clock fall time	t_{CF}		15		15	ns	
Clock high pulse width	t_{CH}	70		52		ns	
Clock low pulse width	t_{CL}	70		52		ns	
Clock cycle	t_{CLK}	165	10000	125	10000	ns	

Note:

(1) For high-byte and low-byte transfers: $t_E = 5\ t_{CLK}$.

Capacitance

T_A = 25°C; V_{CC} = GND = 0V

Parameter	Symbol	Limits Min	Typ	Max	Unit	Test Conditions
Input Capacitance	C_{IN}			10	pF	
I/O Capacitance	$C_{I/O}$			20	pF	fc = 1 MHz
Output Capacitance	C_{OUT}			20	pF	V_I (unmeasured)
Clock Input Capacitance	$C\phi$			20	pF	= 0V

DC Characteristics

T_A = 0°C to +70°C; V_{CC} = 5V ± 10%; GND = 0V

Parameter	Symbol	Limits Min	Typ	Max	Unit	Test Conditions
Input Low Voltage	V_{IL}	−0.5		0.8	V	①
Input High Voltage	V_{IH}	2.2		V_{CC} + 0.5	V	② ③
Output Low Voltage	V_{OL}			0.45	V	I_{OL} = 2.2 mA
Output High Voltage	V_{OH}	2.4			V	I_{OH} = −400 µA
Input Low Leak Current (except VSYNC, $\overline{DACK}$)	I_{IL}			−10	µA	V_I = 0V
Input Low Leak Current (VSYNC, $\overline{DACK}$)	I_{IL}			−500	µA	
Input High Leak Current (except LPEN/DH)	I_{IH}			+10	µA	V_I = V_{CC}
Input High Leak Current (LPEN/DH)	I_{IH}			+500	µA	
Output Low Leak Current	I_{OL}			−10	µA	V_O = 0V
Output High Leak Current	I_{OH}			+10	µA	V_O = V_{CC}
Clock Input Low Voltage	V_{CL}	−0.5		0.6	V	
Clock Input High Voltage	V_{CH}	3.5		V_{CC} + 1.0	V	
V_{CC} Supply Current	I_{CC}			270	mA	

Notes: ① For 2xWCLK, V_{IL} = −0.5V to +0.6V.
② For 2xWCLK, V_{IH} = +3.9V to V_{CC} + 1.0V.
③ For $\overline{WR}$, V_{IH} = 2.5V to V_{CC} + 0.5V.

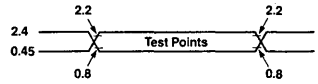
Absolute Maximum Ratings* (Tentative)

Ambient Temperature under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−0.5V to +7V
Power Dissipation	1.5 w

* Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

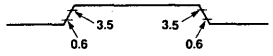
AC Testing Conditions

Input Waveform for AC Test (Except 2xCCLK)



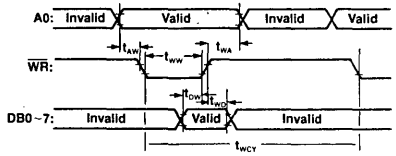
Output Waveform for AC Test

Clock Timing (2xCCLK)

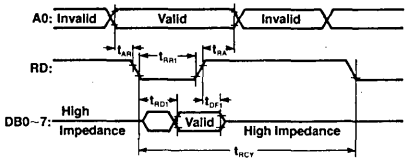


Timing Waveforms

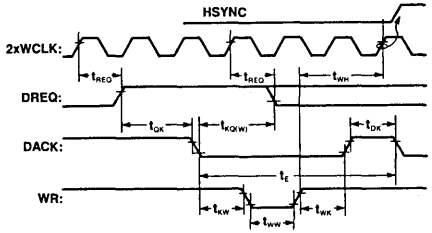
Microprocessor Interface Write Timing



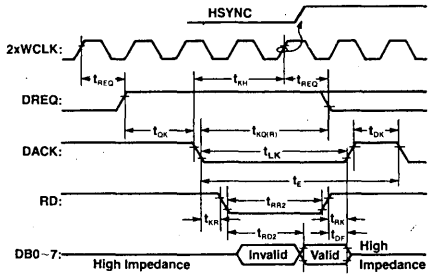
Microprocessor Interface Read Timing



Microprocessor Interface DMA Write Timing

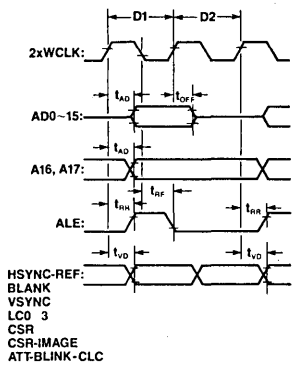


Microprocessor Interface DMA Read Timing

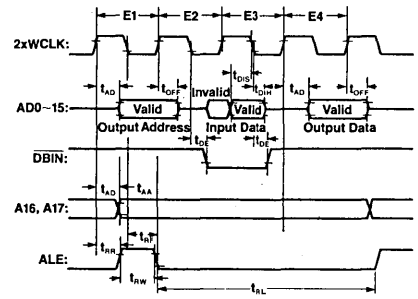


Timing Waveforms (Cont.)

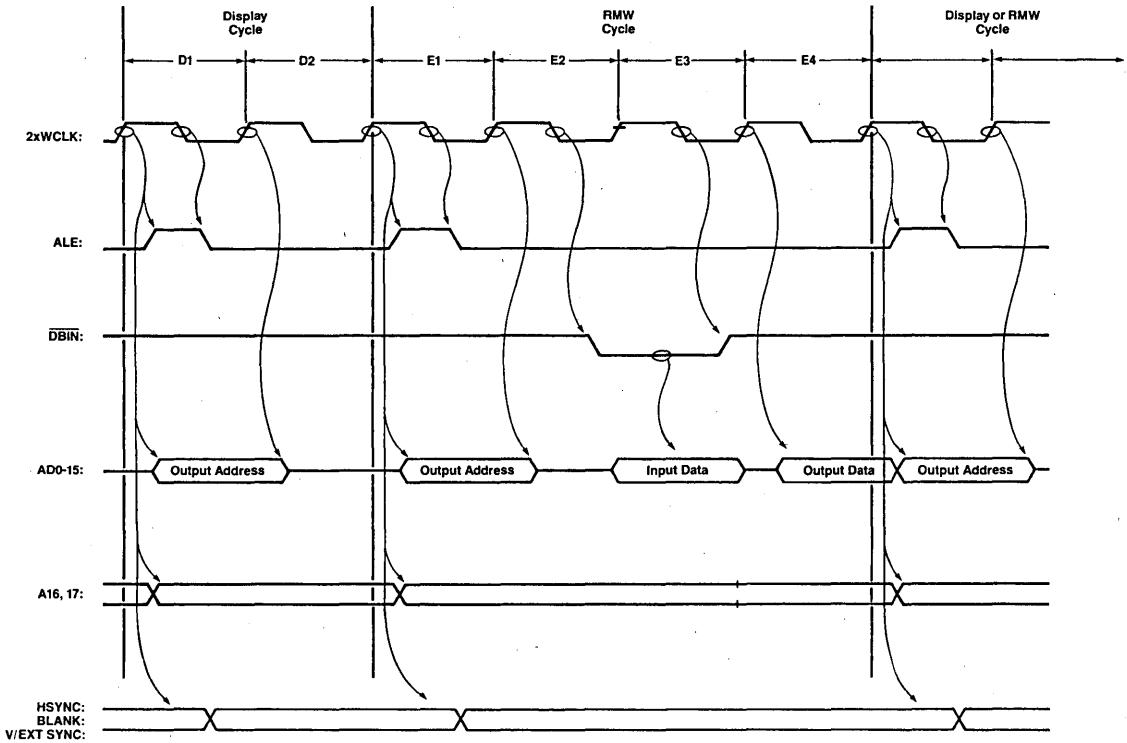
Display Memory Display Cycle Timing



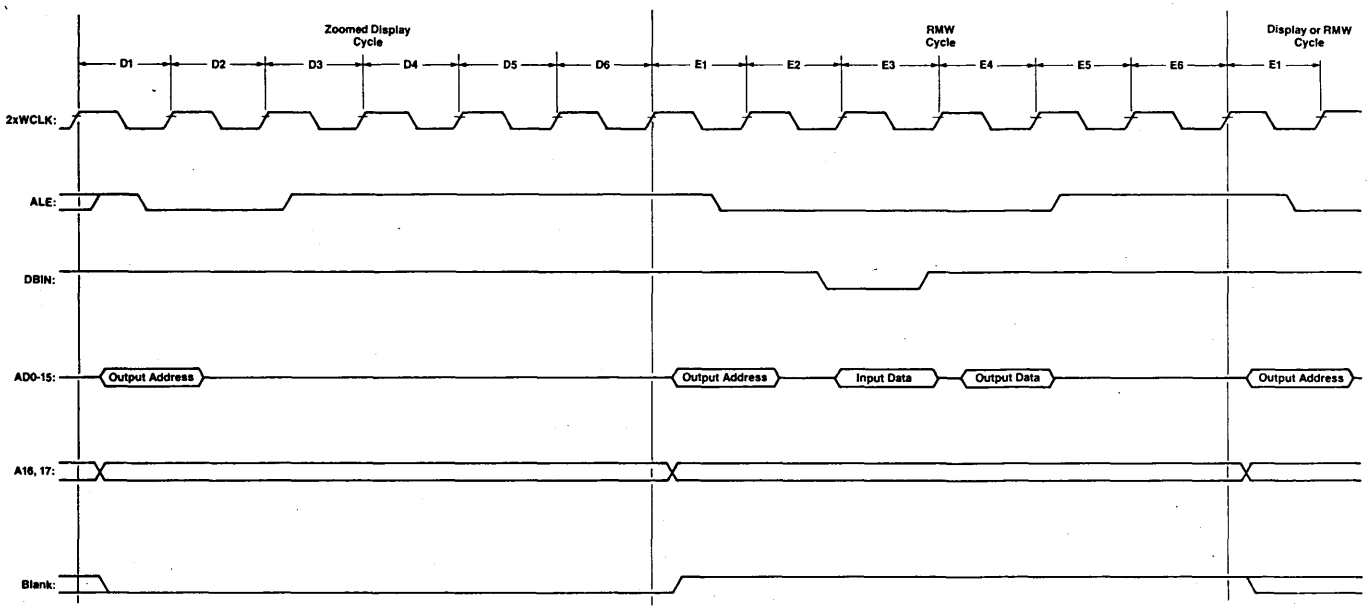
Display Memory RMW Timing



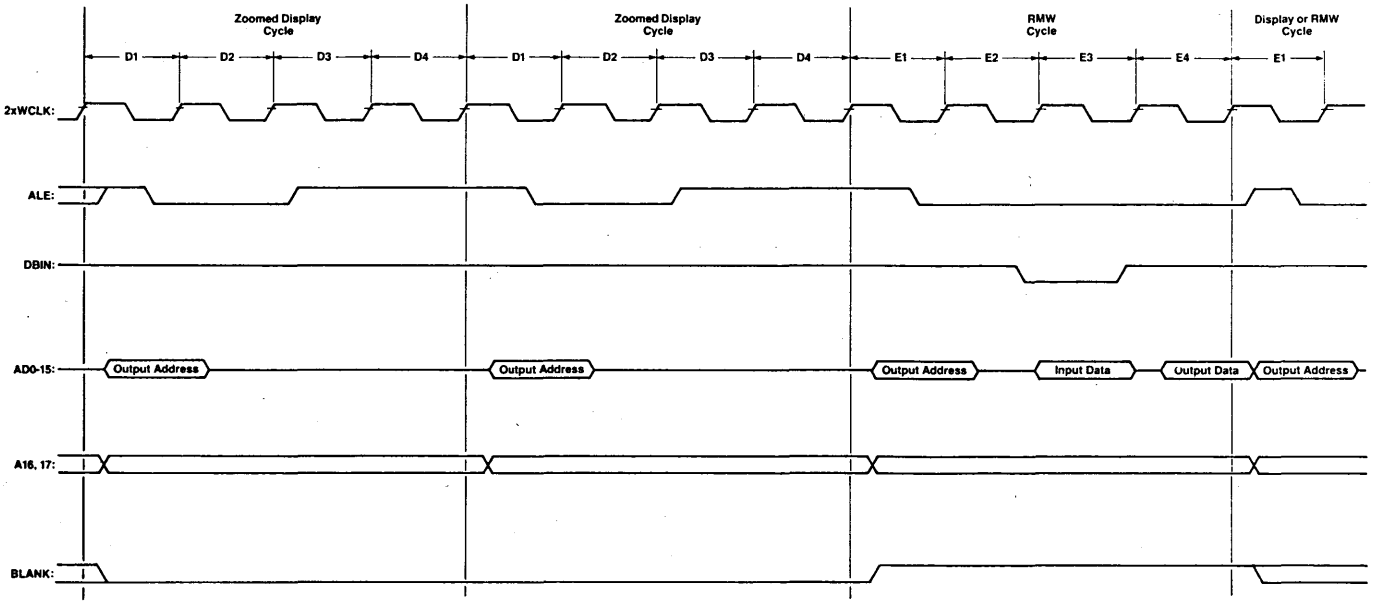
Display and RMW Cycles (1x Zoom)



Display and RMW Cycles (2x Zoom)

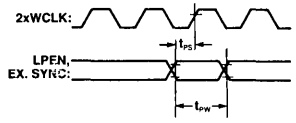


Zoomed Display Operation with RMW Cycle (3x Zoom)

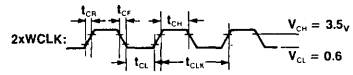


Timing Waveforms (Cont.)

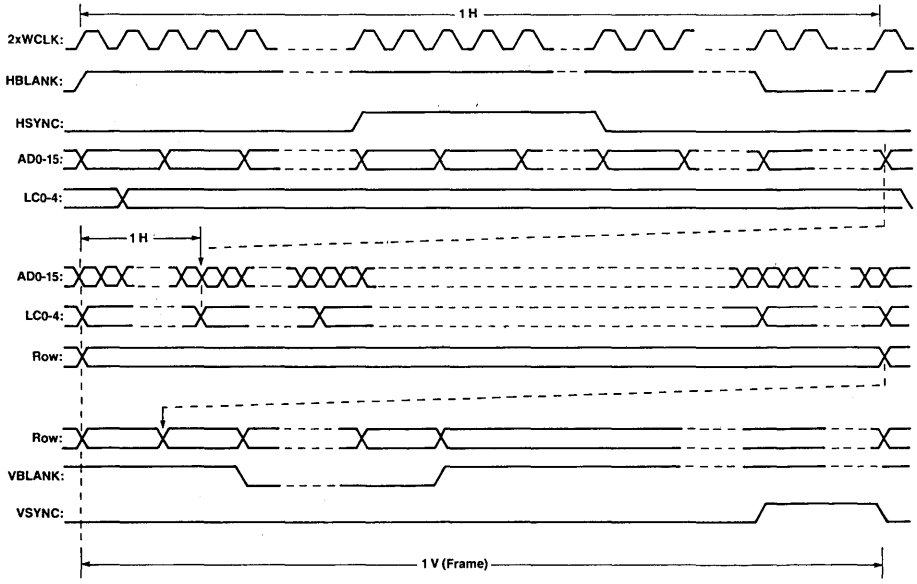
Light Pen and External Sync Input Timing



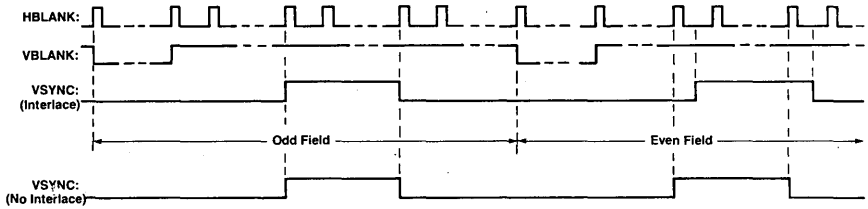
Clock Timing (2xWCLK)



Video Sync Signals Timing

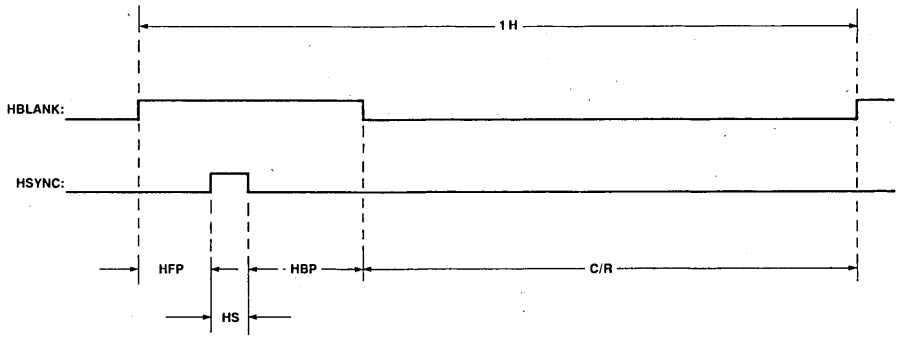


Interlaced Video Timing

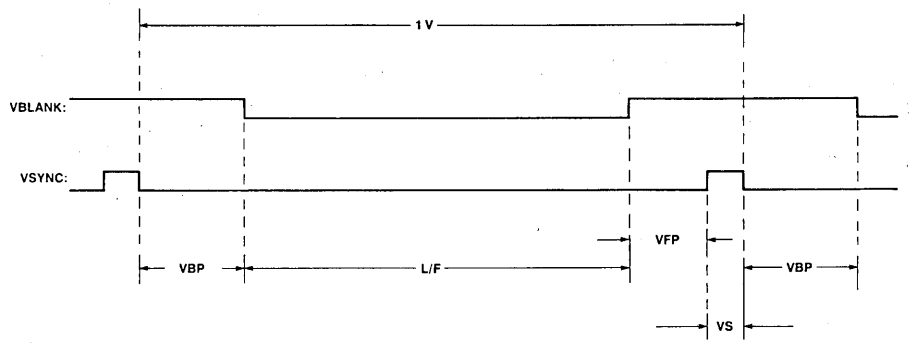


Timing Waveforms (Cont.)

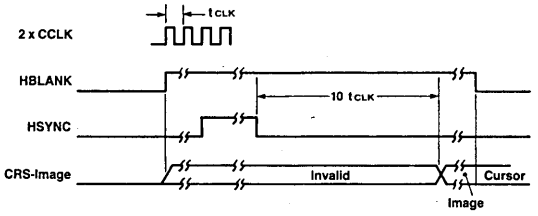
Video Horizontal Sync Generator Parameters



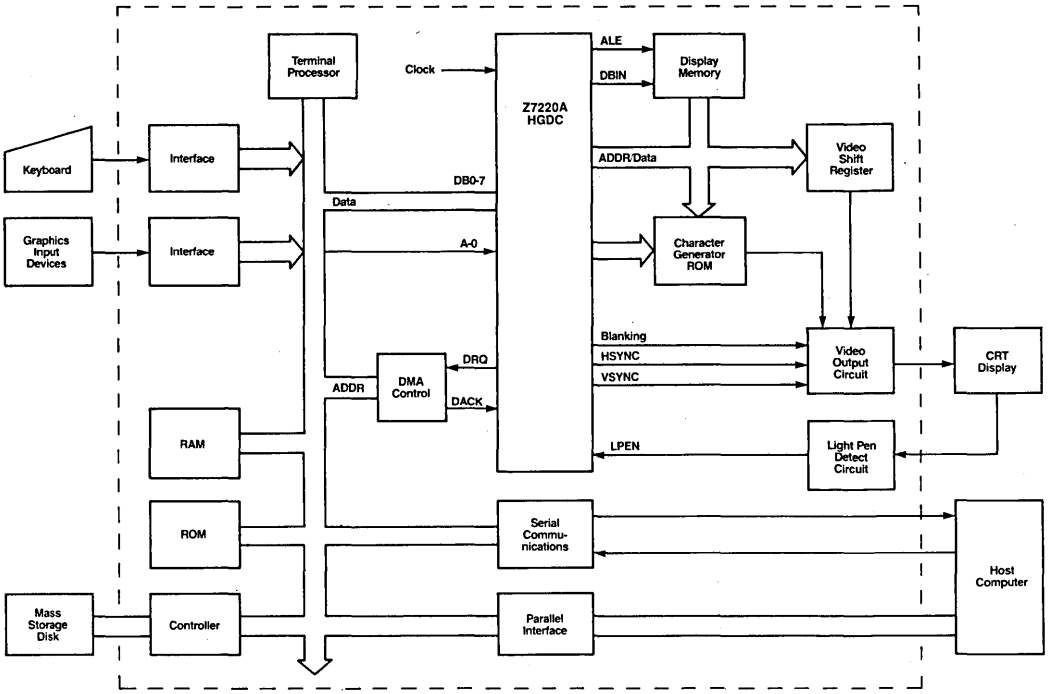
Video Vertical Sync Generator Parameters



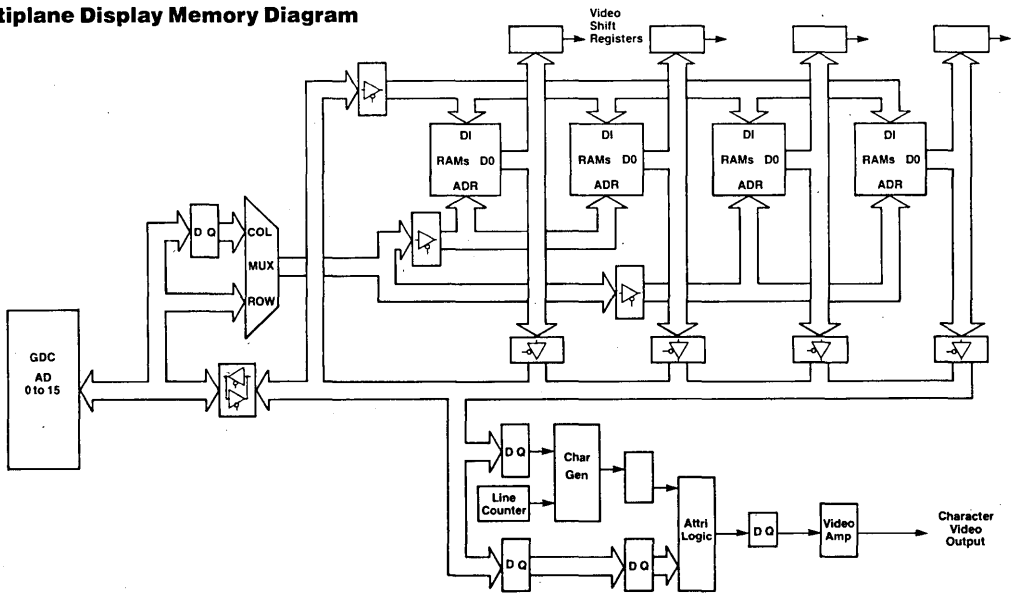
Cursor — Image Bit Flag



Block Diagram of a Graphics Terminal



Multiplane Display Memory Diagram



Z765A FDC
Floppy Disk Controller

October 1988

FEATURES

Address Mark detection circuitry internal to the FDC simplifies the phase locked loop and read electronics. The track stepping rate, head load time, and head unload time are user-programmable.

Z765A features are:

- IBM-compatible format, Single and Double Density
- Multisector and multitrack transfer capability
- Data scan capability—scans a single sector or an entire cylinder comparing byte-for-byte host memory and disk data
- Drives up to 4 floppy-disk drives (FDD)
- Data transfers in DMA or non-DMA mode
- Parallel seek operations on up to four drives
- Compatible with most general-purpose microprocessors
- Single phase 8 MHz clock
- +5V Only
- **40-Pin Dual-In-Line (DIP) package, 44-Pin plastic chip carrier (PLCC) package.**

GENERAL DESCRIPTION

The Z765A is an LSI Floppy Disk Controller (FDC) chip which contains the circuitry and control functions for interfacing a processor to four floppy-disk drives. It supports IBM System 3740 Single Density format (FM) and IBM System 34 Double Density format (MFM) including double-sided recording. The Z765A provides control signals which simplify the design of an external phase locked loop and write precompensation circuitry. The FDC simplifies and handles most of the burdens associated with implementing a floppy-disk interface. (Figure 1).

Handshaking signals make DMA operation easily incorporated with the aid of an external DMA Controller chip, such as the Z80 DMA. The FDC operates in either the DMA or non-DMA mode. In the non-DMA mode the FDC generates interrupts to the processor every time a data byte is to be transferred. In the DMA mode, the processor need only load the command into the FDC and all data transfers occur under control of the FDC and DMA controllers.

The Z765A executes 15 commands; each command requires multiple 8-bit bytes to fully specify the operation which the processor wishes the FDC to perform. The commands are:

- READ DATA
- WRITE DATA
- WRITE DELETED DATA
- READ DELETED DATA
- READ TRACK
- READ ID
- FORMAT TRACK
- SCAN EQUAL
- SCAN HIGH OR EQUAL
- SCAN LOW OR EQUAL
- SEEK
- RECALIBRATE
- SENSE INTERRUPT STATUS
- SPECIFY
- SENSE DRIVE STATUS

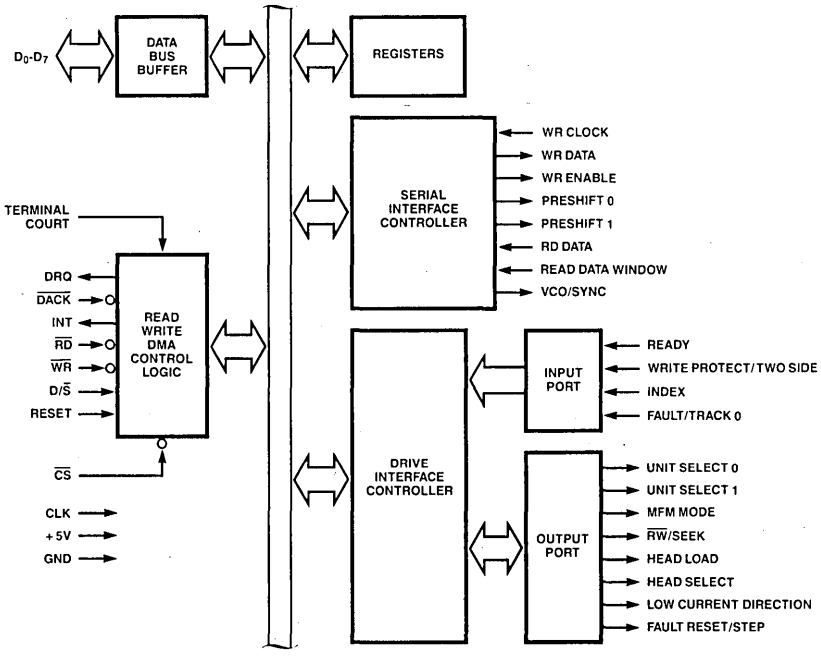


Figure 1. Z765A FDC Block Diagram

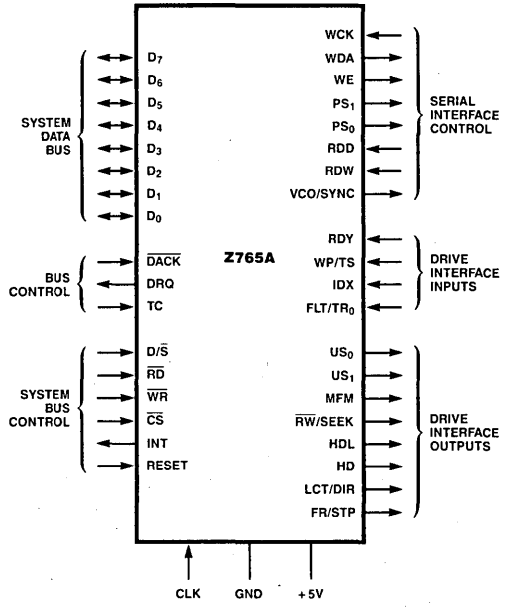


Figure 2. Pin Functions

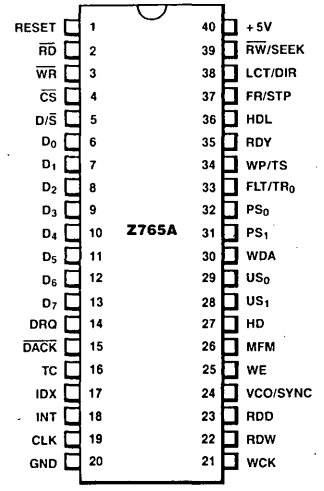


Figure 3a. Pin Assignments

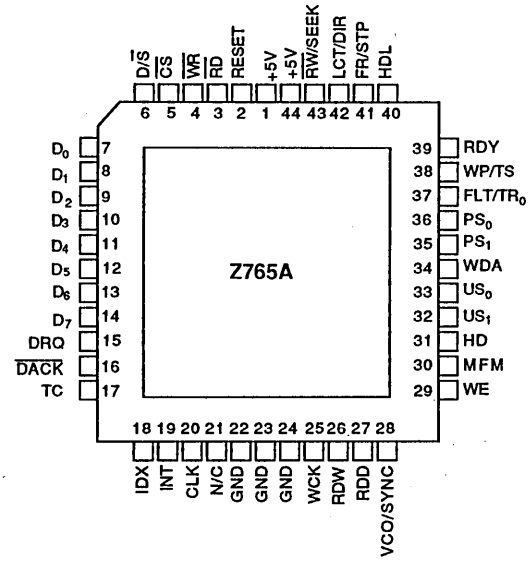


Figure 3b. Pin Assignments

PIN DESCRIPTIONS (Figures 2 and 3)

CLK. *Clock* (input). Single phase 8MHz square wave clock.

$\overline{CS}$. *Chip Select* (input). IC selected when 0 (Low), allowing $\overline{RD}$ and $\overline{WR}$ to be enabled.

D_0-D_7. *Data Bus.* Bidirectional 8-bit Data Bus. Disabled when $\overline{CS}$ = 1.

$\overline{DACK}$. *DMA Acknowledge* (input). DMA cycle is active when 0, and controller is performing DMA transfer.

DRQ. *Data DMA Request* (output). DMA Request is being made by FDC when DRQ = 1.

D/$\overline{S}$. *Data/Status Register Select* (input). Selects Data Register (D/$\overline{S}$ = 1) or Status Register (D/$\overline{S}$ = 0) contents of the FDC to be sent to Data Bus. Disabled when $\overline{CS}$ = 1.

FR/STP. *Fault Reset/Step* (output). Resets fault FF in FDD in Read/Write mode, contains step pulses to move head to another cylinder in Seek mode.

FLT/TR_0. *Fault/Track 0* (input). Senses FDD fault condition in Read/Write mode and Track 0 condition in Seek mode.

HD. *Head Select* (output). Head 1 selected when 1 (High); Head 0 selected when 0 (Low).

HDL. *Head Load* (output). Command which causes read/write head in FDD to contact diskette.

IDX. *Index* (input). Indicates the beginning of a disk track.

INT. *Interrupt* (output). Interrupt Request generated by FDC.

LCT/DIR. *Low Current/Direction* (output). Lowers Write current on inner tracks in Read/Write mode; determines direction head will step in Seek mode. A fault reset pulse is issued at the beginning of each Read or Write command prior to the occurrence of the Head Load signal.

MFM. *MFM Mode* (output). MFM mode when 1; FM mode when 0.

PS_1, PS_0. *Precompensation (preshift)* (output). Write precompensation status during MFM mode. Determines early, late, and normal times.

$\overline{RD}$. *Read* (input). When 0, control signal for transfer of data from FDC to Data Bus. Disabled when $\overline{CS}$ = 1.

RDD. *Read Data* (input). Read data from FDD, containing clock and data bits.

RDW. *Read Data Window* (input). Generated by PLL, and used to sample data from FDD.

RDY. *Ready* (input). Indicates FDD is ready to send or receive data.

RESET. *Reset* (input). Places FDC in idle state. Resets output lines to FDD to 0. Does not affect SRT, HUT or HLT in Specify command. If RDY pin is held High during Reset, FDC generates an interrupt within 1.024 msec. To clear this interrupt use Sense Interrupt Status command.

$\overline{RW}$/SEEK. *Read Write/Seek* (output). When 1 (High) Seek mode selected; when 0 (Low) Read/Write mode selected.

TC. *Terminal Count* (input). Indicates the termination of a DMA transfer when 1 (High). It terminates data transfer during Read/Write/Scan command in DMA or Interrupt mode.

US_1, US_0. *Unit Select* (output). FDD Unit selected.

VCO/SYNC. (output). Inhibits VCO in PLL when 0 (Low); enables VCO when 1.

WCK. *Write Clock* (input). Write data rate to FDD. FM = 500 KHz, MFM = 1 MHz with a pulse width of 250 ns for both FM and MFM.

WDA. *Write Data* (output). Serial clock and data bits to FDD.

WE. *Write Enable* (output). Enables write data into FDD.

WP/TS. *Write Protect/Two Side* (input). Senses Write Protect status in Read/Write mode and Two-Side Media in Seek mode.

$\overline{WR}$. *Write* (input). When 0, control signal for transfer of data to FDC via Data Bus. Disabled when $\overline{CS}$ = 1.

Table 1. Internal Registers

The bits in the Main Status Register are defined as follows:

Bit No.	Name	Symbol	Description
D_0	FDD 0 Busy	D_0B	FDD number 0 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_1	FDD 1 Busy	D_1B	FDD number 1 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_2	FDD 2 Busy	D_2B	FDD number 2 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_3	FDD 3 Busy	D_3B	FDD number 3 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_4	FDC Busy	CB	A read or write command is in process. FDC will not accept any other command.
D_5	Execution Mode	EXM	This bit is set only during execution phase in non-DMA mode. When D_5 goes low, execution phase has ended and result phase has started. It operates only during non-DMA mode of operation.
D_6	Data Input/Output	DIO	Indicates direction of data transfer between FDC and Data Register. If DIO = 1, then transfer is from Data Register to the processor. If DIO = 0, transfer is from the processor to Data Register.
D_7	Request for Master	RQM	Indicates Data Register is ready to send or receive data to or from the processor. Both bits DIO and RQM should be used to perform the handshaking functions of "ready" and "direction" to the processor.

INTERNAL REGISTERS

The Z765A contains two registers which may be accessed by the main system processor: a Status register and a Data register. The 8-bit Main Status register (Table 1) contains the FDC status information and may be accessed at any time. The 8-bit Data register is several registers in a stack; one register at a time is presented to the data bus. The Data register stores data, commands, parameters, and FDD status information. Data bytes are read out of, or written into, the Data register in order to program or obtain the results after a particular command. Only the Status register may be read and used to facilitate the transfer of data between the processor and Z765A.

The relationship between the Status/Data registers and the signals $\overline{RD}$, $\overline{WR}$, and D/$\overline{S}$ is shown in Table 2.

The Data Input/Output (DIO) and Request for Master (RQM) bits in the Status register indicate when data is ready and the direction transfer on the data bus (Figure 4). The maximum time between the last $\overline{RD}$ or $\overline{WR}$ during a command or result phase and the set or reset DIO and RQM is 12μs; every time the Main Status register is read the CPU should wait 12μs. The maximum time from the trailing edge of the last $\overline{RD}$ in the result phase to when D_4 (FDC busy) goes Low is 12μs.

Table 2. Relationships Between Status/Data Registers and $\overline{RD}$, $\overline{WR}$, and D/$\overline{S}$

D/$\overline{S}$	$\overline{RD}$	$\overline{WR}$	Function
0	0	1	Read Main Status Register
0	1	0	Illegal
0	0	0	Illegal
1	0	0	Illegal
1	0	1	Read from Data Register
1	1	0	Write into Data Register

STATUS REGISTER IDENTIFICATION

Bit No.	Name	Symbol	Description
			Status Register 0
D_7 D_6	Interrupt Code	IC	D_7 = 0 and D_6 = 0 Normal Termination of command, (NT). Command was completed and properly executed.
			D_7 = 0 and D_6 = 1 Abnormal Termination of command, (AT). Execution of command was started but was not successfully completed.
			D_7 = 1 and D_6 = 0 Invalid Command issue, (IC). Command which was issued was never started.
			D_7 = 1 and D_6 = 1 Abnormal Termination because during command execution the ready signal from FDD changed state.
D_5	Seek End	SE	When the FDC completes the SEEK command, this flag is set to 1 (High).
D_4	Equipment Check	EC	If a fault signal is received from the FDD, or if the Track 0 signal fails to occur after 77 step pulses (Recalibrate Command) then this flag is set.
D_3	Not Ready	NR	When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to Side 1 of a single-sided drive, then this flag is set.
D_2	Head Address	HD	This flag is used to indicate the state of the head at Interrupt.
D_1	Unit Select 1	US_1	This flag is used to indicate a Drive Unit Number at Interrupt.
D_0	Unit Select 0	US_0	This flag is used to indicate a Drive Unit Number at Interrupt.
			Status Register 1
D_7	End of Cylinder	EN	When the FDC tries to access a sector beyond the final sector of a cylinder, this flag is set.
D_6			Not used. This bit is always 0 (Low).
D_5	Data Error	DE	When the FDC detects a Cyclic Redundancy Check (CRC) error in either the ID field or the data field, this flag is set.
D_4	Overrun	OR	If the FDC is not serviced by the host system during data transfers within a certain time interval, this flag is set.
D_3			Not used. This bit always 0 (Low).
D_2	No Data	ND	During execution of READ DATA, WRITE DELETED DATA or SCAN command, if the FDC cannot find the sector specified in the Internal Data Register (IDR), this flag is set.
			During execution of the READ ID command, if the FDC cannot read the ID field without an error, then this flag is set.
			During execution of the READ A cylinder command, if the starting sector cannot be found, then this flag is set.

STATUS REGISTER IDENTIFICATION (Continued)

Bit No.	Name	Symbol	Description
			Status Register 1 (Continued)
D_1	Not Writeable	NW	During execution of WRITE DATA, WRITE DELETED DATA or Format A cylinder command, if the FDC detects a write protect signal from the FDD, then this flag is set.
D_0	Missing Address Mark	MA	If the FDC cannot detect the ID Address Mark after encountering the index hole twice, then this flag is set. If the FDC cannot detect the Data Address Mark or Deleted Data Address Mark, this flag is set. Also at the same time, the MD (Missing Address Mark in data field) of Status register 2 is set.
			Status Register 2
D_7			Not used. This bit is always 0 (Low).
D_6	Control Mark	CM	During execution of the READ DATA or SCAN command, if the FDC encounters a sector which contains a Deleted Data Address Mark, this flag is set.
D_5	Data Error in Data Field	DD	If the FDC detects a CRC error in the data field then this flag is set.
D_4	Wrong Cylinder	WC	This bit is related to the ND bit, and when the contents of Cylinder (C) on the medium is different from that stored in IDR, this flag is set.
D_3	Scan Equal Hit	SH	During execution of the SCAN command, if the condition of "equal" is satisfied, this flag is set.
D_2	Scan Not Satisfied	SN	During execution of the SCAN command, if the FDC cannot find a sector on the cylinder which meets the condition, then this flag is set.
D_1	Bad Cylinder	BC	This bit is related to the ND bit, and when the contents of C on the medium is different from that stored in the IDR and the contents of C is FF_H, then this flag is set.
D_0	Missing Address Mark in Data Field	MD	When data is read from the medium, if the FDC cannot find a Data Address Mark or Deleted Data Address Mark, then this flag is set.
			Status Register 3
D_7	Fault	FT	This bit is used to indicate the status of the Fault signal from the FDD.
D_6	Write Protected	WP	This bit is used to indicate the status of the Write Protected signal from the FDD.
D_5	Ready	RY	This bit is used to indicate the status of the Ready signal from the FDD.
D_4	Track 0	T0	This bit is used to indicate the status of the Track 0 signal from the FDD.
D_3	Two Side	TS	This bit is used to indicate the status of the Two Side signal from the FDD.
D_2	Head Address	HD	This bit is used to indicate the status of the Side Select signal to the FDD.
D_1	Unit Select 1	US_1	This bit is used to indicate the status of the Unt Select 1 signal to the FDD.
D_0	Unit Select 0	US_0	This bit is used to indicate the status of the Unit Select 0 signal to the FDD.

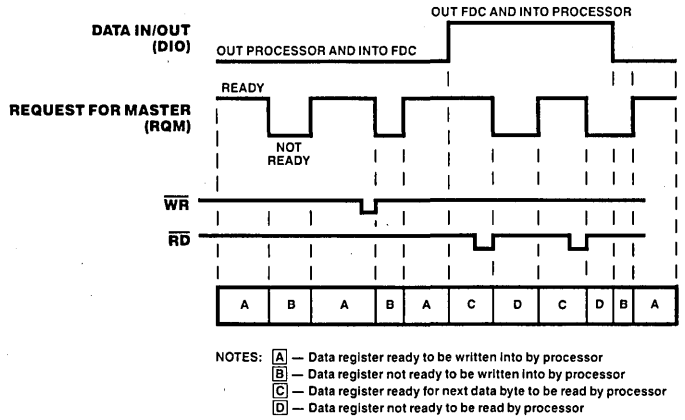


Figure 4. Data Transfer

COMMAND SEQUENCE

The Z765A is capable of performing 15 different commands. Each command is initiated by a multibyte transfer from the processor; the result after execution of the command may also be a multibyte transfer back to the processor. Because of this multibyte interchange of information between the Z765A and the processor, each command consists of three phases:

Command Phase. The FDC receives all information required to perform a particular operation form the processor.

Execution Phase. The FDC performs the operation it was instructed to do.

Result Phase. After completion of the operation, status and other housekeeping information are made available to the processor.

The Instruction set shows the required preset parameters and results for each command. Most commands require 9 command bytes and return 7 bytes during the result phase. The W to the left of each byte indicates a command phase byte to be written; an R indicates a result byte.

PROCESSOR INTERFACE

During Command or Result phases the Main Status register must be read by the processor before each byte of information is written into, or read from, the Data register. Then the CPU should wait for 12μs before reading the Main Status register. Bits D_6 and D_7 in the Main Status register must be in a 0 and 1 state, respectively, before each byte of the command word may be written into the Z765A. Many of the commands require multiple bytes and, as a result, the Main Status register must be read prior to each byte transfer to the Z765A. During the Result phase, D_6 and D_7 in the Main Status register must both be 1's before reading each byte from the Data Register. Reading the Main Status register before each byte transfer to the Z765A is required only in the Command and Result phases, not during the Execution phase.

If the Z765A is in the non-DMA mode and reading data from FDD, then the receipt of each data byte is indicated by an interrupt signal on pin 18(INT = 1). The generation of a Read signal ($\overline{RD}$ = 0) or Write signal ($\overline{WR}$ = 0) will clear the interrupt and output the data onto the data bus. If the processor cannot handle interrupts fast enough (every 13μs for the MFM mode and 27μs for the FM mode), then it may poll the Main Status register and bit D_7 (RQM) functions as the interrupt signal. If a Write command is in process, the $\overline{WR}$ signal negates the reset to the interrupt signal.

In the non-DMA mode it is necessary to examine the Main Status register to determine the cause of the interrupt, since it could be a data interrupt or a command termination interrupt, either normal or abnormal. If the Z765A is in the

COMMAND SYMBOL DESCRIPTION

Symbol	Name	Description
D/$\overline{S}$	Data/Status Select	D/$\overline{S}$ controls selection of Main Status register (D/$\overline{S}$ = 0) or Data register (D/$\overline{S}$ = 1)
C	Cylinder Number	C stands for the current/selected cylinder (track) numbers 0 through 76 of the medium.
D	Data	D stands for the data pattern which is going to be written into a sector.
D_7-D_0	Data Bus	8-bit Data Bus, where D_7 stands for a most significant bit, and D_0 stands for a least significant bit.
DTL	Data Length	When N is defined as 00, DTL stands for the data length which users are going to read out or write into the sector.
EOT	End of Track	EOT stands for the final sector number on a cylinder. During Read or Write operations, FDC will stop data transfer after a sector number equal to EOT.
GPL	Gap Length	GPL stands for the length of Gap 3. During Read/Write commands this value determines the number of bytes that VCO/SYNC will stay low after two CRC bytes. During Format command it determines the size of Gap 3.
H	Head Address	H stands for head number 0 or 1, as specified in ID field.
HD	Head	HD stands for a selected head number 0 or 1 and controls the polarity of pin 27. (H = HD in all command words.)
HLT	Head Load Time	HLT stands for the head load time in the FDD (2 to 254 ms in 2 ms increments).
HUT	Head Unload Time	HUT stands for the head unload time after a Read or Write operation has occurred (16 to 240 ms in 16 ms increments).
MF	FM or MFM Mode	If MF is Low, FM mode is selected, and if it is High. MFM mode is selected.
MT	Multitrack	If MT is high, a Multitrack operation is performed. If MT = 1 after finishing Read/Write operation on side 0, FDC automatically starts searching for sector 1 on side 1.
N	Number	N stands for the Number of data bytes written in a sector.
NCN	New Cylinder Number	NCN stands for a New Cylinder Number or desired position of head which is going to be reached as a result of the Seek operation.
ND	Non-DMA Mode	ND stands for operation in the Non-DMA mode.
PCN	Present Cylinder Number	PCN stands for the cylinder number or present position of Head at the completion of Sense Interrupt Status command.
R	Record	R stands for the sector number which will be read or written.
R/W	Read/Write	R/W stands for either Read (R) or Write (W) signal.
SC	Sector	SC indicates the number of Sectors per Cylinder.
SK	Skip	SK stands for Skip Deleted Data Address mark.
SRT	Step Rate Time	SRT stands for the Stepping Rate for the FDD (1 to 16 ms in 1 ms increments). Stepping Rate applies to all drives ($F_{(16)}$ = 1 ms, $E_{(16)}$ = 2 ms, $D_{(16)}$ = 3 ms, . . .).
ST0 ST1 ST2 ST3	Status 0 Status 1 Status 2 Status 3	ST0-3 stands for one of four registers which store the status information after a command has been executed. This information is available during the result phase after command execution. These registers should not be confused with the main status register (selected by D/$\overline{S}$ = 0). ST0-3 may be read only after a command has been executed and contains information relevant to that particular command.
STP	Step	During a Scan operation, if STP = 1, the data in contiguous sectors is compared byte by byte with data sent from the processor (or DMA); if STP = 2, then alternate sectors are read and compared.
US_0, US_1	Unit Select	Used to select between drives 0-3.

INSTRUCTION SET[1, 2]

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
					Read Data					
Command	W	MT	MF	SK	0	0	1	1	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	See Note 3
	W	C								Sector ID information prior to command execution. The 4 bytes are commanded against header on Floppy disk.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
					Read Deleted Data					
Command	W	MT	MF	SK	0	1	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution. The 4 bytes are commanded against header on Floppy Disk.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Write Data										
Command	W	MT	MF	0	0	0	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector IC information prior to command execution. The 4 bytes are commanded against header on Floppy Disk.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the main system and FDD
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution.
	R	H								
	R	R								
	R	N								
Write Deleted Data										
Command	W	MT	MF	0	0	1	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution. The 4 bytes are commanded against header on Floppy disk.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Read A Track										
Command	W	0	MF	SK	0	0	0	1	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system. FDC reads all data fields from index hole to EOT.
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
Read ID										
Command	W	0	MF	0	0	1	0	1	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
Execution										The first correct ID information on the cylinder is stored in Data Register.
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information read during Execution phase from Floppy Disk.
	R	H								
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
					Format A Track					
Command	W	0	MF	0	0	1	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	N								Bytes Sector
	W	SC								Sectors/Track
	W	GPL								Gap 3
	W	D								Filler byte
Execution										FDC formats an entire track.
Result	R	ST0								Status information after command
	R	ST1								execution
	R	ST2								
	R	C								In this case, the ID information
	R	H								has no meaning.
	R	R								
	R	N								
					Scan Equal					
Command	W	MT	MF	SK	1	0	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to
	W	H								command execution
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data compared between the FDD and the main system.
Result	R	ST0								Status information after command
	R	ST1								execution
	R	ST2								
	R	C								Sector ID information after
	R	H								command execution
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
					Scan Low or Equal					
Command	W	MT	MF	SK	1	1	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and main system
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
					Scan High or Equal					
Command	W	MT	MF	SK	1	1	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and main system.
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution.
	R	H								
	R	R								
	R	N								
					Recalibrate					
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	X	X	X	X	X	0	US_1	US_0	
Execution										Head retracted to Track 0

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Sense Interrupt Status										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	ST0								Status information about the FDC
	R	PCN								at the end of seek operation
Specify										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	SRT							HUT	
	W	HLT							ND	
Sense Drive Status										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
Result	R	ST3								Status information about FDD
Seek										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	NCN								
Execution										Head is positioned over proper cylinder on diskette.
Invalid										
Command	W	Invalid Codes								Invalid Command Codes (NoOp—FDC goes into Standby state.)
Result	R	ST0								ST0 = $80_{(H)}$

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

DMA mode, no interrupts are generated during the Execution phase. The Z765A generates DRQs (DMA Requests) when each byte of data is available. The DMA Controller responds to this request with both a $\overline{DACK}$ (DMA Acknowledge) = 0 and an $\overline{RD}$ (Read signal) = 0. When the DMA Acknowledge signal goes Low ($\overline{DACK}$ = 0), then the DMA request is cleared (DRQ = 0). If a Write command has been issued, a $\overline{WR}$ signal appears instead of $\overline{RD}$. After the Execution phase has been completed [Terminal Count (TC) has occurred] or the last sector on the cylinder (EOT) read/written, then an interrupt occurs (INT = 1) which signifies the beginning of the Result phase. When the first byte of data is read during the Result phase, the interrupt is automatically cleared (INT = 0).

The $\overline{RD}$ or $\overline{WR}$ signals should be asserted while $\overline{DACK}$ is true. The $\overline{CS}$ signal is used in conjunction with $\overline{RD}$ and $\overline{WR}$ as a gating function during programmed I/O operations. $\overline{CS}$ has no effect during DMA operations. If the non-DMA mode is chosen, the $\overline{DACK}$ signal should be pulled up to V_{CC}.

During the Result phase all bytes shown in the Command Table must be read. For example, the Read Data command has seven bytes of data in the Result phase; all seven bytes must be read to successfully complete the Read Data command and allow the Z765A to accept a new command.

The Z765A contains five Status registers. The Main Status register can be read at any time by the processor. The other four Status registers (ST0, ST1, ST2, and ST3) are available only during the Result phase and can be read only after completing a command. The particular command that has been executed determines how many of the Status registers are read.

The bytes of data which are sent to the Z765A to form the Command phase and are read out of the Z765A in the Result phase must occur in the order shown in the Command Table. That is, the Command Code must be sent first and the other bytes sent in the prescribed sequence. No foreshortening of the Command or Result phases is allowed. After the last byte of data in the Command phase is sent to the Z765A, the Execution phase automatically starts. In a similar fashion, when the last byte of data is read out in the Result phase, the command is automatically ended and the Z765A is ready for a new command.

POLLING FEATURE OF THE Z765A

After Reset is sent to the Z765A, the Unit Select lines US_0 and US_1 automatically go into a polling mode (Figure 5). Between commands (and between step pulses in the Seek command) the Z765A polls all four FDDs looking for a change in the Ready line from any of the drives. If the Ready line changes state (usually due to a door opening or closing), then the Z765A generates an interrupt. When Status register 0 (ST0) is read (after Sense Interrupt Status is issued), Not Ready (NR) is indicated. The polling of the Ready line by the Z765A occurs continuously between commands, thus notifying the processor which drives are on or off line. Each drive is polled every 1.024 ms except during the Read/Write commands. When used with a 4 MHz clock for interfacing to minifloppies, the polling rate is 2.048 ms.

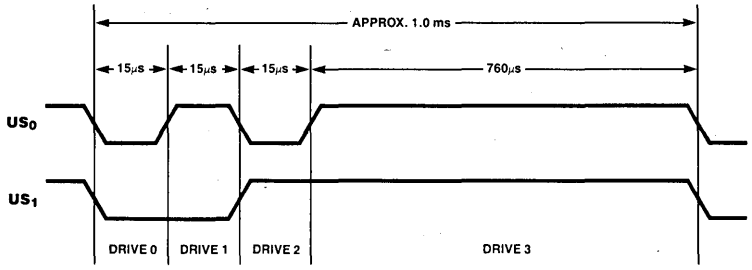


Figure 5. Polling Features

COMMANDS

Read Data

A set of nine (9) byte words are required to place the FDC into the Read Data Mode. After the Read Data command is issued, the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify command), and begins reading ID Address Marks and ID fields. When the current sector number (R) stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC, via the data bus, outputs data byte-to-byte from the data field to the main system.

After completion of the read operation from the current sector, the Sector Number is incremented by one, and the data from the next sector is read and output on the data bus. This continuous read function is called a Multi-Sector Read Operation. The Read Data command can be terminated by the receipt of a TC signal which should be issued when the DACK for the last byte of data is sent. Upon receipt of this signal, the FDC stops outputting data to the processor, but continues to read data from the current sector, checks Cyclic Redundancy Count (CRC), and at the end of the sector, terminates the Read Data command. The amount of data which can be handled with a single command to the FDC depends upon multitrack (MT), MFM/FM (MF), and Number of Bytes/Sector (N). Table 3 shows the Transfer Capacity.

Table 3. Transfer Capacity

Multi-Track MT	MFM/FM MF	Bytes/Sector N	Maximum Transfer Capacity (Bytes/Sector) (Number of Sectors)	Final Sector Read from Diskettes
0	0	00	(128) (26) = 3,328	26 at Side 0
0	1	01	(256) (26) = 6,656	or 26 at Side 1
1	0	00	(128) (52) = 6,656	26 at Side 1
1	1	01	(256) (52) = 13,312	
0	0	01	(256) (15) = 3,840	15 at Side 0
0	1	02	(512) (15) = 7,680	or 15 at Side 1
1	0	01	(256) (30) = 7,680	15 at Side 1
1	1	02	(512) (30) = 15,360	
0	0	02	(512) (8) = 4,096	8 at Side 0
0	1	03	(1024) (8) = 8,192	or 8 at Side 1
1	0	02	(512) (16) = 8,192	8 at Side 1
1	1	03	(1024) (16) = 16,384	

MT allows the FDC to read data from both sides of the diskette. For a particular cylinder, data is transferred starting at Sector 1, Side 0 and completing at the last sector, Sector L, Side 1. This function pertains to only one cylinder (the same track) on each side of the diskette.

When N = 0, then DTL defines the data length which the FDC must treat as a sector. If DTL is smaller than the actual data length in a Sector, the data beyond DTL in the Sector is not sent to the Data Bus. The FDC internally reads the complete sector performing the CRC check and, depending upon the manner of command termination, may perform a Multi-Sector Read Operation. When N is non-zero, then DTL has no meaning and should be set to FF_H.

At the completion of the Read Data Command the head is unloaded, after the Head Unload Time Interval specified in the Specify Command has elapsed. If the processor issues another command before the head unloads, there is no head settling time between subsequent reads. This time saved is particularly valuable when a diskette is copied.

If the FDC twice detects the index hole without finding the right sector (R), then the FDC sets Status register 1's No Data (ND) flag to 1, and terminates the Read Data command. (Status register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

After reading the ID and Data fields in each sector, the FDC checks the CRC bytes. If a read error is detected indicating incorrect CRC in the ID field, the FDC sets Status register 1's Data Error (DE) flag to 1, and if a CRC error occurs in the Data Field, the FDC also sets Status register 2's Data Error in Data Field (DD) flag to 1, and terminates the Read Data command. (Status register 0, bit 7 = 0, bit 6 = 1.)

If the FDC reads a Deleted Data Address Mark off the diskette, and the SK bit D in the first Command Word = 0, then the FDC sets Status register 2's Control Mark (CM) flag to 1, and after reading all the data in the sector, terminates the Read Data command. If SK = 1, the FDC skips the sector with the Deleted Data Address Mark and reads the next sector. When SK = 1, the CRC bits in the deleted data field are not checked.

During disk data transfers between the FDC and the processor, via the data bus, the FDC must be serviced by the processor every 27μs in the FM Mode, and every 13μs in the MFM Mode, or the FDC sets Status register 1's Overrun (OR) flag to 1, and terminates the Read Data command.

If the processor terminates a read or write operation in the FDC, then the ID information in the Result Phase is dependent upon the state of the MT bit and EOT byte. Table 4 shows the values for C, H, R, and N when the processor terminates the command.

Table 4. C, H, R, and N Values When Processor Terminates Commands

MT	HD	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	0	Less than EOT	NC	NC	R+1	NC
	0	Equal to EOT	C + 1	NC	R = 01	NC
	1	Less than EOT	NC	NC	R + 1	NC
	1	Equal to EOT	C + 1	NC	R = 01	NC
1	0	Less than EOT	NC	NC	R + 1	NC
	0	Equal to EOT	NC	LSB	R = 01	NC
	1	Less than EOT	NC	NC	R + 1	NC
	1	Equal to EOT	C + 1	LSB	R = 01	NC

NOTES: NC (No Change): The same value as the one at the beginning of command execution.
LSB (Least Significant Bit): The least significant bit of H is complemented.

Write Data

A set of nine (9) bytes is required to set the FDC in the Write Data mode. After the Write Data command is issued, the FDC loads the head, waits the specified head setting time, and begins reading ID fields. When all four bytes (C, H, R, and N) loaded during the command match the four bytes of the ID field from the diskette, the FDC takes data from the processor byte-by-byte via the data bus and outputs it to the FDD.

After writing data into the current sector, the sector number stored in the R register is incremented by one, and new data is written into the next data field. The FDC continues this Multisector Write Operation until a Terminal Count signal is issued. If a Terminal Count signal is sent to the FDC, it continues writing into the current sector to complete the data field. If the Terminal Count signal is received while a data field is being written, the remainder of the data field is filled with zeros.

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (CRC error) in one of the ID fields, it sets Status register 1's DE flag to 1, and terminates the Write Data command. (Status register 0, bit 7 = 0, bit 6 = 1.)

The Write command operates in the same manner as the Read command for the following items:

- Transfer capacity
- End of cylinder (EN) flag
- No data (ND) flag
- Head unload time interval
- ID information when the processor terminates command
- Definition of DTL when N = 0 and when N ≠ 0

Refer to the Read Data command for details.

In the Write Data mode, data transfers between the processor and FDC via the data bus, must occur every 27μs in the FM mode and every 13μs in the MFM mode. If the time interval between data transfers is longer, then the FDC sets Status register 1's Overrun (OR) flag to 1, and terminates the Write Data command. (Status register 0, bit 7 = 0, bit 6 = 1.)

Write Deleted Data

This command is the same as the Write Data command except a Deleted Data Address mark, instead of the normal Data Address mark, is written at the beginning of the data field.

Read Deleted Data

This command is the same as the Read Data command except that when the FDC detects a Data Address mark at the beginning of a data field and SK = 0, the FDC reads all the data in the sector and sets Status register 2's CM flag to 1, and terminates the command. If SK = 1, then the FDC skips the sector with the Data Address mark and reads the next sector.

Read Track

This command is similar to the Read Data command except that this is a continuous Read operation where the entire data field from each of the sectors is read. Immediately after

sensing the index hole, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID or Data CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the value stored in the IDR and, if there is no comparison, sets Status register 1's ND flag to 1. Multitrack or skip operations are not allowed with this command.

This command terminates when the number of sectors read is equal to EOT. If the FDC does not find an ID Address mark on the diskette after it senses the index hole for the second time, it sets Status register 1's Missing Address mark (MA) flag to 1 and terminates the command. (Status Register 0, bit 7 = 0, bit 6 = 1.)

Read ID

The Read ID command gives the present position of the recording head. The FDC stores the values from the first ID field it can read. If no proper ID Address mark is found on the diskette before the index hole is encountered for the second time, Status register 1's MA flag is set to 1; if no data is found, Status register 1's No Data (ND) flag is set to 1. The command is then terminated with STO bit 7 = 0 and bit 6 = 1. During this command, data transfer between FDC and the CPU occurs only during the result phase.

Format Track

The Format command allows an entire track to be formatted. After the index hole is detected, data is written on the diskette; Gaps, Address marks, ID fields and data fields, all per the IBM 3740 Single Density format or IBM System 34 Double Density format, are recorded. The processor, during the command phase, supplies values i.e., Number of bytes/sector (N), Sectors Cylinder (SC), Gap Length (GPL), and Data Pattern (D) which determine the particular format to be written.

The data field is filled with the byte of data stored in D. The ID field for each sector is supplied by the processor; that is, four data requests per sector are made by the FDC for Cylinder number (C), Head number (H), Sector number (R), and Number of bytes/sector (N). This allows diskette formatting with nonsequential sector numbers.

The processor must send new values for C, H, R, and N to the Z765A for each sector on the track. If FDC is set for the DMA mode, it issues four DMA requests per sector. If it is set for the Interrupt mode, it issues four interrupts per sector and the processor must supply C, H, R, and N loads for each sector. The contents of the R register are incremented by 1 after each sector is formatted; thus, the R register contains a value of R when it is read during the Result phase. This incrementing and formatting continues for the whole track until the FDC detects the index hole for the second time, whereupon it terminates the command.

If the Fault signal is received from the FDD at the end of a Write operation, the FDC sets Status register 0's EC flag to 1 and terminates the command after setting Status register 0, bit 7 to 0 and bit 6 to 1. Also the loss of a Ready signal at the beginning of a command execution phase causes Status register 0, bit 7 and 6 to be set to 0 and 1 respectively.

Table 5 shows the sector size relationship between N, SC, and GPL.

Table 5. Functional Description of Commands

Format	Sector Size	N	SC	GPL[1]	GPL[2,3]
		8" Standard Floppy			
FM Mode	128 bytes sector	00	1A	07	1B
	256	01	0F	0E	2A
	512	02	08	1B	3A
	1024	03	04	47	8A
	2048	04	02	C8	FF
	4096	05	01	C8	FF
MFM Mode[4]	256	01	1A	0E	36
	512	02	0F	1B	54
	1024	03	08	35	74
	2048	04	04	99	FF
	4096	05	02	C8	FF
	8192	06	01	C8	FF
		5¼" Minifloppy			
FM Mode	128 bytes/sector	00	12	07	09
	128	00	10	10	19
	256	01	08	18	30
	512	02	04	46	87
	1024	03	02	C8	FF
	2048	04	01	C8	FF
MFM Mode[4]	256	01	12	0A	0C
	256	01	10	20	32
	512	02	08	2A	50
	1024	03	04	80	F0
	2048	04	02	C8	FF
	4096	05	01	C8	FF

NOTES: 1. Suggested values of GPL in Read or Write commands to avoid splice point between data field and ID field of contiguous sections.
2. Suggested values of GPL in format command.
3. All values except sector size are hexidecimal.
4. In MFM mode FDC cannot perform a Read/Write format operation with 128 bytes sector. (N = 00)

Scan Commands

The Scan commands allow comparison of data read from the diskette and data supplied from the main system. The FDC compares the data on a byte-by-byte basis and looks for a sector of data which meets the conditions of $D_{FDD} = D_{Processor}$, $D_{FDD} \leq D_{Processor}$, or $D_{FDD} \geq D_{Processor}$. The hexadecimal byte of FF from memory or from FDD can be used as a mask byte because it always meets the condition of the comparison. One's complement arithmetic is used for comparison (FF = largest number, 00 = smallest number). After a whole sector of data is compared, if the conditions are not met, the sector number is incremented (R + STP → R) and the scan operation continues until one of the following conditions occur: the conditions for scan are met (equal, low, or high), the last sector on the track is reached (EOT), or the terminal count (TC) signal is received.

If the conditions for scan are met, the FDC sets the Status register 2's Scan Hit (SH) flag to 1 and terminates the Scan command. If the conditions for scan are not met between the starting sector number (R) and the last sector on the cylinder (EOT), then the FDC sets Status register 2's Scan Not Satisfied (SN) flag to 1, and terminates the Scan command. During the scan operation, the receipt of a signal from the processor or DMA controller causes the FDC to complete the comparison of the particular byte in process and then to terminate the command. Table 6 shows the status of bits SH and SN under various conditions of Scan.

Table 6.

	Status Register 2		
Command	**Bit 2 = SN**	**Bit 3 = SH**	**Comments**
Scan Equal	0	1	$D_{FDD} = D_{Processor}$
	1	0	$D_{FDD} \neq D_{Processor}$
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} < D_{Processor}$
	1	0	$D_{FDD} > D_{Processor}$
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} > D_{Processor}$
	1	0	$D_{FDD} < D_{Processor}$

If the FDC encounters a Deleted Data Address mark on one of the sectors and SK = 0, then it regards the sector as the last sector on the cylinder, sets Status register 2's Control Mark (CM) flag to 1 and terminates the command. If SK = 1, the FDC skips the sector with the Deleted Address mark, reads the next sector, and sets Status register 2's Control Mark (CM) flag to 1 to show that a Deleted sector has been encountered.

When either the Step (STP) (contiguous sectors = 01 or alternate sectors = 02) sectors are read or the Multitrack (MT) is programmed, the last sector on the track must be read. For example, if STP = 02, MT = 0, the sectors are numbered sequentially 1 through 26 and the Scan command is started at sector 21, the following happens. Sectors 21, 23, and 25 are read, then the next sector, 26, is skipped and the index hole is encountered before the EOT value of 26 can be read resulting in an abnormal termination of the command. If the EOT had been set at 25 or the scanning started at sector 20, then the Scan command would be completed in a normal manner.

During the Scan command, data is supplied by either the processor or DMA Controller for comparison against the data read from the diskette. In order to avoid having Status register 1's Overrun (OR) flag set, it is necessary to have the data available in less than 27μs (FM mode) or 13μs (MFM mode). If an Overrun occurs, the FDC ends the command with Status register 0, bit 7 cleared to 0 and bit 6 set to 1.

Seek

The Read/Write head within the FDD is moved from cylinder to cylinder under control of the Seek command. The FDC has four independent Present Cylinder registers for each drive which are cleared only after the Recalibrate command. The FDC compares the Present Cylinder Number (PCN) which is the current head position with the New Cylinder Number (NCN), and if there is a difference, performs the following operations:

PCN < NCN: Direction signal to FDD set to 1, and Step Pulses are issued. (Step In)

PCN > NCN: Direction signal to FDD cleared to 0, and Step Pulses are issued. (Step Out)

The rate at which Step pulses are issued is controlled by Stepping Rate Time (SRT) in the Specify command. After each Step pulse is issued NCN is compared against PCN, and when NCN = PCN, Status register 0's Seek End (SE) flag is set to 1, and the command is terminated. At this point FDC interrupt goes High. Bits D_0-D_3 in the Main Status register are set during the Seek operation and are cleared by the Sense Interrupt Status command.

During the command phase of the Seek operation the FDC is in the FDC Busy state, but during the execution phase it is in the Nonbusy state. While the FDC is in the Nonbusy state, another Seek command may be issued, and in this manner parallel Seek operations may be done on up to four drives at once. No other command can be issued for as long as the FDC is in the process of sending step pulses to any drive.

If an FDD is in a Not Ready state at the beginning of the command execution phase or during the Seek operation, then Status register 0's Not Ready (NR) flag is set to 1, and the command is terminated after bit 7 is set to 1 and bit 6 to 0.

If writing three bytes of Seek command exceeds 150μs, the timing between the first two step pulses may be 1ms shorter than that set in the Specify command.

Recalibrate

The function of this command is to retract the Read/Write head within the FDD to the Track 0 position. The FDC clears the contents of the PCN counter and checks the status of the Track 0 signal from the FDD. As long as the Track 0 signal is Low, the Direction signal remains 0 and step pulses are issued. When the Track 0 signal goes High, the Status register 0's SE flag is set to 1 and the command is terminated. If the Track 0 signal is still Low after 77 step pulses have been issued, the FDC sets Status register 0's SE and Equipment Check (EC) flags to 1s and terminates the command after Status register 0, bit 7 is cleared to 0 and bit 6 is set to 1.

The ability to do overlap Recalibrate commands to multiple FDDs and the loss of the Ready signal, as described in the Seek command, also applies to the Recalibrate command. If the Diskette has more than 77 tracks, the Recalibrate command should be issued twice, in order to position the Read/Write head to Track 0.

Sense Interrupt Status

An interrupt signal is generated by the FDC for one of the following reasons:

1. Upon entering the Result phase of command:
   - ☐ Read Data
   - ☐ Write Data
   - ☐ Write Deleted Data
   - ☐ Read Deleted Data
   - ☐ Read Track
   - ☐ Read ID
   - ☐ Format Track
   - ☐ Scan
2. Ready Line of FDD changes state
3. End of Seek or Recalibrate command
4. During Execution phase in the non-DMA mode

Interrupts caused by reasons 1 and 4 occur during normal command operations and are easily discernible by the processor. During an execution phase in non-DMA mode, D_5 in the Main Status Register is High. Upon entering the Result phase this bit is cleared. Reasons 1 and 4 do not require Sense Interrupt Status commands. The interrupt is cleared by Reading/Writing data to the FDC. Interrupts caused by reasons 2 and 3 may be uniquely identified with the aid of the Sense Interrupt Status command which resets the Interrupt signal and, via bits 5, 6, and 7 of Status register 0, identifies the cause of the interrupt (Table 7).

Table 7. Interrupt Identification

Seek End	Interrupt Code		
Bit 5	Bit 6	Bit 7	Cause
0	1	1	Ready Line changed state, either polarity
1	0	0	Normal Termination of Seek or Recalibrate command
1	1	0	Abnormal Termination of Seek or Recalibrate command

The Sense Interrupt Status command is used in conjunction with the Seek and Recalibrate commands which have no result phase. When the disk has reached the desired head position, the Z765A sets the interrupt line true. The host CPU must then issue a Sense Interrupt Status command to determine the actual cause of the interrupt, which could be Seek End or a change in ready status from one of the drives. Figure 6 is a graphic example.

Specify

The Specify command sets the initial values for each of the three internal timers. The Head Unload Time (HUT) defines the time from the end of the execution phase of one of the Read/Write commands to the head unload state. This timer is programmable from 16 to 240ms in increments of 16ms (01 = 16ms, 02 = 32ms...OF_{16} = 240ms). The Step Rate Time (SRT) defines the time interval between adjacent step pulses. This timer is programmable from 1 to 16ms in increments of 1ms (F = 1ms, E = 2ms, and D = 3ms). The Head Load Time (HLT) defines the time between the Head Load signal's going High and the start of the Read/Write operation. This timer is programmable from 2 to 254ms in increments of 2ms (01 = 2ms, 02 = 4ms, 03 = 6ms...7F = 254ms).

The time intervals mentioned are a direct function of the 8MHz clock; if the clock were reduced to 4MHz (minifloppy application), all time intervals would be increased by a factor of 2.

The choice of a DMA or non-DMA operation is made by the Non-DMA (ND) bit. When this bit is High (ND = 1), the Non-DMA mode is selected; when ND = 0, the DMA mode is selected.

Sense Drive Status

The processor uses this command to obtain the status of the FDDs. Status register 3 contains the Drive Status information stored internally in FDC registers.

Invalid

If an Invalid command (not defined above) is sent to the FDC, then the FDC terminates the command after Status Register 0 bit 7 is set to 1 and bit 6 to 0. No interrupt is generated by the Z765A during this condition. Bits 6 and 7 (DIO and RQM) in the Main Status register are both High, indicating to the processor that the Z765A is in the Result phase and the contents of Status register 0 (STO) must be read. When the processor reads Status register 0, it finds an 80_H indicating the receipt of an Invalid command.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate Interrupt, otherwise the FDC considers the next command as an Invalid command.

This command may be used as a No-Op command to place the FDC in a standby or No Operation state.

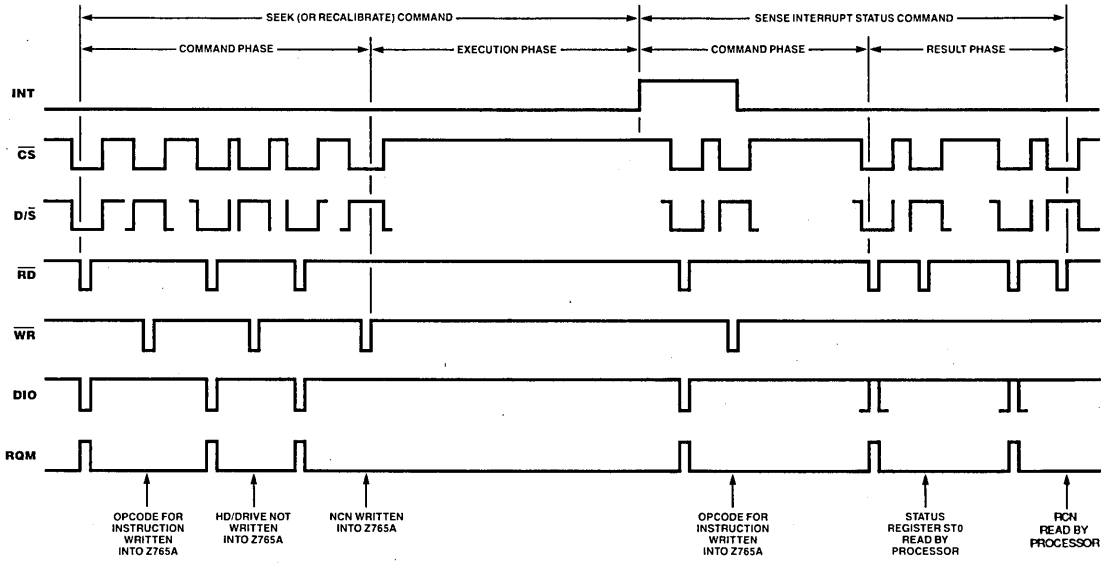


Figure 6. Seek, Recalibrate, and Sense Interrupt Status

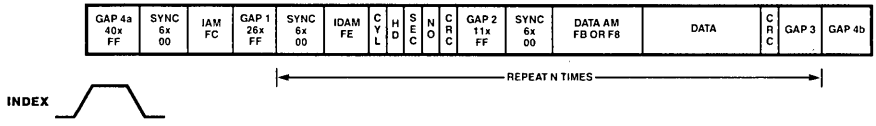


Figure 7. Data Format, FM Mode

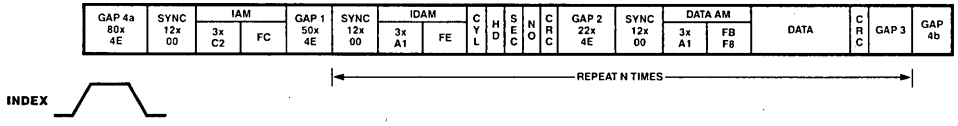


Figure 8. Data Format, MFM Mode

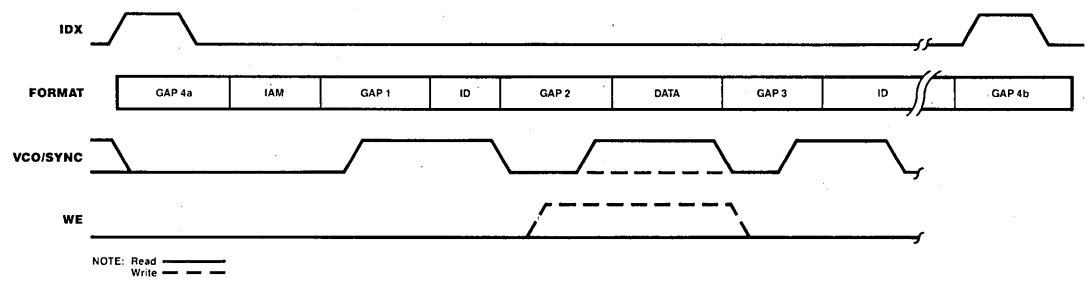


Figure 9. Data Timing Relationships

AC CHARACTERISTICS

$T_A = -10°C$ to $+70°C$; $V_{CC} = +5V \pm 5\%$ unless otherwise specified.

Number	Symbol	Parameter	Min	Typ[1]	Max	Unit	Test Condition
			120	125	500	ns	
1	TcC	Clock Cycle Time		125		ns	8″ FDD
				250		ns	5¼″ FDD
2	TwCh	Clock Width (High)	40			ns	
2a	TwCl	Clock Width (Low)	40			ns	
3	TrC	Clock Rise Time			20	ns	
4	TfC	Clock Fall Time			20	ns	
5	TsAR	D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$ to $\overline{RD}$ ↓ Setup Time	0			ns	
6	ThRA	D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$ from $\overline{RD}$ ↑ Hold Time	0			ns	
7	TwRD	$\overline{RD}$ Width	250			ns	
8	TdRDf (Do)	$\overline{RD}$ ↓ to Data Output Delay			200	ns	C_L = 100 pf
9	TdRDr (Dz)	$\overline{RD}$ ↑ to Data Float Delay	20		100	ns	C_L = 100 pf
10	TsCS(WRf)	Control Signal (D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$) to $\overline{WR}$ ↓ Setup Time	0			ns	
11	ThCS(WRr)	Control Signal (D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$) from $\overline{WR}$ ↑ Hold Time	0			ns	
12	TwWR	$\overline{WR}$ Width	250			ns	
13	TsD(WRr)	Data to $\overline{WR}$ ↑ Setup Time	150			ns	
14	ThD(WRr)	Data from $\overline{WR}$ ↑ Hold Time	5			ns	
15	TdRDr(INT)	$\overline{RD}$ ↑ to INT Delay Time			500	ns	
16	TdWRr(INT)	$\overline{WR}$ ↑ to INT Delay Time			500	ns	
17	TcDRQ	DRQ Cycle Time	13			µs	
18	TdDRQ(DACK)	$\overline{DACK}$ ↓ to DRQ ↓ Delay			200	ns	
19	TdDACK(DRQ)	DRQ ↑ to $\overline{DACK}$ ↓ Delay	200			ns	TcC = 125 ns
20	TwDACK	$\overline{DACK}$ Width	2			TcC	
21	TwTC	TC Width	1			TcC	
22	TwRST	Reset Width	14			TcC	
				4		µs	MFM = 0 5¼″
				2		µs	MFM = 1 5¼″
23	TcWCK	WCK Cycle Time		2		µs	MFM = 0 8″
				1		µs	MFM = 1 8″
24	TwWCKh	WCK Width (High)	80	250	350	ns	
25	TrWCK	WCK Rise Time			20	ns	
26	TfWCK	WCK Fall Time			20	ns	
27	TdWCKr(PS)	WCK ↑ to Preshift Delay Time	20		100	ns	
28	TdWCKr(WEr)	WCK ↑ to WE ↑ Delay Time	20		100	ns	
29	TdWCKr(WDA)	WCK ↑ to WDA Delay Time	20		100	ns	
30	TwRDDh	RDD Width (High)	40			ns	
				4		µs	MFM = 0 5¼″
				2		µs	MFM = 1 5¼″
31	TWCY	Window Cycle Time		2		µs	MFM = 0 8″
				1		µs	MFM = 1 8″
32	TsW(RDDh) ThW(RDDl)	Window to RDD ↑ Setup Time Window from RDD ↓ Hold Time	15			ns	
33	TsUS(RWh)	Unit Select to $\overline{RW}$/SEEK ↑ Setup Time	12			µs	
34	TsRWr(DIR)	$\overline{RW}$/SEEK ↑ to LCT/DIR Setup Time	7			µs	
35	TsDIR(STEPr)	LCT/DIR to STEP ↑ Setup Time	1			µs	
36	ThUS(STEPl)	Unit Select from STEP ↓ Hold Time	5			µs	

NOTES: 1. Typical values for T_A = 25°C and nominal supply voltage.
2. Under software control, the range is from 1 ms to 16 ms at 8 MHz clock period.

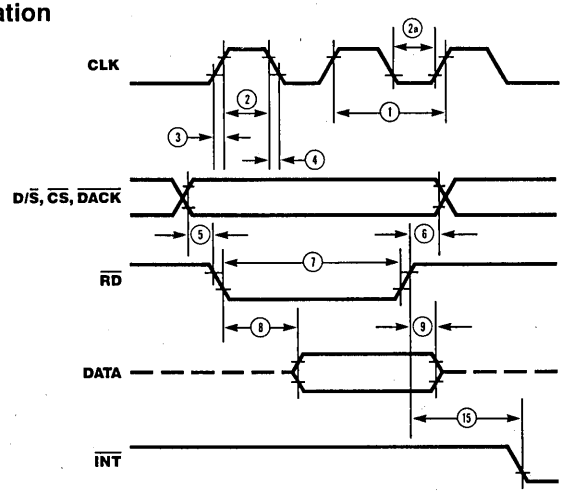
AC CHARACTERISTICS (Continued)

$T_A = -10°C$ to $+70°C$; $V_{CC} = +5V \pm 5\%$ unless otherwise specified.

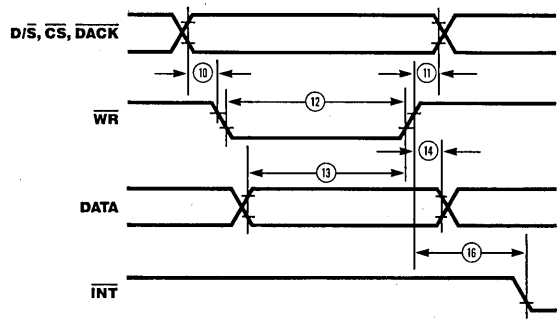
Number	Symbol	Parameter	Min	Typ[1]	Max	Unit	Test Condition
37	TwSTEPh	STEP Width (High)	6	7	8	µs	
38	TcSTEP	STEP Cycle Time	16	Note 2	Note 2	µs	
39	TwFRh	FAULT RESET Width (High)	8		10	µs	
40	TwWDAh	Write Data (WDA) Width (High)	T_0-50			ns	
41	ThUS(SEEKf)	Unit Select from $\overline{RW}$/SEEK ↓ Hold Time	15			µs	
42	ThSEEK(DIR)	$\overline{RW}$/SEEK from LCT/DIR Hold Time	30			µs	
43	ThDIR(STEPf)	LCT/DIR from STEP ↓ Hold Time	24			µs	
44	TwIDX	INDEX Width (High and Low)	4			TcC	
45	TdDRQh(RDl)	DRQ ↑ to $\overline{RD}$ ↓ Delay Time	800			ns	
46	TdDRQh(WRl)	DRQ ↑ to $\overline{WR}$ ↓ Delay Time	250			ns	
47	TdDRQh(RWh)	DRQ ↑ to $\overline{RD}$ ↑ or $\overline{WR}$ ↑ Delay Time			12	µs	

NOTES: 1. Typical values for $T_A = 25°C$ and nominal supply voltage.
2. Under software control, the range is from 1 ms to 16 ms at 8 MHz clock period.

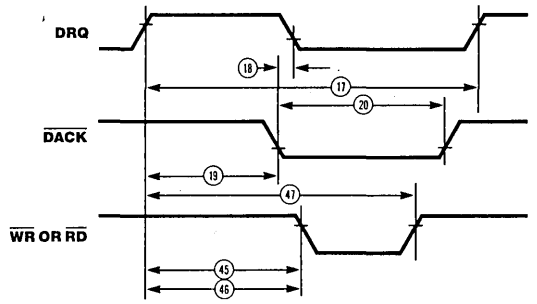
Processor Read Operation



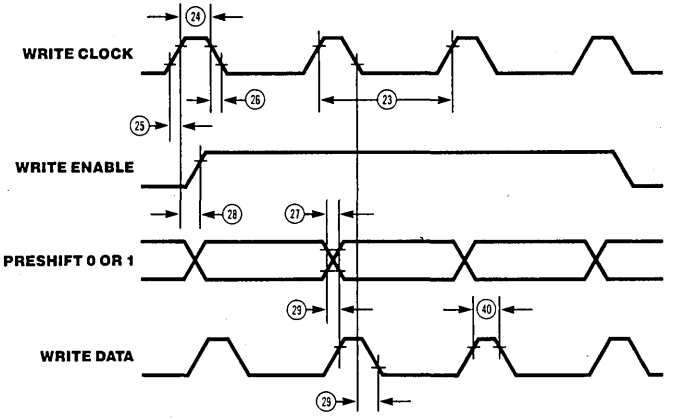
Processor Write Operation



DMA Operation

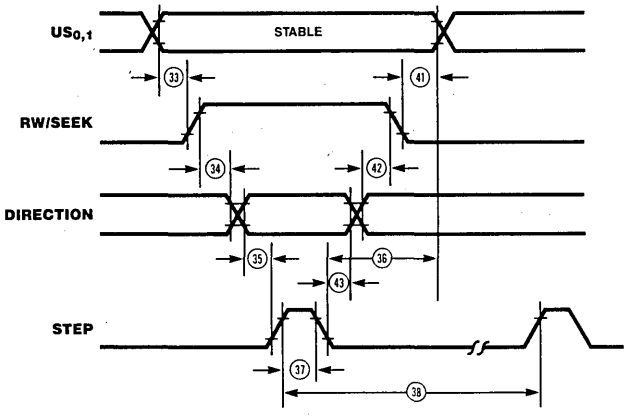


FDD Write Operation



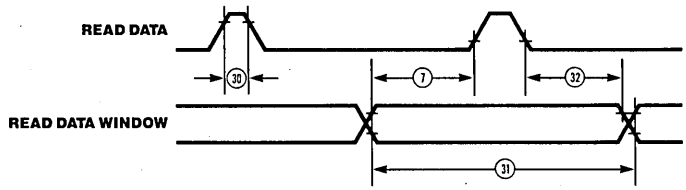
	Preshift 0	Preshift 1
Normal	0	0
Late	0	1
Early	1	0

Seek Operation

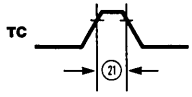


INDEX

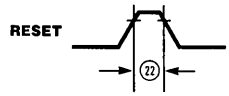
FDD Read Operation



Terminal Count



RESET



ABSOLUTE MAXIMUM RATINGS

T_A = 25°C

Operating Temperature . . . . . .0°C to + 70°C
Storage Temperature . . . . . .−65°C to +150°C
All Output Voltages . . . . . .−.5V to +7V
All Input Voltages . . . . . .−.5V to +7V
Supply Voltage V_{CC} . . . . . .−.5V to +7V
Power Dissipation 1W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above these indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = +5V ± 5% unless otherwise specified.

Symbol	Parameter	Min	Typ*	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage			0.40	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V_{CC}	V	I_{OH} = −200 μA
V_{ILC}	Input Low Voltage (CLK + WR Clock)	−0.5		0.65	V	
V_{IHC}	Input High Voltage (CLK + WR Clock)	2.4		V_{CC} + 0.5	V	
I_{CC}	V_{CC} Supply Current			150	mA	
I_{LI}	Input Load Current			10	μA	V_{IN} = V_{CC}
	(All Input Pins)			−10	μA	V_{IN} = 0V
I_{LOH}	High Level Output Leakage Current			10	μA	V_{OUT} = V_{CC}
I_{LOL}	Low Level Output Leakage Current			−10	μA	V_{OUT} = +0.40V

*Typical values for T_A = 25°C and nominal supply voltage.

CAPACITANCE

T_A = 25°C; f_c = 1 MHz; V_{CC} = 0V

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{CLOCK}	Clock Input Capacitance		20	pF	All pins except pin under
C_{IN}	Input Capacitance		10	pF	test tied to AC Ground
C_{OUT}	Output Capacitance		20	pF	

Zilog

ADVANCED INFORMATION
Product Specification

Z8001® / Z8002®
Z8000® CPU
Central Processing Unit

October 1988

FEATURES

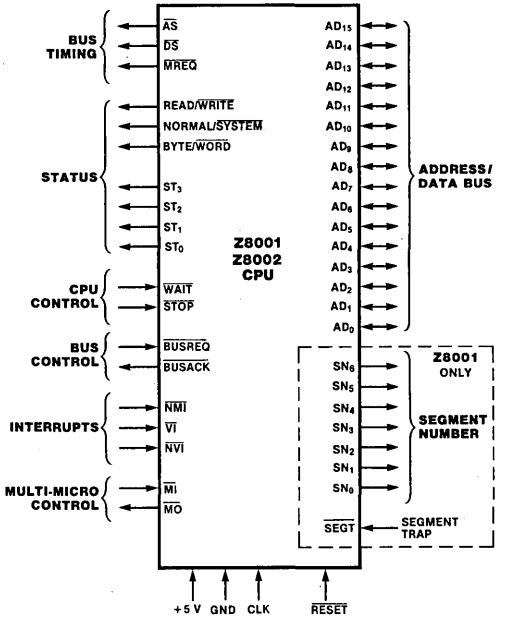
- Regular, easy-to-use architecture
- Instruction set more powerful than many minicomputers
- Directly addresses 8 Mbytes
- Eight user-selectable addressing modes
- Seven data types that range from bits to 32-bit long words and byte and word strings
- System and Normal operating modes
- Separate code, data, and stack spaces
- Sophisticated interrupt structure
- Resource-shaping capabilities for multiprocessing systems
- Multi-programming support
- Compiler support
- Memory management and protection provided by Z8010 Memory Management Unit
- 32-bit operations, including signed multiply and divide
- Z-BUS compatible
- 4, 6, and 10 MHz clock rate

GENERAL DESCRIPTION

The Z8000 is an advanced high-end 16-bit microprocessor that spans a wide variety of applications ranging from simple stand-alone computers to complex parallel-processing systems. Essentially a monolithic minicomputer central processing unit, the Z8000 CPU is characterized by an instruction set more powerful than many minicomputers; abundant resources in registers, data types, addressing modes and addressing range, and a regular architecture that enhances throughput by avoiding critical bottlenecks such as implied or dedicated registers.

CPU resources include sixteen 16-bit general-purpose registers, seven data types that range from bits to 32-bit long words and byte and word strings, and eight user-selectable addressing modes. The 110 distinct instruction types can be combined with the various data types and addressing modes to form a powerful set of 414 instructions. Moreover, the instruction set is regular; most instructions can use any of the five main addressing modes and can operate on byte, word, and long-word data types.

The CPU can operate in either the system or normal mode. The distinction between these two modes permits privileged operations, thereby improving operating system organization and implementation. Multiprogramming is supported by the "atomic" Test and Set instruction; multiprocessing by a combination of instruction and



hardware features; and compilers by multiple stacks, special instructions, and addressing modes.

The Z8000 CPU is offered in three versions: the Z8001/ Z160 segmented CPUs and the Z8002 nonsegmented CPU (Figure 1). The main difference is in addressing range. The Z8001 can directly address 8 megabytes of memory; the Z160 directly addresses 2 megabytes; the Z8002 directly addresses 64 kilobytes. The two operating modes - system and normal - and the distinction between code, data, and stack spaces within each mode allows memory extension up to 48 megabytes for the Z8001, 12 megabytes for the Z160 and 384 kilobytes for the Z8002.

To meet the requirements of complex, memory-intensive applications, a companion memory-management device is offered for the Z8001. The Z8010 Memory Management Unit manages the large address space by providing features such as segment relocation and memory protection. The Z8001 can be used with or without the Z8010. If used by itself, the Z8001 still provides an 8 megabyte direct addressing range, extendable to 48 megabytes.

The Z8001, Z8002 and Z8010 are fabricated with high-density, high-performance scaled n-channel silicon-gate depletion-load technology, and are housed in dual-in-line packages (DIPs) and leadless chip carriers (LCC).

REGISTER ORGANIZATION

The Z8000 CPU is a register-oriented machine that offers sixteen 16-bit general-purpose registers and a set of special system registers. All general-purpose registers can be used as accumulators and all but one as index registers or memory pointers.

Register flexibility is created by grouping and overlapping multiple registers (Figures 2 and 3). For byte operations, the first eight 16-bit registers (R0... R7) are treated as sixteen 8-bit registers (RL0, RH0..., RL7, RH7). The sixteen 16-bit registers are grouped in pairs (RR0... RR14) to form 32-bit long-word registers. Similarly, the register set is grouped in quadruples (RQ0... RQ12) to form 64-bit registers.

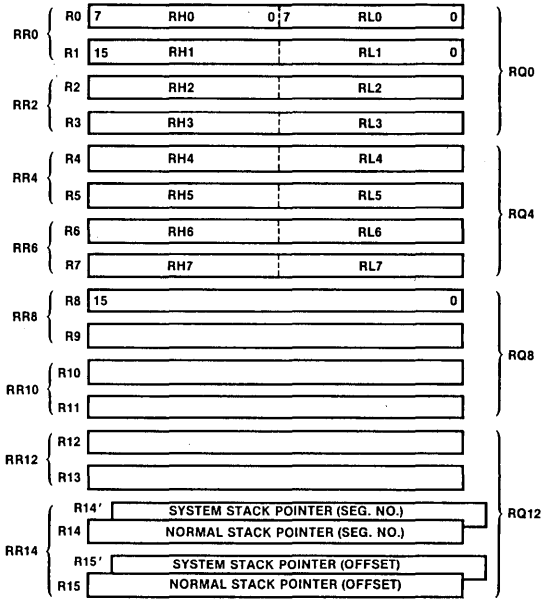


Figure 2. Z8001 General-Purpose Registers

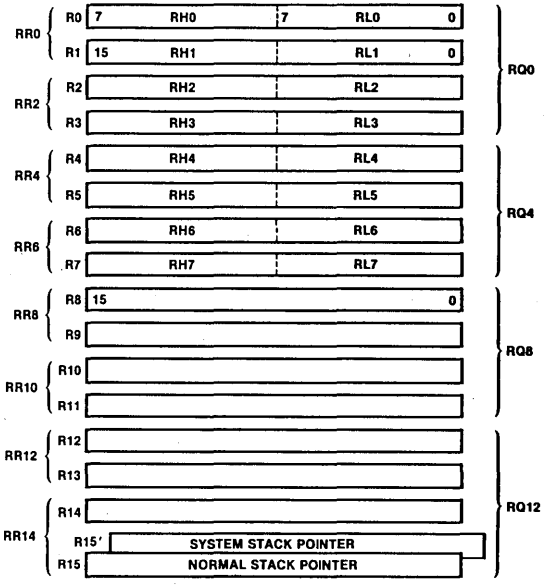


Figure 3. Z8002 General-Purpose Registers

STACKS

The Z8001, Z8002 and Z160 can use stacks located anywhere in memory. Call and Return instructions as well as interrupts and traps use implied stacks. The distinction between normal and system stacks separates system information from the application program information. Two stack pointers are available: the system stack pointer and the normal stack pointer. Because they are part of the general-purpose register group, the user can manipulate the stack pointers with any instruction available for register operations.

In the Z8001, register pair RR14 is the implied stack pointer. Register R14 contains the 7-bit segment number and R15 contains the 16-bit offset. In the Z8002, register R15 is the implied 16-bit stack pointer.

REFRESH

The Z8000 CPU contains a counter that can be used to automatically refresh dynamic memory. The refresh counter register consists of a 9-bit row counter, a 6-bit rate counter, and an enable bit (Figure 4). The 9-bit row counter can address up to 256 rows and is incremented by two each time the rate counter reaches end-of-count. The rate counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler (n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed by 1 to 64 μs with a 4 MHz clock. Refresh can be disabled by programming the refresh enable/disable bit.

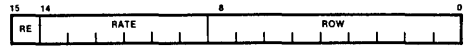


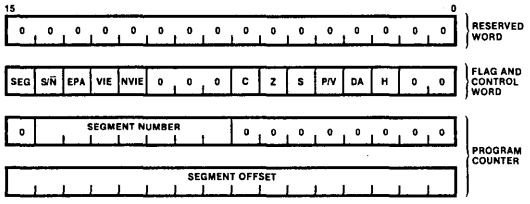
Figure 4. Refresh Counter

PROGRAM STATUS INFORMATION

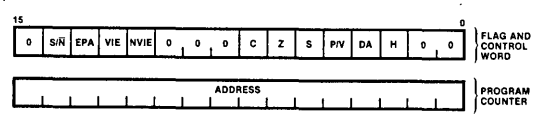
This group of status registers contains the program counter, flags, and control bits. When an interrupt or trap occurs, the entire group is saved and a new program status group is loaded.

Figure 5 illustrates how the program status groups of the Z8001 and Z8002 differ. In the nonsegmented Z8002, the program status group consists of two words: the program counter (PC), and the flag and control word (FCW). In the segmented Z8001, the program status group consists of four words: a two-word program counter, the flag and control word, and an unused word reserved for future use. Seven bits of the first PC word designate one of the 128 memory segments. The second word supplies the 16-bit offset that designates a memory location within the segment.

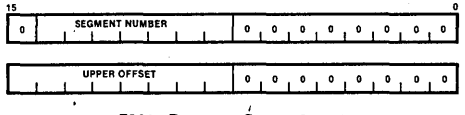
With the exception of the segment enable bit in the Z8001 program status group, the flags and control bits are the same for both CPUs.



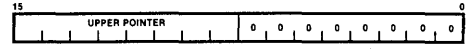
Z8001 Program Status Registers



Z8002 Program Status Registers



Z8001 Program Status Area Pointer



Z8002 Program Status Area Pointer

Figure 5. Z8000 CPU Special Registers

INTERRUPT AND TRAP STRUCTURE

The Z8000 provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions. Both are processed in a similar manner by the CPU.

The CPU supports three types of interrupts (non-maskable, vectored, and non-vectored) and four traps [system call, Extended Process Architecture (EPA) instruction, privileged instructions, and segmentation trap]. The vectored and non-vectored interrupts are maskable. Of the four traps, the only external one is the segmentation trap, which is generated by the Z8010.

The remaining traps occur when instructions limited to the system mode are used in the normal mode, or as a result of the System Call instruction, or for an EPA instruction. The descending order of priority for traps and interrupts is: internal traps, nonmaskable interrupt, segmentation trap, vectored interrupt, and non-vectored interrupt.

When an interrupt or trap occurs, the current program status is automatically pushed on the system stack. The program status consists of the processor status (PC and FCW) plus a 16-bit identifier. The identifier contains the reason or source of the trap or interrupt. For internal traps, the identifier is the first word of the trapped instruction. For external traps or interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle.

After saving the current program status, the new program status is automatically loaded from the program status area in system memory. This area is designated by the program status area pointer (PSAP).

DATA TYPES

Z8000 instructions can operate on bits, BCD digits (4 bits), bytes (8 bits), words (16 bits), long words (32 bits), and byte strings and word strings (up to 64 kilobytes long). Bits can be set, reset, and tested; digits are used in BCD arithmetic operations; bytes are used for characters or small integer values; words are used for integer values, instructions and nonsegmented addresses; long words are used for long integer values and segmented addresses. All data elements except strings can reside either in registers or memory. Strings are stored in memory only.

The basic data element is the byte. The number of bytes used when manipulating a data element is either implied by the operation or—for strings and multiple register operations—explicitly specified in the instruction.

SEGMENTATION AND MEMORY MANAGEMENT

High-level languages, sophisticated operating systems, large programs and data bases, and decreasing memory prices are all accelerating the trend toward larger memory requirements in microcomputer systems. The Z8001 meets this requirement with an eight megabyte addressing space. This large address space is directly accessed by the CPU using a segmented addressing scheme and can be managed by the Z8010 Memory Management Unit.

Segmented Addressing

A segmented addressing space—compared with linear addressing—is closer to the way a programmer uses memory because each procedure and data space resides in its own segment. The 8 megabytes of Z8001 addressing space is divided into 128 relocatable segments up to 64 kilobytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any location relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately. The segmented Z8001 can run any code written for the nonsegmented Z8002 in any one of its 128 segments, provided it is set to the nonsegmented mode.

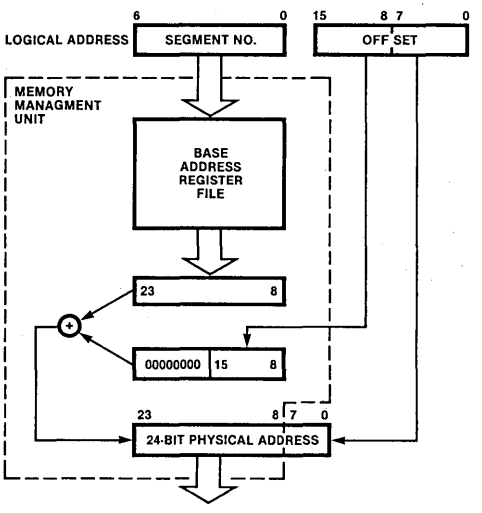


Figure 6. Logical-to-Physical Address Translation

In hardware, segmented addresses are contained in a register pair or long-word memory location. The segment number and offset can be manipulated separately or together by all the available word and long-word operations.

When contained in an instruction, a segmented address has two different representations: long offset and short offset. The long offset occupies two words, whereas the short offset requires only one and combines in one word the 7-bit segment number with an 8-bit offset (range 0-256). The short offset mode allows very dense encoding of addresses and minimizes the need for long addresses required by direct accessing of this large address space.

Memory Management

The addresses manipulated by the programmer, used by instructions and output by the Z8001, are called *logical* addresses. The Memory Management Unit takes the logical addresses and transforms them into the *physical* addresses required for accessing the memory (Figure 6). This address transformation process is called relocation. Segment relocation makes user software addresses independent of the physical memory so the user is freed from specifying where information is actually located in the physical memory.

The relocation process is transparent to user software. A translation table in the Memory Management Unit associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit offset is added to the physical base address to obtain the actual physical address. The system may dynamically reload translation tables as tasks are created, suspended, or changed.

In addition to supporting dynamic segment relocation, the Memory Management Unit also provides segment protection and other segment management features. The protection features prevent illegal uses of segments, such as writing into a write-protected zone.

Each Memory Management Unit stores 64 segment entries that consist of the segment base address, its attributes, size, and status. Segments are variable in size from 256 bytes to 64 kilobytes in increments of 256 bytes. Pairs of Management Units support the 128 segment numbers available for each of the six CPU address spaces. Within an address space, several Management Units can be used to create multiple translation tables.

EXTENDED PROCESSING ARCHITECTURE

The Zilog Extended Processing Architecture (EPA) provides an extremely flexible and modular approach to expanding both the hardware and software capabilities of the Z8000 CPU. Features of the EPA include:

- Specialized instructions for external processors or software traps may be added to CPU instruction set.
- Increases throughput of the system by using up to four specialized external processors in parallel with the CPU.
- Permits modular design of Z8000-based systems.
- Provides easy management of multiple microprocessor configurations via "single instruction stream" communication.
- Simple interconnection between extended processing units and Z8000 CPU requires no additional external supporting logic.
- Supports debugging of suspect hardware against proven software.
- Standard features on all Zilog Z8000 CPUs.

Specific benefits include:

- EPUs can be added as the system grows and as EPUs with specialized functions are developed.
- Control of EPUs is accomplished via a "single instruction stream" in the Z8000 CPU, eliminating many significant system software and bus contention management obstacles that occur in other multiprocessor (e.g., master-slave) organization schemes.

The processing power of the Zilog Z8000 16-bit microprocessor can be boosted beyond its intrinsic capability by Extended Processing Architecture. Simply stated, EPA allows the Z8000 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream (Figure 7).

The use of extended processors to boost the main CPU's performance capability has been proven with large mainframe computers and minicomputers. In these systems, specialized functions such as array processing, special input/output processing, and data communications processing are typically assigned to extended processor hardware. These extended processors are complex computers in their own right.

The Zilog Extended Processing Architecture combines the best concepts of these proven performance boosters with the latest in high-density MOS integrated-circuit design. The result is an elegant expansion of design capability—a powerful microprocessor architecture capable of connecting single-chip EPUs that permits very effective parallel processing and makes for a smoothly integrated instruction stream from the Z8000 programmer's point of view. A typical addition to the current Z8000 instruction set is a set of Floating Point Instructions.

The Extended Processing Units connect directly to the Z8000 Bus (Z-BUS) and continuously monitor the CPU instruction stream. When an extended instruction is detected, the appropriate EPU responds, obtaining or

placing data or status information on the Z-BUS using the Z8000-generated control signals and performing its function as directed.

The Z8000 CPU is responsible for instructing the EPU and delivering operands and data to it. The EPU recognizes instructions intended for it and executes them, using data supplied with the instruction and/or data within its internal registers. There are four classes of EPU instructions:

- Data transfers between main memory and EPU registers
- Data transfers between CPU registers and EPU registers
- EPU internal operations
- Status transfers between the EPUs and the Z8000 CPU Flag and Control Word register (FCW)

Four Z8000 addressing modes may be utilized with transfers between EPU registers and the CPU and main memory:

- Register
- Indirect Register
- Direct Address
- Index

In addition to the hardware-implemented capabilities of the Extended Processing Architecture, there is an extended instruction trap mechanism to permit software simulation of EPU functions. A control bit in the Z8000 FCW register indicates whether actual EPUs are present or not. If not, when an extended instruction is detected, the Z8000 traps on the instruction, so that a software "trap handler" can emulate the desired EPU function—a very useful development tool. The EPA software trap routine supports the debugging of suspect hardware against proven software. This feature will increase in significance as designers become familiar with the EPA capability of the Z8000 CPU.

This software trap mechanism facilitates the design of systems for later addition of EPUs: initially, the extended function is executed as a trap subroutine; when the EPU is finally attached, the trap subroutine is eliminated and the EPA control bit is set. Application software is unaware of the change.

Extended Processing Architecture also offers protection against extended instruction overlapping. Each EPU connects to the Z8000 CPU via the STOP line so that if an EPU is requested to perform a second extended instruction function before it has completed the previous one, it can put the CPU into the Stop/Refresh state until execution of the previous extended instruction is complete.

EPA and CPU instruction execution are shown in Figure 8. The CPU begins operation by fetching an instruction and determining whether it is a CPU or an EPU command. The EPU meanwhile monitors the Z-BUS for its own instructions. If the CPU encounters an EPU command, it checks to see whether an EPU is present; if not, the EPU may be simulated by an EPU instruction trap software routine; if an EPU is present, the necessary data and/or address is placed on the Z-BUS. If the EPU is free when the instruction and data for it appear, the extended instruction is executed. If the EPU is still processing a previous instruction, it activates the CPU's STOP line to lock the CPU off at the Z-BUS until execution is complete. After the instruction is finished, the EPU deactivates the STOP line and CPU transactions continue.

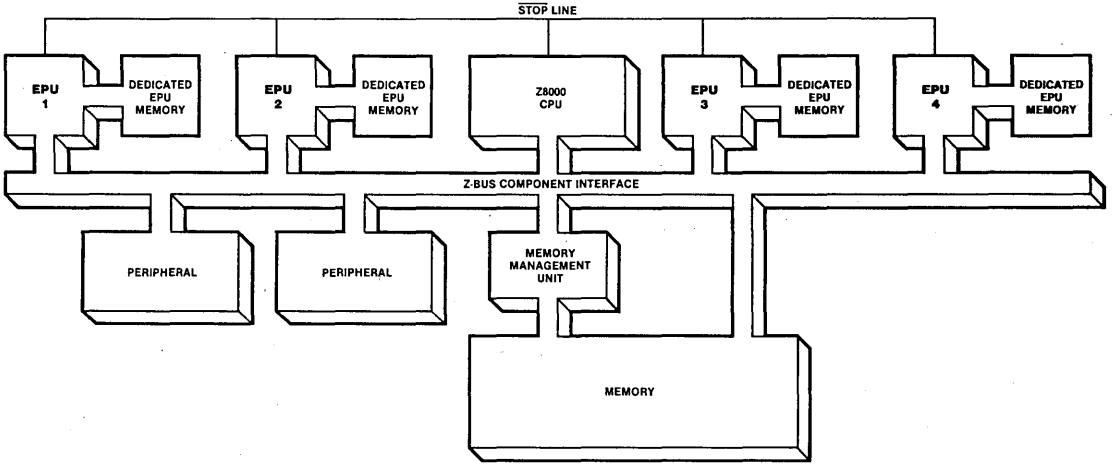


Figure 7. Typical Extended Processor Configuration

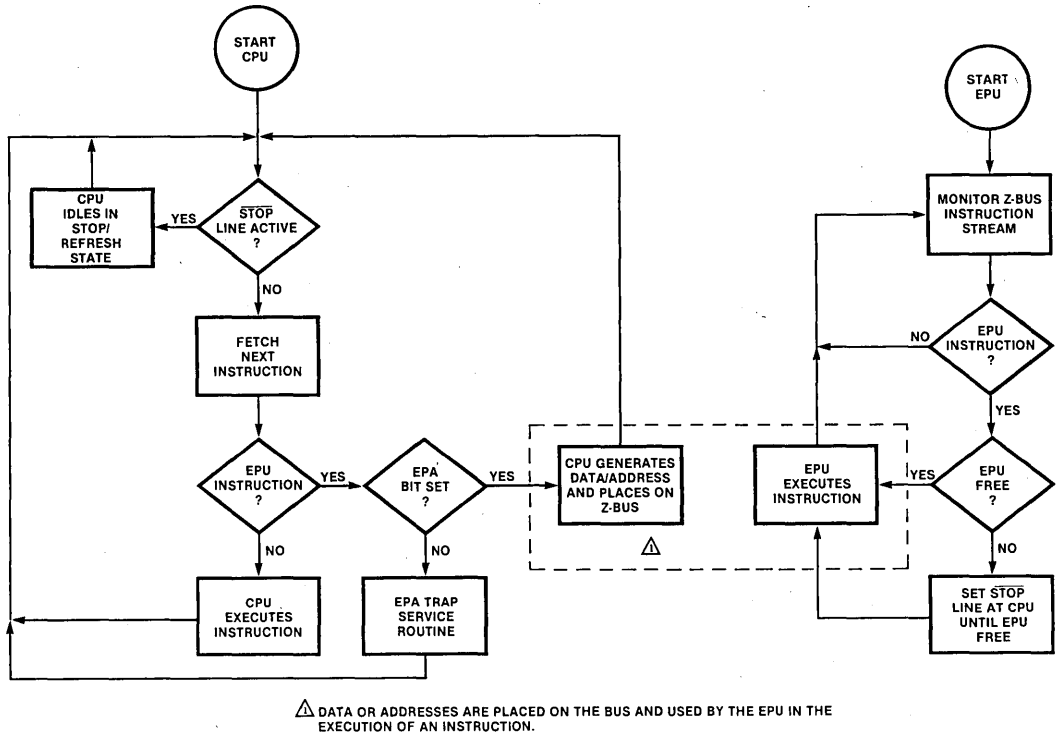


Figure 8. EPA and Z8000 CPU Instruction Execution

INPUT/OUTPUT

A set of I/O instructions performs 8-bit or 16-bit transfers between the CPU and I/O devices. I/O devices are addressed with a 16-bit I/O port address. The I/O port address is similar to a memory address; however, I/O address space need not be part of the memory address space. I/O port and memory addresses coexist on the same bus lines and they are distinguished by the status outputs.

Two types of I/O instructions are available: standard and special. Each has its own address space. The I/O instructions include a comprehensive set of In, Out, and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status information distinguishes between standard and special I/O references.

MULTI-MICROPROCESSOR SUPPORT

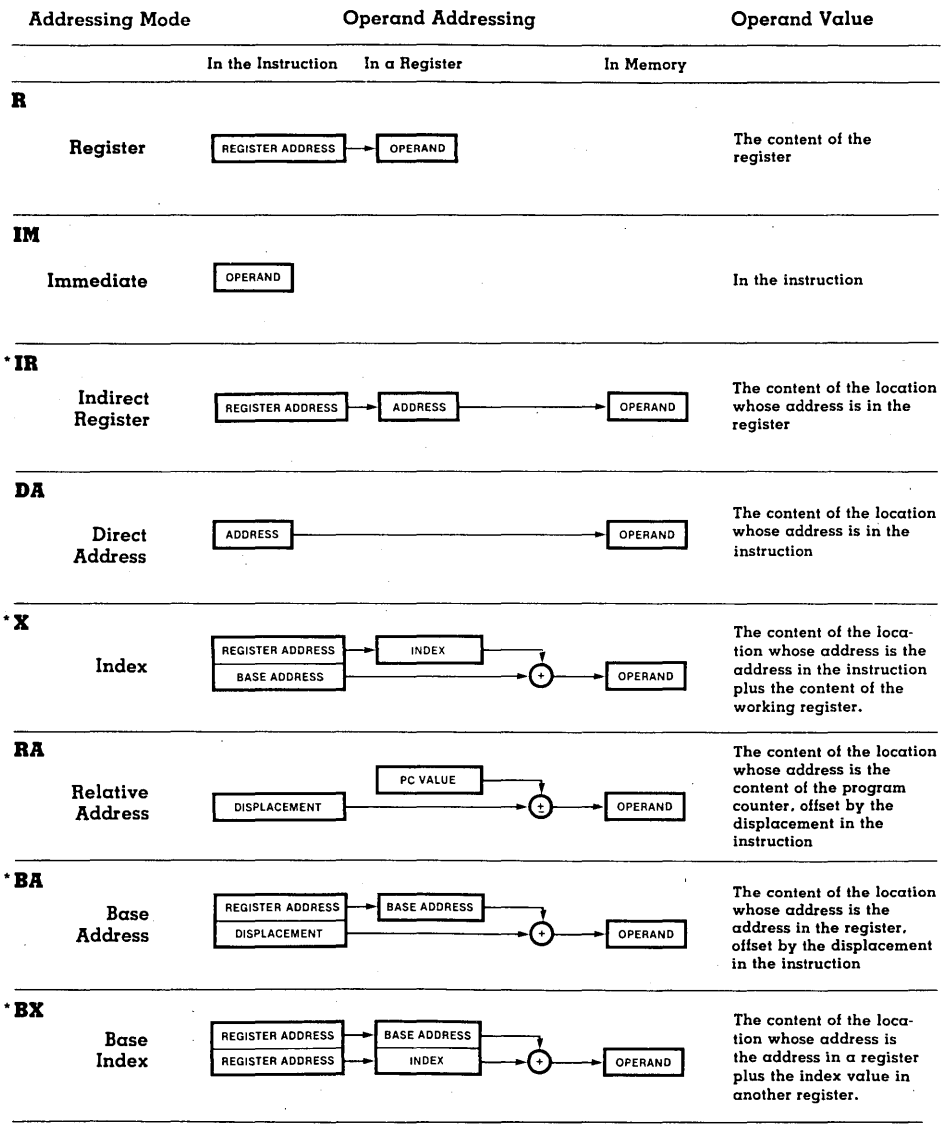
Multi-microprocessor systems are supported in hardware and software. A pair of CPU pins is used in conjunction with certain instructions to coordinate multiple microprocessors. The Multi-Micro Out pin issues a request for the resource, while the Multi-Micro In pin is used to recognize the state of the resource. Thus, any CPU in a multiple microprocessor system can exclude all other asynchronous CPUs from a critical shared resource.

Multi-microprocessor systems are supported in software by the instructions Multi-Micro Request, Test Multi-Micro In, Set Multi-Micro Out, and Reset Multi-Micro Out. In addition, the eight megabyte CPU address space is beneficial in multiple microprocessor systems that have large memory requirements.

ADDRESSING MODES

The information included in Z8000 instructions consists of the function to be performed, the type and size of data elements to be manipulated, and the location of the data elements. Locations are designated by register addresses, memory addresses, or I/O addresses. The addressing mode of a given instruction defines the address space it references and the method used to compute the address itself. Addressing modes are explicitly specified or implied by the instruction.

Figure 9 illustrates the eight addressing modes: Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Index (X), Relative Address (RA), Base Address (BA), and Base Index (BX). In general, an addressing mode explicitly specifies either register address space or memory address space. Program memory address space and I/O address space are usually implied by the instruction.



*Do not use R0 or RR0 as indirect, index, or base registers.

Figure 9. Addressing Modes

INSTRUCTION SET SUMMARY

The Z8000 provides the following types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

LOAD AND EXCHANGE

			Clock Cycles*						
		Addr.	Word, Byte			Long Word			
Mnemonics	**Operands**	**Modes**	**NS**	**SS**	**SL**	**NS**	**SS**	**SL**	**Operation**
CLR	dst	R	7	7	7				**Clear**
CLRB		IR	8	8	8				dst ← 0
		DA	11	12	14				
		X	12	12	15				
EX	R, src	R	6	6	6				**Exchange**
EXB		IR	12	12	12				R ↔ src
		DA	15	16	18				
		X	16	16	19				
LD	R, src	R	3	3	3	5	5	5	**Load into Register**
LDB		IM	7	7	7	11	11	11	R ← src
LDL		IM	5 (byte only)						
		IR	7	7	7	11	11	11	
		DA	9	10	12	12	13	15	
		X	10	10	13	13	13	16	
		BA	14	14	14	17	17	17	
		BX	14	14	14	17	17	17	
LD	dst, R	IR	8	8	8	11	11	11	**Load into Memory** (Store)
LDB		DA	11	12	14	14	15	17	dst ← R
LDL		X	12	12	15	15	15	18	
		BA	14	14	14	17	17	17	
		BX	14	14	14	17	17	11	
LD	dst, IM	IR	11	11	11				**Load Immediate into Memory**
LDB		DA	14	15	17				dst ← IM
		X	15	15	18				
LDA	R, src	DA	12	13	15				**Load Address**
		X	13	13	16				R ← source address
		BA	15	15	15				
		BX	15	15	15				
LDAR	R, src	RA	15	15	15				**Load Address Relative**
									R ← source address
LDK	R, src	IM	5	5	5				**Load Constant**
									R ← n (n = 0... 15)
LDM	R, src, n	IR	11	11	11 + 3n				**Load Multiple**
		DA	14	15	17 + 3n				R ← src (n consecutive words)
		X	15	15	18 + 3n				(n = 1... 16)

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

LOAD AND EXCHANGE (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
LDM	dst, R, n	IR	11	11	11 + 3n				**Load Multiple** (Store Multiple)
		DA	14	15	17 + 3n				dst ← R (n consecutive words)
		X	15	15	18 + 3n				(n = 1... 16)
LDR	R, src	RA	14	14	14	17	17	17	**Load Relative**
LDRB									R ← src
LDRL									(range −32768... +32767)
LDR	dst, R	RA	14	14	14	17	17	17	**Load Relative** (Store Relative)
LDRB									dst ← R
LDRL									(range −32768... +32767)
POP	dst, IR	R	8	8	8	12	12	12	**Pop**
POPL		IR	12	12	12	19	19	19	dst ← IR
		DA	16	16	18	23	23	25	Autoincrement contents of R
		X	16	16	19	23	23	26	
PUSH	IR, src	R	9	9	9	12	12	12	**Push**
PUSHL		IM	12	12	12	19	19	19	Autodecrement contents of R
		IR	13	13	13	20	20	20	IR ← src
		DA	14	14	16	21	21	23	
		X	14	14	17	21	21	24	

ARITHMETIC

ADC	R, src	R	5	5	5				**Add with Carry**
ADCB									R ← R + src + carry
ADD	R, src	R	4	4	4	8	8	8	**Add**
ADDB		IM	7	7	7	14	14	14	R ← R + src
ADDL		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP	R, src	R	4	4	4	8	8	8	**Compare with Register**
CPB		IM	7	7	7	14	14	14	R − src
CPL		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP	dst, IM	IR	11	11	11				**Compare with Immediate**
CPB		DA	14	15	17				dst − IM
		X	15	15	18				
DAB	dst	R	5	5	5				**Decimal Adjust**
DEC	dst, n	R	4	4	4				**Decremented by n**
DECB		IR	11	11	11				dst ← dst − n
		DA	13	14	16				(n = 1... 16)
		X	14	14	17				

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

ARITHMETIC (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
DIV	R, src	R	107	107	107	744	744	744	**Divide** (signed)
DIVL		IM	107	107	107	744	744	744	Word: $R_{n+1} \leftarrow R_{n,n+1}$ + src
		IR	107	107	107	744	744	744	$R_n \leftarrow$ remainder
		DA	108	109	111	745	746	748	Long Word: $R_{n+2,n+3} \leftarrow R_{n...n+3}$ + src
		X	109	109	112	746	746	749	$R_{n,n+2} \leftarrow$ remainder
EXTS	dst	R	11	11	11	11	11	11	**Extend Sign**
EXTSB									Extend sign of low order half of dst
EXTSL									through high order half of dst
INC	dst, n	R	4	4	4				**Increment by n**
INCB		IR	11	11	11				dst ← dst + n
		DA	13	14	16				(n = 1... 16)
		X	14	14	17				
MULT	R, src	R	70	70	70	282†	282†	282†	**Multiply** (signed)
MULTL		IM	70	70	70	282†	282†	282†	Word: $R_{n,n+1} \leftarrow R_{n+1}$ • src
		IR	70	70	70	282†	282†	282†	Long Word: $R_{n...n+3} \leftarrow R_{n+2,n+3}$
		DA	71	72	74	283†	284†	286†	†Plus seven cycles for each 1 in the
		X	72	72	75	284†	284†	287†	multiplicand
NEG	dst	R	7	7	7				**Negate**
NEGB		IR	12	12	12				dst ← 0 − dst
		DA	15	16	18				
		X	16	16	19				
SBC	R, src	R	5	5	5				**Subtract with Carry**
SBCB									R ← R − src − carry
SUB	R, src	R	4	4	4	8	8	8	**Subtract**
SUBB		IM	7	7	7	14	14	14	R ← R − src
SUBL		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

LOGICAL

AND	R, src	R	4	4	4				**AND**
ANDB		IM	7	7	7				R ← R AND src
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				
COM	dst	R	7	7	7				**Complement**
COMB		IR	12	12	12				dst ← NOT dst
		DA	15	16	18				
		X	16	16	19				
OR	R, src	R	4	4	4				**OR**
ORB		IM	7	7	7				R ← R OR src
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

LOGICAL (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
TCC **TCCB**	cc, dst	R	5	5	5				**Test Condition Code** Set LSB if cc is true
TEST	dst	R	7	7	7	13	13	13	**Test**
TESTB		IR	8	8	8	13	13	13	dst OR 0
TESTL		DA	11	12	14	16	17	19	
		X	12	12	15	17	17	20	
XOR **XORB**	R, src	R	4	4	4				**Exclusive OR** R ← R XOR src
		IM	7	7	7				
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				

PROGRAM CONTROL

CALL	dst	IR	10	15	15				**Call Subroutine** Autodecrement SP @ SP ← PC PC ← dst
		DA	12	18	20				
		X	13	18	21				
CALR	dst	RA	10	10	15				**Call Relative** Autodecrement SP @ SP ← PC PC←PC + dst (range −4094 to +4096)
DJNZ **DBJNZ**	R, dst	RA	11	11	11				**Decrement and Jump if Non-Zero** R ← R − 1 If R ≠ 0: PC←PC + dst(range −254 to 9)
IRET†	—	—	13	13	16				**Interrupt Return** PS ← @ SP Autoincrement SP
JP	cc, dst	IR	10	10	15		(taken)		**Jump Conditional** If cc is true: PC ← dst
		IR	7	7	7		(not taken)		
		DA	7	8	10				
		X	8	8	11				
JR	cc, dst	RA	6	6	6				**Jump Conditional Relative** If cc is true: PC ← PC + dst (range −256 to +254)
RET	cc	—	10	10	13		(taken)		**Return Conditional** If cc is true: PC ← @ SP Autoincrement SP
			7	7	7		(not taken)		
SC	src	IM	33	33	39				**System Call** Autodecrement SP @ SP ← old PS Push instruction PS ← System Call PS

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

BIT MANIPULATION

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
BIT **BITB**	dst, b	R IR DA X	4 8 10 11	4 8 11 11	4 8 13 14				**Test Bit Static** Z flag ← NOT dst bit specified by b
BIT **BITB**	dst, R	R	10	10	10				**Test Bit Dynamic** Z flag ← NOT dst bit specified by contents of R
RES **RESB**	dst, b	R IR DA X	4 11 13 14	4 11 14 14	4 11 16 17				**Reset Bit Static** Reset dst bit specified by b
RES **RESB**	dst, R	R	10	10	10				**Reset Bit Dynamic** Reset dst bit specified by contents R
SET **SETB**	dst, b	R IR DA X	4 11 13 14	4 11 14 14	4 11 16 17				**Set Bit Static** Set dst bit specified by b
SET **SETB**	dst, R	R	10	10	10				**Set Bit Dynamic** Set dst bit specified by contents of R
TSET **TSETB**	dst	R IR DA X	7 11 14 15	7 11 15 15	7 11 17 18				**Test and Set** S flag ← MSB of dst dst ← all 1s

ROTATE AND SHIFT

RL **RLB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left** by n bits (n = 1, 2)
RLC **RLCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left through Carry** by n bits (n = 1, 2)
RLDB	R, src	R	9	9	9				**Rotate Digit Left**
RR **RRB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right** by n bits (n = 1, 2)
RRC **RRCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right through Carry** by n bits (n = 1, 2)
RRDB	R, src	R	9	9	9				**Rotate Digit Right**
SDA **SDAB** **SDAL**	dst, R	R		(15 + 3 n)			(15 + 3 n)		**Shift Dynamic Arithmetic** Shift dst left or right by contents of R
SDL **SDLB** **SDLL**	dst, R	R		(15 + 3 n)			(15 + 3 n)		**Shift Dynamic Logical** Shift dst left or right by contents of R

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

ROTATE AND SHIFT (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
SLA **SLAB** **SLAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Arithmetic** by n bits
SLL **SLLB** **SLLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Logical** by n bits
SRA **SRAB** **SRAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Arithmetic** by n bits
SRL **SRLB** **SRLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Logical** by n bits

BLOCK TRANSFER AND STRING MANIPULATION

CPD **CPDB**	R_X,src,R_Y,cc	IR	20	20	20				**Compare and Decrement** R_X – src Autodecrement src address $R_Y \leftarrow R_Y - 1$
CPDR **CPDRB**	R_X,src,R_Y,cc	IR	(11 + 9 n)						**Compare, Decrement, and Repeat** R_X – src Autodecrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPI **CPIB**	R_X,src,R_Y,cc	IR	20	20	20				**Compare and Increment** R_X – src Autoincrement src address $R_Y \leftarrow R_Y - 1$
CPIR **CPIRB**	R_X,src,R_Y,cc	IR	(11 + 9 n)						**Compare, Increment, and Repeat** R_X – src Autoincrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPSD **CPSDB**	dst,src,R,cc	IR	25	25	25				**Compare String and Decrement** dst – src Autodecrement dst and src addresses R ← R – 1
CPSDR **CPSDRB**	dst,src,R,cc	IR	(11 + 14 n)						**Compare String, Decrement, and Repeat** dst – src Autodecrement dst and src addresses R ← R – 1 Repeat until cc is true or R = 0

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset

BLOCK TRANSFER AND STRING MANIPULATION (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CPSI **CPSIB**	dst,src,R,cc	IR	25	25	25				**Compare String and Increment** dst – src Autoincrement dst and src addresses R ← R – 1
CPSIR **CPSIRB**	dst,src,R,cc	IR	(11 + 14 n)						**Compare String, Increment and Repeat** dst – src Autoincrement dst and src addresses R ← R – 1 Repeat until cc is true or R = 0
LDD **LDDB**	dst,src,R	IR	20	20	20				**Load and Decrement** dst ← src Autodecrement dst and src addresses R ← R – 1
LDDR **LDDRB**	dst,src,R	IR	(11 + 9 n)						**Load, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R – 1 Repeat until R = 0
LDI **LDIB**	dst,src,R	IR	20	20	20				**Load and Increment** dst ← src Autoincrement dst and src addresses R ← R – 1
LDIR **LDIRB**	dst,src,R	IR	(11 + 9 n)						**Load, Increment and Repeat** dst ← src Autoincrement dst and src addresses R ← R – 1 Repeat until R = 0
TRDB	dst,src,R	IR	25	25	25				**Translate and Decrement** dst ← src (dst) Autodecrement dst address R ← R – 1
TRDRB	dst,src,R	IR	(11 + 14 n)						**Translate, Decrement and Repeat** dst ← src (dst) Autodecrement dst address R ← R – 1 Repeat until R = 0
TRIB	dst,src,R	IR	25	25	25				**Translate and Increment** dst ← src (dst) Autoincrement dst address R ← R – 1

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
*Privileged instruction. Executed in system mode only.

BLOCK TRANSFER AND STRING MANIPULATION (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
TRIRB	dst,src,R	IR	(11 + 14 n)						**Translate, Increment and Repeat** dst ← src (dst) Autoincrement dst address R ← R − 1 Repeat until R = 0
TRTDB	src1,src2,R	IR	25	25	25				**Translate and Test, Decrement** RH1 ← src2 (src1) Autodecrement src 1 address R ← R − 1
TRTDRB	src1,src2,R	IR	(11 + 14 n)						**Translate and Test, Decrement, and Repeat** RH1 ← src2 (src1) Autodecrement src1 address R ← R − 1 Repeat until R = 0 or RH1 = 0
TRTIB	src1,src2,R	IR	25	25	25				**Translate and Test, Increment** RH1 ← src2 (src1) Autoincrement src1 address R ← R − 1
TRTIRB	src1,src2,R	IR	(11 + 14 n)						**Translate and Test, Increment and Repeat** RH1 ← src2 (src1) Autoincrement src 1 address R ← R − 1 Repeat until R = 0 or RH1 = 0

INPUT/OUTPUT

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
IN†	R,src	IR	10	10	10				**Input**
INB†		DA	12	12	12				R ← src
IND†	dst,src,R	IR	21	21	21				**Input and Decrement**
INDB†									dst ← src Autodecrement dst address R ← R − 1
INDR†	dst,src,R	IR	(11 + 10 n)						**Input, Decrement and Repeat**
INDRB†									dst ← src Autodecrement dst address R ← R − 1 Repeat until R = 0
INI†	dst,src,R	IR	21	21	21				**Input and Increment**
INIB†									dst ← src Autoincrement dst address R ← R − 1

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

INPUT/OUTPUT (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
INIR† **INIRB**†	dst,src,R	IR	(11 + 10 n)						**Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R − 1 Repeat until R = 0
OUT† **OUTB**†	dst,R	IR	10	10	10				**Output** dst ← R
		DA	12	12	12				
OUTD† **OUTDB**†	dst,src,R	IR	21	21	21				**Output and Decrement** dst ← src Autodecrement src address R ← R − 1
OTDR† **OTDRB**†	dst,src,R	IR	(11 + 10 n)						**Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R − 1 Repeat until R = 0
OUTI† **OUTIB**†	dst,src,R	IR	21	21	21				**Output and Increment** dst ← src Autoincrement src address R ← R − 1
OTIR† **OTIRB**†	dst,src,R	IR	(11 + 10 n)						**Output, Increment, and Repeat** dst ← src Autoincrement src address R ← R − 1 Repeat until R = 0
SIN† **SINB**†	R,src	DA	12	12	12				**Special Input** R ← src
SIND† **SINDB**†	dst,src,R	IR	21	21	21				**Special Input and Decrement** dst ← src Autodecrement dst address R ← R − 1
SINDR† **SINDRB**†	dst,src,R	IR	11 + 10 n)						**Special Input, Decrement, and** Repeat dst ← src Autodecrement dst address R ← R − 1 Repeat until R = 0
SINI† **SINIB**†	dst,src,R	IR	21	21	21				**Special Input and Increment** dst ← src Autoincrement dst address R ← R − 1

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

INPUT/OUTPUT (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
SINIR† **SINIRB**†	dst,src,R	IR	(11 + 10 n)						**Special Input, Increment, and** Repeat dst ← src Autoincrement dst address R ← R − 1 Repeat until R = 0
SOUT† **SOUTB**†	dst,src	DA	12	12	12				**Special Output** dst ← src
SOUTD† **SOUTDB**†	dst,src,R	IR	21	21	21				**Special Output and Decrement** dst ← src Autodecrement src address R ← R − 1
SOTDR† **SOTDRB**†	dst,src,R	IR	(11 + 10 n)						**Special Output, Decrement, and** Repeat dst ← src Autodecrement src address R ← R − 1 Repeat until R = 0
SOUTI† **SOUTIB**†	dst,src,R	IR	21	21	21				**Special Output and Increment** dst ← src Autoincrement src address R ← R − 1
SOTIR† **SOTIRB**†	dst,src,R	R	(11 + 10 n)						**Special Output, Increment, and** Repeat dst ← src Autoincrement src address R ← R − 1 Repeat until R = 0

CPU CONTROL

COMFLG	flags	—	7	7	7				**Complement Flag** (Any combination of C, Z, S, P/V)
DI†	int	—	7	7	7				**Disable Interrupt** (Any combination of NVI, VI)
EI†	int	—	7	7	7				**Enable Interrupt** (Any combination of NVI, VI)
HALT†	—	—	(8 + 3 n)						**HALT**
LDCTL†	CTLR,src	R	7	7	7				**Load into Control Register** CTLR ← src
LDCTL†	dst,CTLR	R	7	7	7				**Load from Control Register** dst ← CTLR

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

CPU CONTROL (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
LDCTLB	FLGR,src	R	7	7	7				**Load into Flag Byte Register** FLGR ← src
LDCTLB	dst,FLGR	R	7	7	7				**Load from Flag Byte Register** dst ← FLGR
LDPS†	src	IR	12	16	16				**Load Program Status** PS ← src
		DA	16	20	22				
		X	17	20	23				
MBIT†	—	—	7	7	7				**Test Multi-Micro Bit** Set S if MI is Low; reset S if MI is High
MREQ†	dst	R		(12 + n)					**Multi-Micro Request**
MRES†	—	—	5	5	5				**Multi-Micro Reset**
MSET†	—	—	5	7	7				**Multi-Micro Set**
NOP	—	—	7	7	7				**No Operation**
RESFLG	flag	—	7	7	7				**Reset Flag** (Any combination of C, Z, S, P/V)
SETFLG	flag	—	7	7	7				**Set Flag** (Any combination of C, Z, S, P/V)

*NS = Non-segmented SS = Segmented Short Offset SL = Segmented Long Offset
†Privileged instruction. Executed in system mode only.

CONDITION CODES

Code	Meaning	Flag Settings	CC Field
F	Always false	—	0000
T	Always true	—	1000
Z	Zero	Z = 1	0110
NZ	Not zero	Z = 0	1110
C	Carry	C = 1	0111
NC	No Carry	C = 0	1111
PL	Plus	S = 0	1101
MI	Minus	S = 1	0101
NE	Not equal	Z = 0	1110
EQ	Equal	Z = 1	0110
OV	Overflow	P/V = 1	0100
NOV	No overflow	P/V = 0	1100
PE	Parity is even	P/V = 1	0100
PO	Parity is odd	P/V = 0	1100
GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001
LT	Less than (signed)	(S XOR P/V) = 1	0001
GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010
LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010
UGE	Unsigned greater than or equal	C = 0	1111
ULT	Unsigned less than	C = 1	0111
UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011
ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Note that some condition codes have identical flag settings and binary fields in the instruction:
Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO

STATUS CODE LINES

ST_0-ST_3	Definition
0000	Internal operation
0001	Memory refresh
0010	I/O reference
0011	Special I/O reference (e.g., to an MMU)
0100	Segment trap acknowledge
0101	Non-maskable interrupt acknowledge
0110	Non-vectored interrupt acknowledge
0111	Vectored interrupt acknowledge
1000	Data memory request
1001	Stack memory request
1010	Data memory request (EPU)
1011	Stack memory request (EPU)
1100	Program reference, nth word
1101	Instruction fetch, first word
1110	Extension processor transfer
1111	Reserved

PIN DESCRIPTION

AD_0-AD_{15}. *Address/Data* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used for I/O and to address memory.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates addresses are valid.

$\overline{BUSACK}$. *Bus Acknowledge* (output active Low). A Low on this line indicates the CPU has relinquished control of the bus.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). This line must be driven Low to request the bus from the CPU.

B/$\overline{W}$. *Byte/Word* (output, Low = Word, 3-state). This signal defines the type of memory reference on the 16-bit address/data bus.

CLK. *System Clock* (input). CLK is a 5V single-phase time-base input.

$\overline{DS}$. *Data Strobe* (output, active Low, 3-state). This line times the data in and out of the CPU.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). A Low on this line indicates that the address/data bus holds a memory address.

$\overline{MI}$, $\overline{MO}$. *Multi-Micro In, Multi-Micro Out* (input and output, active Low). These two lines form a resource-request daisy chain that allows one CPU in a multi-microprocessor system to access a shared resource.

$\overline{NMI}$. *Non-Maskable Interrupt* (edge triggered, input, active Low). A high-to-low transition on $\overline{NMI}$ requests a non-maskable interrupt. The $\overline{NMI}$ interrupt has the highest priority of the three types of interrupts.

N/$\overline{S}$. *Normal/System Mode* (output, Low = System Mode, 3-state). N/$\overline{S}$ indicates the CPU is in the normal or system mode.

$\overline{NVI}$. *Non-Vectored Interrupt* (input, active Low). A Low on this line requests a non-vectored interrupt.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the CPU.

R/$\overline{W}$. *Read/Write* (output, Low = Write, 3-state). R/$\overline{W}$ indicates that the CPU is reading from or writing to memory or I/O.

$\overline{SEGT}$. *Segment Trap* (input, active Low). The Memory Management Unit interrupts the CPU with a Low on this line when the MMU detects a segmentation trap. Input on Z8001 only.

SN_0-SN_6. *Segment Number* (outputs, active High, 3-state). These lines provide the 7-bit segment number used to address one of 128 segments by the Z8010 memory Management Unit. Output by the Z8001 only.

ST_0-ST_3. *Status* (outputs, active High, 3-state). These lines specify the CPU status (see Status Code Lines).

$\overline{STOP}$. *Stop* (input, active Low). This input can be used to single-step instruction execution.

$\overline{VI}$. *Vectored Interrupt* (input, active Low). A Low on this line requests a vectored interrupt.

$\overline{WAIT}$. *Wait* (input, active Low). This line indicates to the CPU that the memory or I/O device is not ready for data transfer.

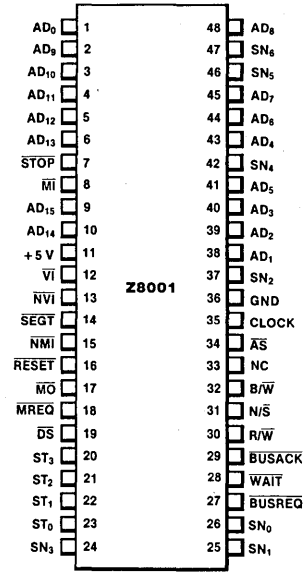


Figure 10a. 48-pin Dual-In-Line Package (DIP), Pin Assignments

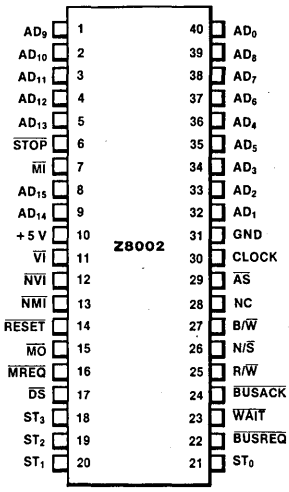
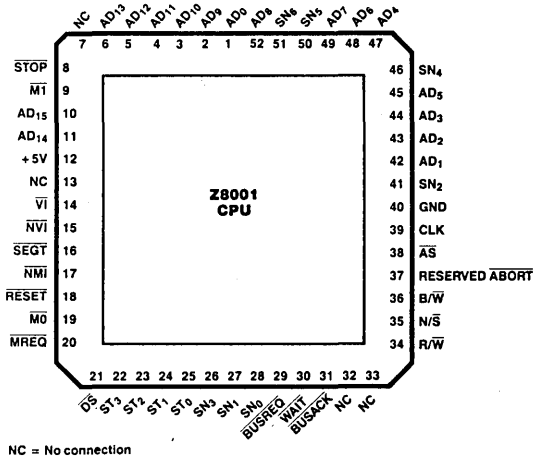


Figure 11a. 40-pin Dual-In-Line Package (DIP), Pin Assignments



NC = No connection

52-pin Chip Carrier, Pin Assignments

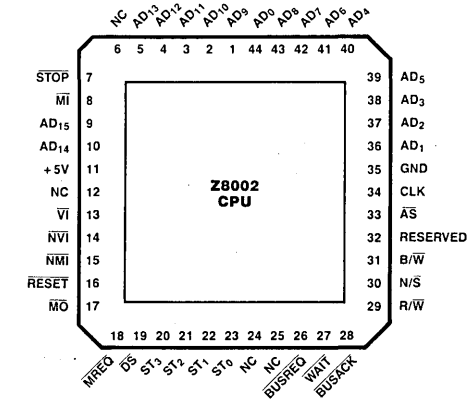


Figure 11b.

44-pin Chip Carrier, Pin Assignments

Z8000 CPU TIMING

The Z8000 CPU executes instructions by stepping through sequences of basic machine cycles, such as memory read or write, I/O device read or write, interrupt acknowledge, and internal execution. Each of these basic cycles requires three to ten clock cycles to execute. Instructions that require more clock cycles to execute are broken up into several machine cycles. Thus no machine cycle is longer than ten clock cycles and fast response to a Bus Request is guaranteed.

The instruction opcode is fetched by a normal memory read operation. A memory refresh cycle can be inserted just after the completion of any first instruction fetch (IF_1) cycle and can also be inserted while the following instructions are being executed: MULT, MULTL, DIV, DIVL, HALT, all Shift instructions, all Block Move instructions, and the Multi-Micro Request instruction (MREQ).

The following timing diagrams show the relative timing relationships of all CPU signals during each of the basic operations. When a machine cycle requires additional clock cycles for CPU internal operation, one to five clock cycles are added. Memory and I/O read and write, as well as interrupt acknowledge cycles, can be extended by activating the $\overline{WAIT}$ input. For exact timing information, refer to the composite timing diagram.

Note that the $\overline{WAIT}$ input is not synchronized in the Z8000 and that the setup and hold times for $\overline{WAIT}$, relative to the clock, must be met. If asynchronous $\overline{WAIT}$ signals are generated, they must be synchronized with the CPU clock before entering the Z8000.

MEMORY READ AND WRITE

Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 outputs. During a memory read cycle, a 16-bit address is placed on the AD_0-AD_{15} outputs early in the first clock period, as shown in Figure 12. In the Z8001, the 7-bit segment number is output on SN_0-SN_6 one clock period earlier than the 16-bit address offset.

A valid address is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The state of the $\overline{WAIT}$ input is sampled in the middle of the second clock cycle by the falling edge of Clock. If $\overline{WAIT}$ is Low, an additional clock period is added between T_2 and T_3. $\overline{WAIT}$ is sampled again in the middle of this wait cycle, and additional wait states can be inserted: this allows interfacing slow memories. No control outputs change during wait states.

Although Z8000 memory is word organized, memory is addressed as bytes. All instructions are word-aligned, using even addresses. Within a 16-bit word, the most significant byte (D_8-D_{15}) is addressed by the low-order address (A_0 = Low), and the least significant byte (D_0-D_7) is addressed by the high-order address (A_0 = High).

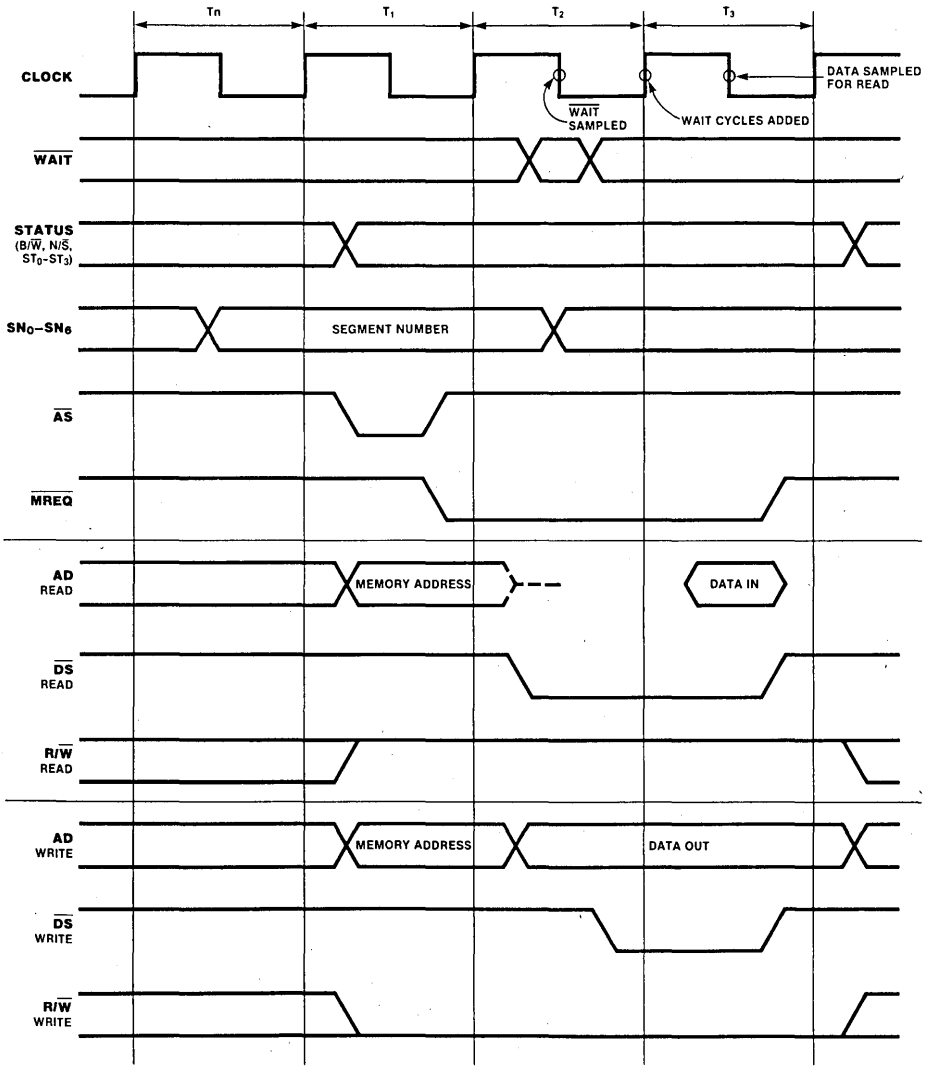


Figure 12. Memory Read and Write Timing

INPUT/OUTPUT

I/O timing is similar to memory read/write timing, except that one wait state is automatically (T_{WA}) inserted between T_2 and T_3 (Figure 13). Both the segmented Z8001/Z8005 and the nonsegmented Z8002 use 16-bit I/O addresses.

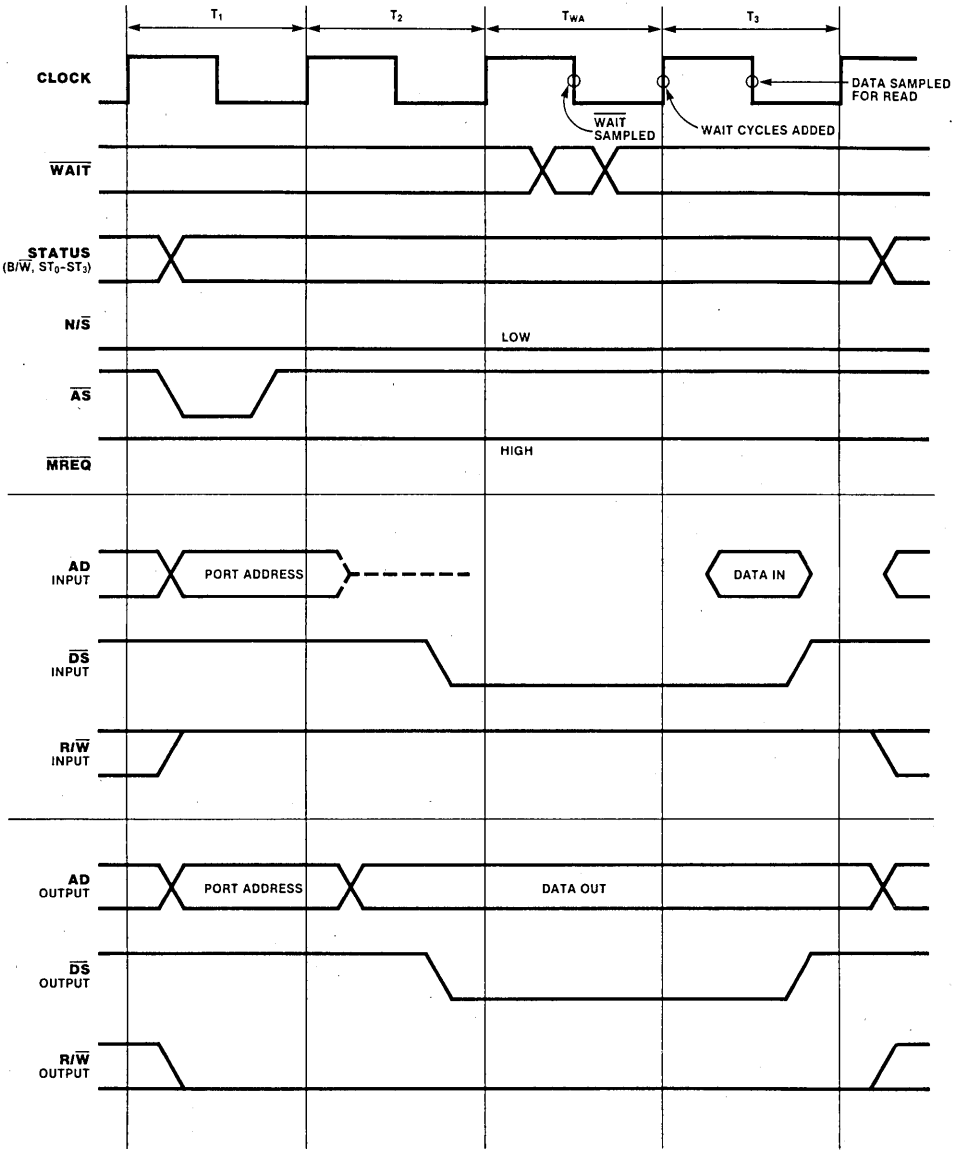


Figure 13. Input/Output Timing

INTERRUPT AND SEGMENT TRAP REQUEST AND ACKNOWLEDGE

The Z8000 CPU recognizes three interrupt inputs (non-maskable, vectored, and nonvectored) and a segmentation trap input. Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge detected and sets the internal $\overline{NMI}$ latch. The $\overline{VI}$, $\overline{NVI}$, and $\overline{SEGT}$ inputs, as well as the state of the internal NMI latch, are sampled at the end of T_2 in the last machine cycle of any instruction.

In response to an interrupt or trap, the subsequent IF_1 cycle is exercised, but ignored. The internal state of the CPU is not altered and the instruction will be refetched and executed after the return from the interrupt routine. The program counter is not updated, but the system stack pointer is decremented in preparation for pushing starting information onto the system stack.

The next machine cycle is the interrupt acknowledge cycle. This cycle has five automatic wait states, with additional wait states possible, as shown in Figure 14.

After the last wait state, the CPU reads the information on AD_0-AD_{15} and temporarily stores it, to be saved on the stack later in the acknowledge sequence. This word identifies the source of the interrupt or trap. For the nonvectored and nonmaskable interrupts, all 16 bits can represent peripheral device status information. For the vectored interrupt, the low byte is the jump vector, and the high byte can be extra user status. For the segmentation trap, the *high* byte is the Memory Management Unit identifier and the *low* byte is undefined.

After the acknowledge cycle, the N/$\overline{S}$ output indicates the automatic change to system mode.

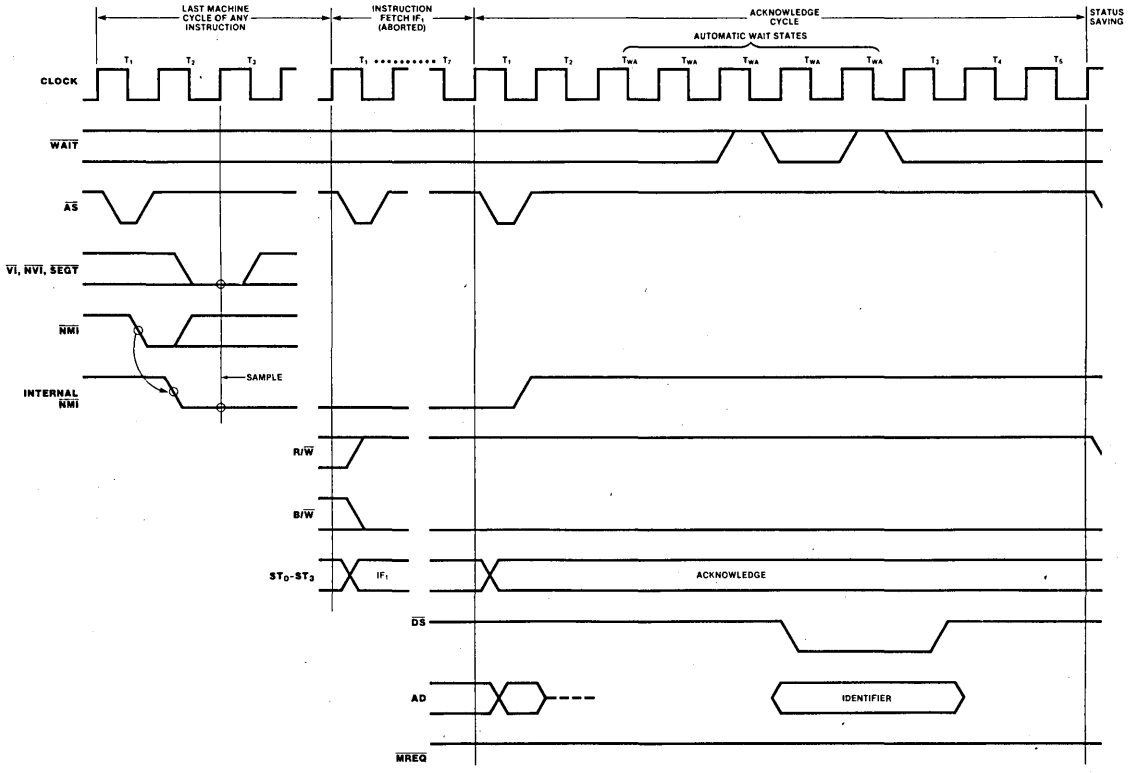


Figure 14. Interrupt and Segment Trap Request/Acknowledge Timing

STATUS SAVING SEQUENCE

The machine cycles, following the interrupt acknowledge or segmentation trap acknowledge cycle, push the old status information on the system stack in the following order: the 16-bit program counter; the 7-bit segment number (Z8001/Z8005 only); the flag control word; and finally the interrupt/trap identifier. Subsequent machine cycles fetch the new program status from the program status area, and then branch to the interrupt/trap service routine.

BUS REQUEST ACKNOWLEDGE TIMING

A Low on the $\overline{\text{BUSREQ}}$ input indicates to the CPU that another device is requesting the Address/Data and control buses. The asynchronous $\overline{\text{BUSREQ}}$ input is synchronized at the beginning of any machine cycle (Figure 15). $\overline{\text{BUSREQ}}$ takes priority over WAIT. If $\overline{\text{BUSREQ}}$ is Low, an internal synchronous $\overline{\text{BUSREQ}}$ signal is generated, which—after completion of the current machine cycle—causes the $\overline{\text{BUSACK}}$ output to go Low and all bus outputs to go into the high-impedance state. The requesting device—typically a DMA—can then control the bus.

When $\overline{\text{BUSREQ}}$ is released, it is synchronized with the rising clock edge; the $\overline{\text{BUSACK}}$ output goes High one clock period later, indicating that the CPU will again take control of the bus.

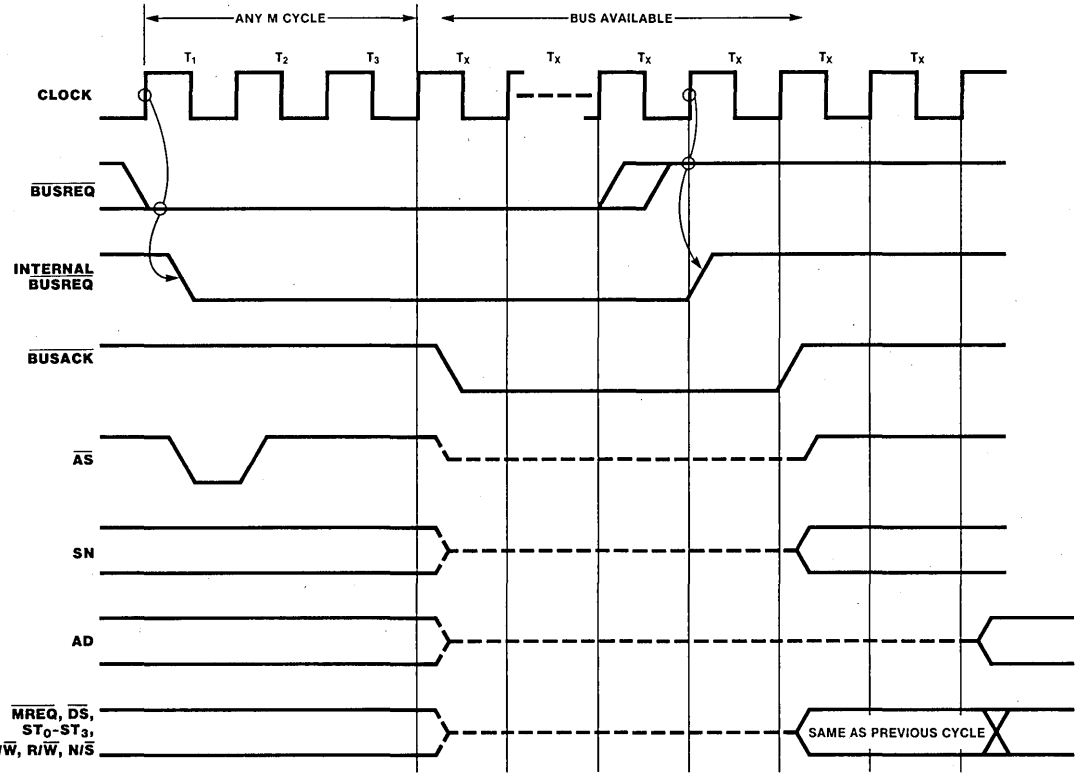


Figure 15. Bus Request/Acknowledge Timing

STOP

The STOP input is sampled by the last falling clock edge immediately preceding any IF_1 cycle (Figure 16) and before the second word of an EPA instruction is fetched. If $\overline{STOP}$ is found Low during the IF_1 cycle, a stream of memory refresh cycles is inserted after T_3, again sampling the $\overline{STOP}$ input on each falling clock edge in the middle of the T_3 states. During the EPA instruction, both EPA instruction words are fetched but any data transfer or subsequent instruction fetch is postponed until $\overline{STOP}$ is sampled High. This refresh operation does not use the refresh prescaler or its divide-by-four clock prescaler; rather, it double-increments the refresh counter every three clock cycles. When $\overline{STOP}$ is found High again, the next refresh cycle is completed, any remaining T states of the IF_1 cycle are then executed, and the CPU continues its operation.

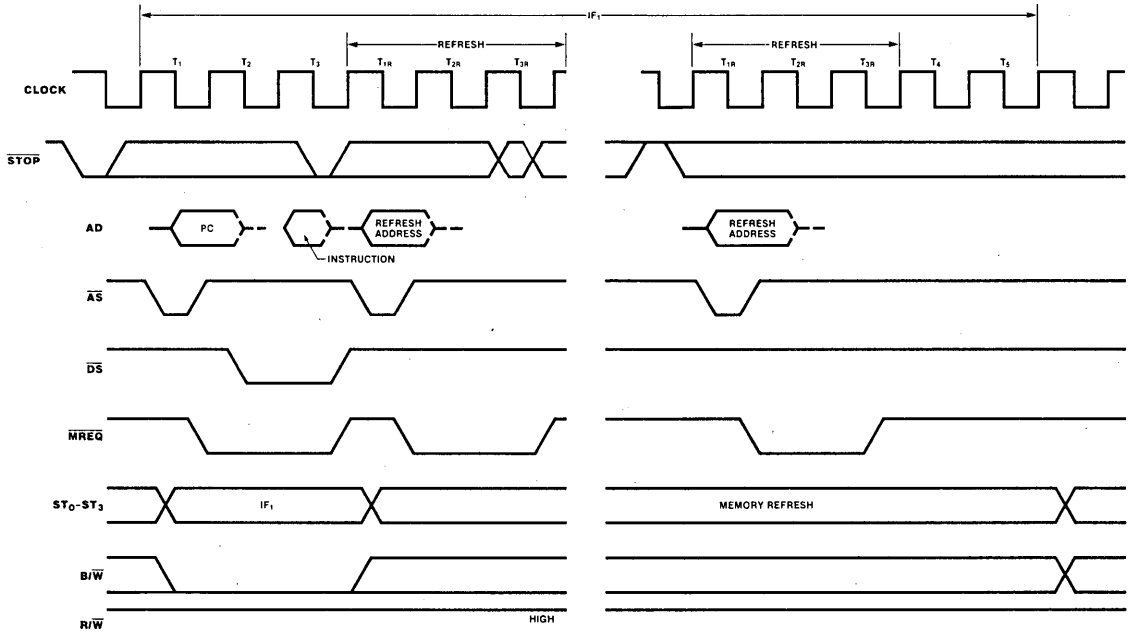


Figure 16. Stop Timing

INTERNAL OPERATION

Certain extended instructions, such as Multiply and Divide, and some special instructions need additional time for the execution of internal operations. In these cases, the CPU goes through a sequence of internal operation machine cycles, each of which is three to eight clock cycles long (Figure 17). This allows fast response to Bus Request and Refresh Request, because bus request or refresh cycles can be inserted at the end of any internal machine cycle.

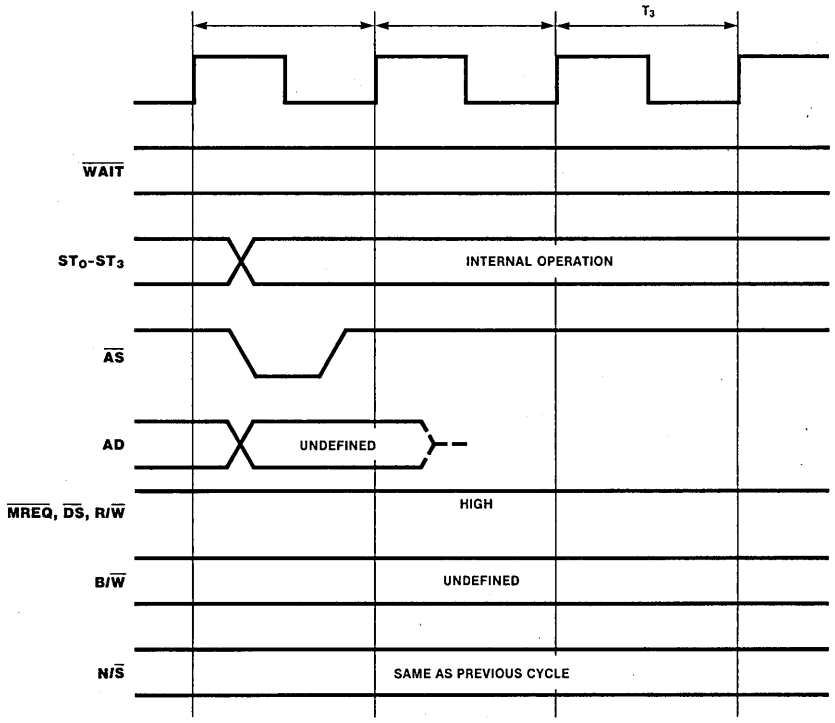


Figure 17. Internal Operation Timing

HALT

A HALT instruction executes an unlimited number of 3-cycle internal operations, interspersed with memory refresh cycles whenever requested. An interrupt, segmentation trap, or reset are the only exits from a HALT instruction.

The CPU samples the $\overline{VI}$, $\overline{NVI}$, $\overline{NMI}$, and $\overline{SEGT}$ inputs at the beginning of every T_3 cycle. If an input is found active during two consecutive samples, the subsequent IF_1 cycle is exercised, but ignored, and the normal interrupt acknowledge cycle is started.

MEMORY REFRESH

When the 6-bit prescaler in the refresh counter has been decremented to zero, a refresh cycle consisting of three T-states is started as soon as possible (that is, after the next IF_1 cycle or Internal Operation cycle).

The 9-bit refresh counter value is put on the low-order side of the address bus (AD_0-AD_8); AD_9-AD_{15} are undefined (Figure 18). Since the memory is word-organized, A_0 is always Low during refresh and the refresh counter is always incremented by two, thus stepping through 256 consecutive refresh addresses on AD_1-AD_8. Unless disabled, the presettable prescaler runs continuously and the delay in starting a refresh cycle is therefore not cumulative.

While the $\overline{STOP}$ input is Low, a continuous stream of memory refresh cycles, each three T-states long, is executed without using the refresh prescaler.

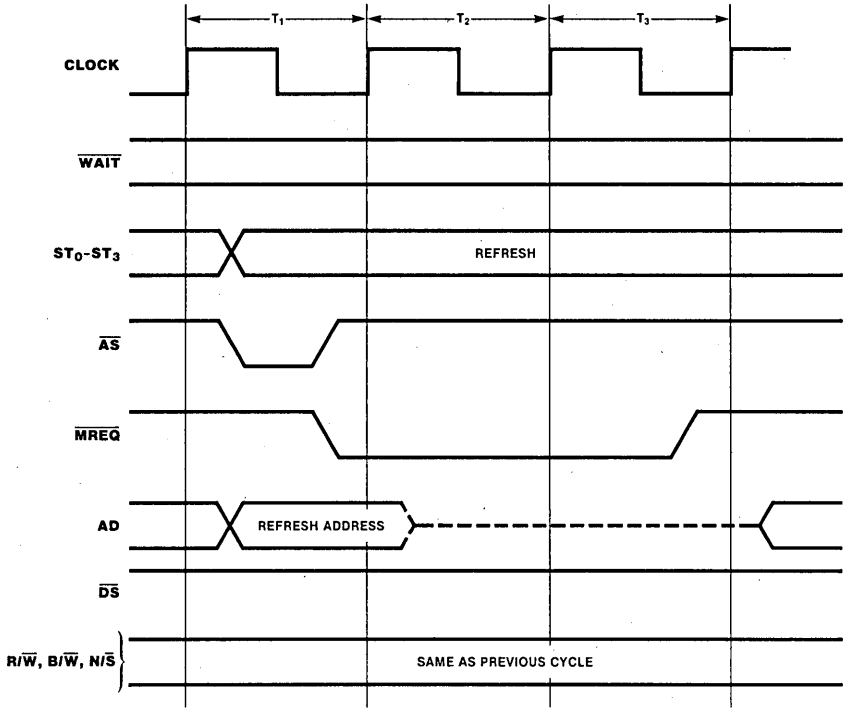


Figure 18. Memory Refresh Timing

RESET

A Low on the $\overline{RESET}$ input causes the following results within five clock cycles (Figure 19):

- AD_0-AD_{15} are 3-stated
- $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, ST_0-ST_3, $\overline{BUSACK}$, and $\overline{MO}$ are forced High
- SN_0-SN_6 are forced Low
- Refresh is disabled
- R/$\overline{W}$, B/$\overline{W}$, and N/$\overline{S}$ are not affected

When $\overline{RESET}$ has been High for three clock periods, three consecutive memory read cycles are executed in the system mode for the Z8001. The Z8002 has two consecutive read cycles. In the Z8001, the first cycle reads the flag and control word from location 0002, the next reads the 7-bit program counter segment number from location 0004, the next reads the 16-bit PC offset from location 0006, and the following IF_1 cycle starts the program. In the Z8002, the first cycle reads the flag and control word from location 0002, the next reads the PC from location 0004, and the following IF_1 cycle starts the program.

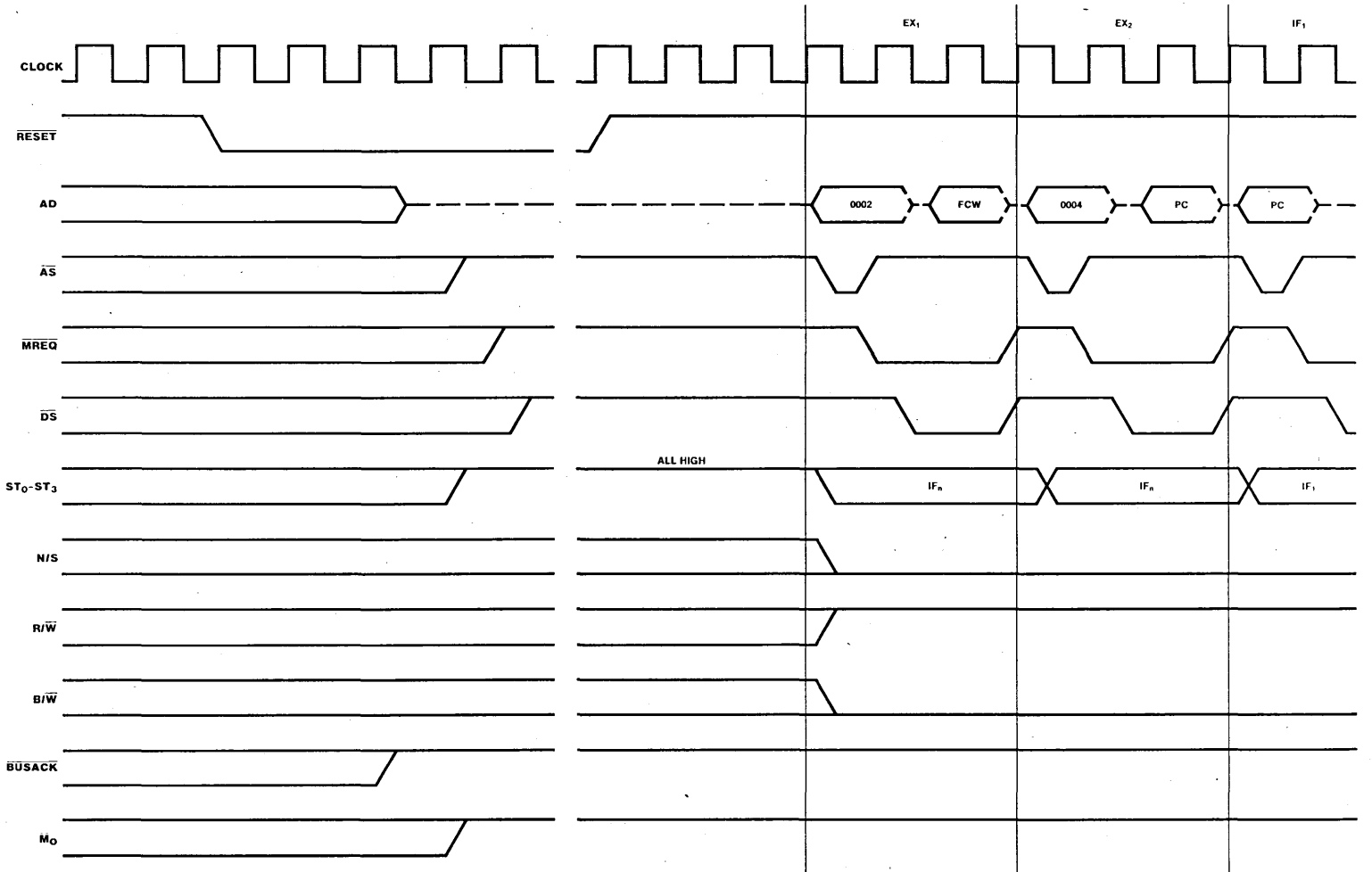
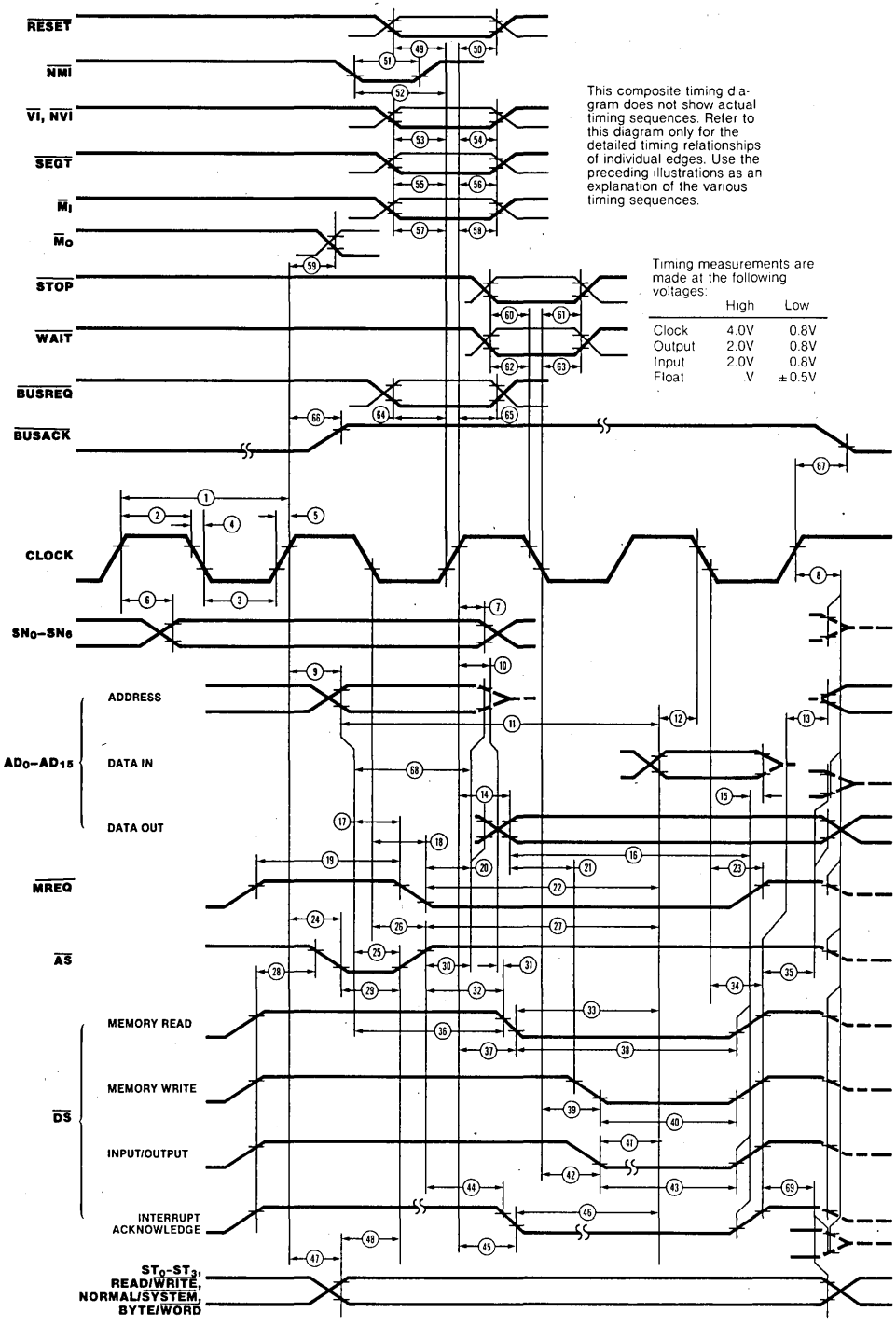


Figure 19. Reset Timing

COMPOSITE AC TIMING DIAGRAM



This composite timing diagram does not show actual timing sequences. Refer to this diagram only for the detailed timing relationships of individual edges. Use the preceding illustrations as an explanation of the various timing sequences.

Timing measurements are made at the following voltages:

	High	Low
Clock	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	V	±0.5V

AC CHARACTERISTICS†

Number	Symbol	Parameter	Z8001/2 4 MHz Min	Z8001/2 4 MHz Max	Z8001/2 6 MHz Min	Z8001/2 6 MHz Max	Z8001/2 10 MHz Min	Z8001/2 10 MHz Max
1	TcC	Clock Cycle Time	250	2000	165	2000	100	2000
2	TwCh	Clock Width (High)	105	1895	70	1930	40	1960
3	TwCl	Clock Width (Low)	105	1895	70	1930	40	1960
4	TfC	Clock Fall Time		20		10		10
5	TrC	Clock Rise Time		20		15		10
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pf load)		130		110		90
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		10		0	
8	TdC(Bz)	Clock ↑ to Bus Float		65		55		50
9	TdC(A)	Clock ↑ to Address Valid		100		75		55
10	TdC(Az)	Clock ↑ to Address Float		65		55		50
11	TdA(DR)	Address Valid to Read Data Required Valid		475*		305*		180*
12	TsDR(C)	Read Data to Clock ↓ Setup time	30		20		10	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80*		45*		20*	
14	TdC(DW)	Clock ↑ to Write Data Valid		100		75		60
15	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		0	
16	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	295*		195*		110*	
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55*		35*		20*	
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80		70		50
19	TwMRh	$\overline{MREQ}$ Width (High)	210*		135*		80*	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70*		35*		20*	
21	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	55*		35*		15*	
22	TdMR(DR)	$\overline{MREQ}$ ↓ to Read Data Required Valid		370*		230*		140*
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		80		60		50
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		60		45
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55*		35*		20*	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		80		45
27	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360*		220*		140*
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70*		35*		15*	
29	TwAS	$\overline{AS}$ Width (Low)	85*		55*		30*	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	70*		45*		20*	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0		0	
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	80*		55*		30*	
33	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid		205*		130*		70*
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65		50
35	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid	75*		45*		25*	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180*		110*		65*	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		85		65
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275*		185*		110*	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		80		65
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185*		110*		75*	

*Clock-cycle time-dependent characteristics. See Footnotes to AC Characteristics.
†Units in nanoseconds (ns).

AC CHARACTERISTICS† (Continued)

Number	Symbol	Parameter	Z8001/2 4 MHz Min	Z8001/2 4 MHz Max	Z8001/2 6 MHz Min	Z8001/2 6 MHz Max	Z8001/2 10 MHz Min	Z8001/2 10 MHz Max
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid		330*		210*		120*
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		90		65
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410*		255*		160*	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065*		690*		410*	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		85		70
46	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Required Delay		455*		295*		165*
47	TdC(S)	Clock ↑ to Status Valid Delay		110		85		65
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	50*		30*		20*	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		70		50	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		0		0	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		70		50	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		70		50	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		50		40	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	20		20		10	
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		55		40	
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		0		0	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		140		80	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		0		0	
59	TdC(MO)	Clock ↑ to $\overline{MO}$ Delay		120		85		80
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		100		50	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		0		0	
62	TsW(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		30		20	
63	ThW(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		10		5	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	90		80		60	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		10		5	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		100		75		65
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		100		75		65
68	TwA	Address Valid Width	150*		95*		50*	
69	TdDS(S)	$\overline{DS}$ ↑ to STATUS Not Valid	80*		55*		30*	

*Clock-cycle time-dependent characteristics. See Footnotes to AC Characteristics.
†Units in nanoseconds (ns).

FOOTNOTES TO AC CHARACTERISTICS

Number	Symbol	Z8001/2 4 MHz Equation	Z8001/2 6 MHz Equation	Z8001/2 10 MHz Equation
11	TdA(DR)	2TcC + TwCh – 130 ns	2TcC + TwCh – 95 ns	2TcC + TwCh – 60 ns
13	TdDS(A)	TwCl – 25 ns	TwCl – 25 ns	TwCl – 20 ns
16	TdDW(DS)	TcC + TwCh – 60 ns	TcC + TwCh – 40 ns	TcC + TwCh – 30 ns
17	TdA(MR)	TwCh – 50 ns	TwCh – 35 ns	TwCh – 20 ns
19	TwMRh	TcC – 40 ns	TcC – 30 ns	TcC – 20 ns
20	TdMR(A)	TwCl – 35 ns	TwCl – 35 ns	TwCl – 20 ns
21	TdDW(DSW)	TwCh – 50 ns	TwCh – 35 ns	TwCh – 25 ns
22	TdMR(DR)	2TcC – 130 ns	2TcC – 100 ns	2TcC – 60 ns
25	TdA(AS)	TwCh – 50 ns	TwCh – 35 ns	TwCh – 20 ns
27	TdAS(DR)	2TcC – 140 ns	2TcC – 110 ns	2TcC – 60 ns
28	TdDS(AS)	TwCl – 35 ns	TwCl – 35 ns	TwCl – 25 ns
29	TwAS	TwCh – 20 ns	TwCh – 15 ns	TwCh – 10 ns
30	TdAS(A)	TwCl – 35 ns	TwCl – 25 ns	TwCl – 20 ns
32	TdAS(DSR)	TwCl – 25 ns	TwCl – 15 ns	TwCl – 10 ns
33	TdDSR(DR)	TcC + TwCh – 150 ns	TcC + TwCh – 105 ns	TcC + TwCh – 70 ns
35	TdDS(DW)	TwCl – 30 ns	TwCl – 25 ns	TwCl – 15 ns
36	TdA(DSR)	TcC – 70 ns	TcC – 55 ns	TcC – 35 ns
38	TwDSR	TcC + TwCh – 80 ns	TcC + TwCh – 50 ns	TcC + TwCh – 30 ns
40	TwDSW	TcC – 65 ns	TcC – 55 ns	TcC – 25 ns
41	TdDSI(DR)	2TcC – 170 ns	2TcC – 120 ns	2TcC – 80 ns
43	TwDS	2TcC – 90 ns	2TcC – 75 ns	2TcC – 40 ns
44	TdAS(DSA)	4TcC + TwCl – 40 ns	4TcC + TwCl – 40 ns	4TcC + TwCl – 30 ns
46	TdDSA(DR)	2TcC + TwCh – 150 ns	2TcC + TwCh – 105 ns	2TcC + TwCh – 75 ns
48	TdS(AS)	TwCh – 55 ns	TwCh – 40 ns	TwCh – 30 ns
68	TwA	TcC – 90 ns	TcC – 70 ns	TcC – 50 ns
69	TdDS(s)	TwCl – 25 ns	TwCl – 15 ns	TwCl – 10 ns

AC Timing Test Conditions

V_{OL} = 0.8V
V_{OH} = 2.0V
V_{IL} = 0.8V
V_{IH} = 2.4V
V_{ILC} = 0.45V
V_{IHC} = V_{CC} – 0.4V

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

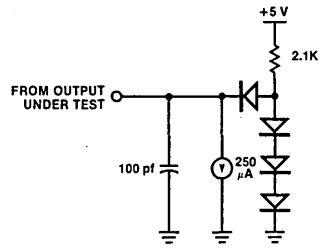
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

- S = 0°C to +70°C, +4.75V ≤ V_{CC} ≤ +5.25V
- **E = -40°C to +100°C, +4.75V ≤ Vcc ≤ +5.25V**



All ac parameters assume a total load capacitance (including parasitic capacitances) or 100 pf max, except for parameter 6 (50 pf max). Timing references between two output signals assume a load difference of 50 pf max.

The Ordering Information section lists package temperature ranges and product numbers.

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC} − 0.4	V_{CC} + 0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
$V_{IH\ RESET}$	Input High Voltage on $\overline{RESET}$ pin	2.4	V_{CC} + 0.3	V	
$V_{IH\ NMI}$	Input High Voltage on NMI pin	2.4	V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	0.4≤V_{IN}≤ +2.4V
$I_{IL\ SEGT}$	Input Leakage on $\overline{SEGT}$ pin	−100	100	µA	
I_{OL}	Output Leakage		±10	µA	0.4≤V_{IN}≤ +2.4V
I_{CC}	V_{CC} Power Supply Current		300	mA	4 MHz and 6 MHz commercial
			400	mA	Extended temperature range
			400	mA	10 MHz speed range

Z8010 Z8000® MMU
Memory Management Unit

October 1988

Features

- Dynamic segment relocation makes software addresses independent of physical memory addresses.
- Sophisticated memory-management features include access validation that protects memory areas from unauthorized or unintentional access, and a write-warning indicator that predicts stack overflow.
- For use with both Z8001 and Z8003 CPU.
- 64 variable-sized segments from 256 to 65,536 bytes can be mapped into a total physical address space of 16M bytes; all 64 segments are randomly accessible.
- Multiple MMUs can support several translation tables for each Z8001 address space.
- MMU architecture supports multi-programming systems and virtual memory implementations.

General Description

The Z8010 Memory Management Unit (MMU) manages the large 8M byte addressing spaces of the Z8001 CPU. The MMU provides dynamic segment relocation as well as numerous memory protection features.

Dynamic segment relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is actually located in the physical memory. It also provides a flexible, efficient method for supporting multi-programming systems. The MMU uses a translation table to transform the 23-bit logical address output from the Z8001 CPU into a 24-bit address for the physical memory. (Only logical memory addresses go to an MMU for translation; I/O addresses and data, in general, must by pass this component.)

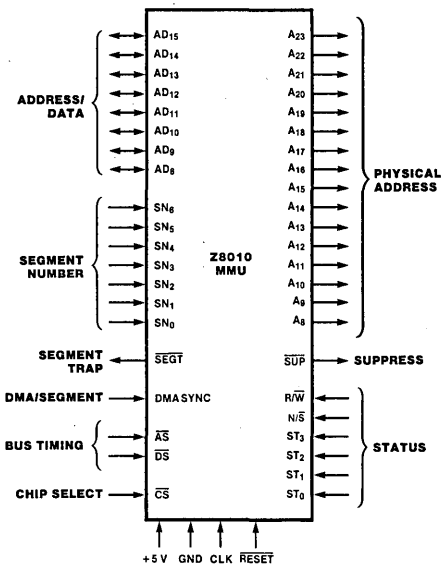


Figure 1. Pin Functions

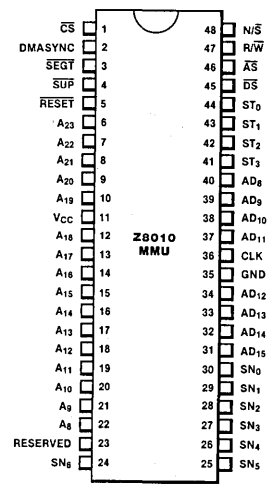


Figure 2. 48-pin Dual-In-Line Package (DIP), Pin Assignments

General Description (Continued)

Memory segments are variable in size from 256 bytes to 64K bytes, in increments of 256 bytes. Pairs of MMUs support the 128 segment numbers available for the various Z8001 CPU address spaces. Within an address space, any number of MMUs can be used to accommodate multiple translation tables for System and Normal operating modes, or to support more sophisticated memory-management systems.

MMU memory-protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each segment. A segment is assigned a number of attributes when its descriptor is entered into the MMU. When a memory reference is made, these attributes are checked against the status information supplied by the Z8001 CPU. If a mismatch occurs, a trap is generated and the CPU is interrupted. The CPU can then check the status registers of the MMU to determine the cause.

Segments are protected by modes of permitted use, such as read only, system only, execute only and CPU-access only. Other segment management features include a write-warning zone useful for stack operations and status flags that record read or write accesses to each segment.

The MMU is controlled via 22 Special I/O instructions from the Z8000 CPU in System mode. With these instructions, system software can assign program segments to arbitrary memory locations, restrict the use of segments and monitor whether segments have been read or written.

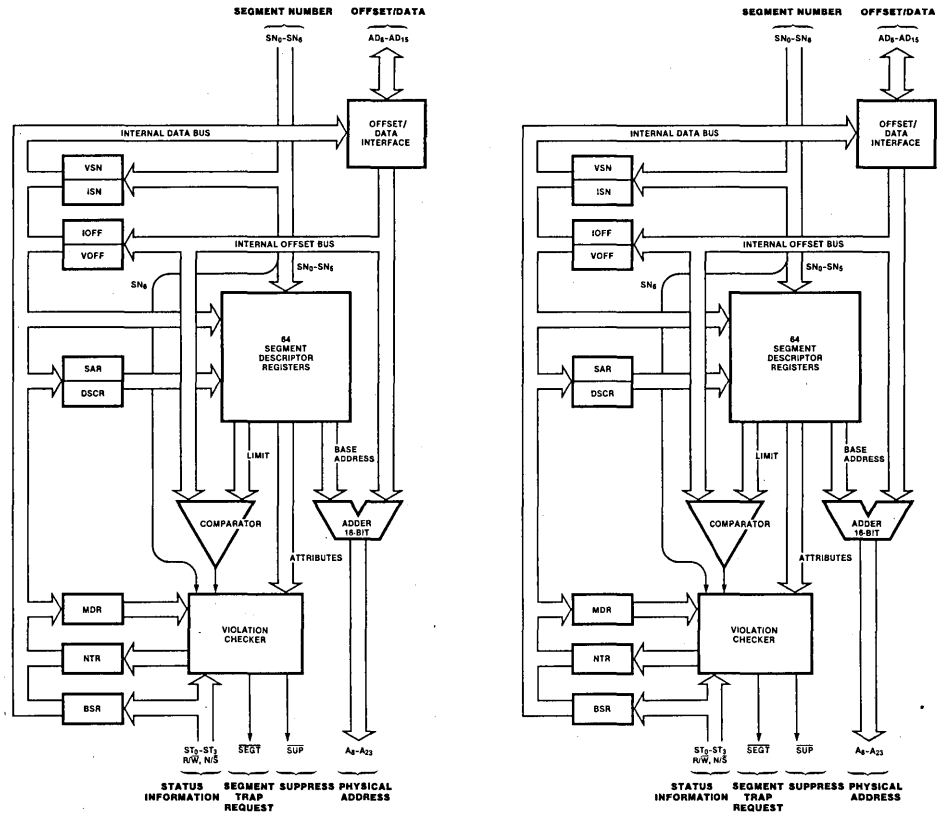


Figure 3. The shaded areas in these block diagrams illustrate the resources used in the two modes of MMU operation. In the Address Translation Mode shown on the left, addresses are translated automatically. In the Command Mode shown on the right, specific registers are accessed using Special I/O commands.

Segmented Addressing

A segmented addressing space—compared with linear addressing—is closer to the way a programmer uses memory because each procedure and data set can reside in its own segment.

The 8M byte Z8001 addressing spaces are divided into 128 relocatable segments of up to 64K bytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any byte relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately.

The MMU divides the physical memory into 256-byte blocks. Segments consist of physically contiguous blocks. Certain segments may be designated so that writes into the last block generate a warning trap. If such a segment is used as a stack, this warning can be used to increase the segment size and prevent a stack overflow error.

The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical addresses*. The MMU takes the logical addresses and transforms them into the *physical addresses* required for accessing the memory (Figure 4). This address transformation process is called *relocation.*

The relocation process is transparent to user software. A translation table in the MMU associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit logical address offset is added to the physical base address to obtain the actual physical memory location. Because a base address always has a low byte equal to zero, only the high-order 16 bits are stored in the MMU and used in the addition. Thus the low-order byte of the physical memory location is the same as the low-order byte of the logical address offset. This low-order byte therefore bypasses the MMU, thus reducing the number of pins required.

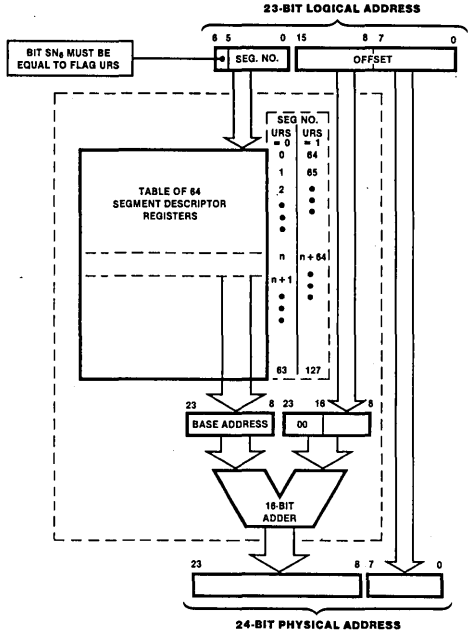


Figure 4. Logical-to-Physical Address Translation

Memory Protection

Each memory segment is assigned several attributes that are used to provide memory access protection. A memory request from the Z8001 CPU is accompanied by status information that indicates the attributes of the memory request. The MMU compares the memory request attributes with the segment attributes and generates a Trap Request whenever it detects an attribute violation. Trap Request informs the Z8001 CPU and the system control program of the violation so that appropriate action can be taken to recover. The MMU also generates the Suppress signal $\overline{SUP}$ in the event of an access violation. Suppress can be used by a memory system to inhibit stores into the memory and thus protect the contents of the memory from erroneous changes.

Five attributes can be associated with each segment. When an attempted access violates any one of the attributes associated with a segment, a Trap Request and a Suppress signal are generated by the MMU. These attributes are read only, execute only, system access only, inhibit CPU accesses and inhibit DMA accesses.

Segments are specified by a base address and a range of legal offsets to this base address. On each access to a segment, the offset is checked against this range to insure that the access falls within the allowed range. If an access that lies outside the segment is attempted, Trap Request and Suppress are generated.

Normally the legal range of offsets within a segment is from 0 to 256N + 255 bytes, where $0 \leq N \leq 255$. However, a segment may be specified so that legal offsets range from 256N to 65,535 bytes, where $0 \leq N \leq 255$. The later type of segment is useful for stacks since the Z8000 stack manipulation instructions cause stacks to grow toward lower memory locations. Thus when a stack grows to the limit of its allocated segment, additional memory can be allocated on the correct end of the segment. As an aid in maintaining stacks, the MMU detects when a write is performed to the lowest allocated 256 bytes of these segments and generates a Trap Request. No Suppress signal is generated so the write is allowed to proceed. This write warning can then be used to indicate that more memory should be allocated to the segment.

MMU Register Organization

The MMU contains three types of registers: Segment Descriptor, Control and Status. A set of 64 Segment Descriptor Registers supplies the information needed to map logical memory addresses to physical memory locations. The segment number of a logical address determines which Segment Descriptor Register is used in address translation. Each Descriptor Register also contains the necessary information for checking that the segment location referenced is within the bounds of the segment and that the type of reference is permitted. It also indicates whether the segment has been read or written.

In addition to the Segment Descriptor Registers, the Z8010 MMU contains three 8-bit control registers for programming the device and six 8-bit status registers that record information in the event of an access violation.

Segment Descriptor Registers. Each of the 64 Descriptor Registers contains a 16-bit base address field, an 8-bit limit field and an 8-bit attribute field (Figure 5). The base address field is subdivided into high- and low-order bytes that are loaded one byte at a time when the descriptor is initialized. The limit field contains a value N that indicates N + 1 blocks of 256 bytes have been allocated to the segment.*

The attribute field contains eight flags (Figure 6). Five are related to protecting the segment against certain types of access, one indicates the special structure of the segment, and two encode the types of accesses that have been made to the segment. A flag is set when its value is 1. The following brief descriptions indicate how these flags are used.

Read-Only (RD). When this flag is set, the segment is read only and is protected against any write access.

System-Only (SYS). When this flag is set, the segment can be accessed only in System mode, and is protected against any access in Normal mode.

CPU-Inhibit (CPUI). When this flag is set, the segment is not accessible to the currently executing process, and is protected against any memory access by the CPU. The segment is, however, accessable under DMA.

Execute-Only (EXC). When this flag is set, the segment can be accessed only during an instruction fetch or access by the relative addressing mode cycle, and thus is protected against any access during other cycles.

DMA-Inhibit (DMAI). When this flag is set, the segment can be accessed only by the CPU, and thus is protected against any access under DMA.

Direction and Warning (DIRW). When this flag is set, the segment memory locations are considered to be organized in descending order and each write to the segment is checked for access to the last 256-byte block. Such an access generates a trap to warn of potential segment overflow, but no Suppress signal is generated.

Changed (CHG). When this flag is set, the segment has been changed (written). This bit is set automatically during any write access to this segment if the write access does not cause any violation.

Referenced (REF). When this flag is set, the segment has been referenced (either read or written). This bit is set automatically during any access to the segment if the access does not cause a violation.

*In the stack mode, segment size is 64K-256N.

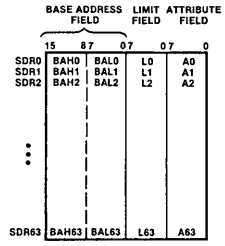


Figure 5. Segment Descriptor Registers

Figure 6. Attribute Field in Segment Descriptor Register

Control Registers. The three user-accessible 8-bit control registers in the MMU direct the functioning of the MMU (Figure 7). The Mode Register provides a sophisticated method for selectively enabling MMUs in multiple-MMU configurations. The Segment Address Register (SAR) selects a particular Segment Descriptor Register to be accessed during a control operation. The Descriptor Selection Counter Register points to a byte within the Segment Descriptor Register to be accessed during a control operation.

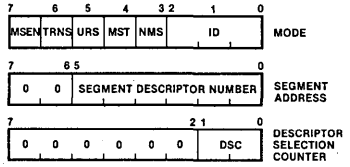


Figure 7. Control Registers

The Mode Register contains a 3-bit identification field (ID) that distinguishes among eight enabled MMUs in a multiple-MMU configuration. This field is used during the segment trap acknowledge sequence (refer to the section on Segment Trap and Acknowledge). In addition, the Mode Register contains five flags.

Multiple Segment Table (MST). This flag indicates whether multiple segment tables are present in the hardware configuration. When this flag is set, more than one table is present and the N/$\overline{S}$ line must be used to determine whether the MMU contains the appropriate table.

Normal Mode Select (NMS). This flag indicates whether the MMU is to translate addresses when the N/$\overline{S}$ line is High or Low. If the MST flag is set, the N/$\overline{S}$ line must match the NMS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

MMU Register Organization (Continued)

Upper Range Select (URS). This flag is used to indicate whether the MMU contains the lower-numbered segment descriptors or the higher-numbered segment descriptors. The most significant bit of the segment number must match the URS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

Translate (TRNS). This flag indicates whether the MMU is to translate logical program addresses to physical memory locations or is to pass the logical addresses unchanged to the memory and without protection checking. In the non-translation mode, the most significant byte of the output is the 7-bit segment number and the most significant bit is 0. When this flag is set, the MMU performs address translation and attribute checking.

Master Enable (MSEN). This flag enables or disables the MMU from performing its address translation and memory protection functions. When this flag is set, the MMU performs these tasks; when the flag is clear the Address lines of the MMU remain 3-stated.

The Segment Address Register (SAR) points to one of the 64 segment descriptors. Control commands to the MMU that access segment descriptors implicitly use this pointer to select one of the descriptors. This register has an auto-incrementing capability so that multiple descriptors can be accessed in a block read/write fashion.

The Descriptor Selection Counter Register holds a 2-bit counter that indicates which byte in the descriptor is being accessed during the reading or writing operation. A value of zero in this counter indicates the high-order byte of the base address field is to be accessed, one indicates the low-order byte of the base address, two indicates the limit field and three indicates the attribute field.

Status Registers. Six 8-bit registers contain information useful in recovering from memory access violations (Figure 8). The Violation Type Register describes the conditions that generated the trap. The Violation Segment Number and Violation Offset Registers record the most-significant 15 bits of the logical address that causes a trap. The Instruction Segment Number and Offset Registers record the most-significant 15 bits of the logical address of the last instruction fetched before the first accessing violation. These two registers can be used in conjunction with external circuitry that records the low-order offset byte. At the time of the addressing violation, the Bus Cycle Status Register records the bus cycle status (status code, read/write mode and normal/system mode).

The MMU generates a Trap Request for two general reasons: either it detects an access violation, such as an attempt to write into a read-only segment, or it detects a warning condition, which is a write into the lowest 256 bytes of a segment with the DIRW flag set. When a violation or warning condition is detected, the MMU generates a Trap Request and automatically sets the appropriate flags. The eight flags in the Violation Type Register describe the cause of a trap.

Read-Only Violation (RDV). Set when the CPU attempts to access a read-only segment and the R/$\overline{W}$ line is Low.

System Violation (SYSV). Set when the CPU accesses a system-only segment and the $\overline{N}$/S line is High.

CPU-Inhibit Violation (CPUIV). Set when the CPU attempts to access a segment with the CPU-inhibit flag set.

Execute-Only Violation (EXCV). Set when the CPU attempts to access an execute-only segment in other than an instruction fetch or load relative instructions cycle.

Segment Length Violation (SLV). Set when an offset falls outside of the legal range of a segment.

Primary Write Warning (PWW). Set when an access is made to the lowest 256 bytes of a segment with the DIRW flag set.

Secondary Write Warning (SWW). Set when the CPU pushes data into the last 256 bytes of the system stack and EXCV, CPUIV, SLV, SYSV, RDV or PWW is set. Once this flag is set, subsequent write warnings for accessing the system stack do not generate a Segment Trap request.

Fatal Condition (FATL). Set when any other flag in the Violation Type Register is set and either a violation is detected or a write warning condition occurs in Normal mode. This flag is not set during a stack push in System mode that results in a warning condition. This flag indicates a memory access error has occurred in the trap processing routine. Once set, no Trap Request signals are generated on subsequent violations. However, Suppress signals are generated on this and subsequent CPU violations until the FATL flag has been reset.

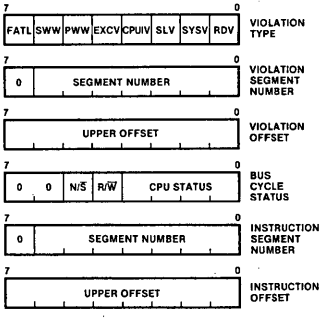


Figure 8. Status Registers

Segment Trap and Acknowledge

The Z8010 MMU generates a Segment Trap when it detects an access violation or a write warning condition. In the case of an access violation, the MMU also activates Suppress, which can be used to inhibit memory writes and to flag special data to be returned on a read access. Segment Trap remains Low until a Trap Acknowledge signal is received. If a CPU-generated violation occurs, Suppress is asserted for that cycle and all subsequent CPU instruction execution cycles until the end of the instruction. Intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only—no Segment Trap Requests are ever generated during DMA cycles.

Segment traps to the Z8001/3 CPU are handled similarly to other types of interrupts. To service a segment trap, the CPU issues a segment trap acknowledge cycle. The acknowledge cycle is always preceded by an instruction fetch cycle that is ignored (the MMU has been designed so that this dummy cycle is ignored). During the acknowledge cycle all enabled MMUs use the Address/Data lines to indicate their status. An MMU that has generated a Segment Trap Request outputs a 1 on the A/D line associated with the number in its ID field; an MMU that has not generated a segment trap request outputs a 0 on its associated A/D line. A/D lines for which no MMU is associated remain 3-stated. During a segment trap acknowledge cycle, an MMU uses A/D line 8+i if its ID field is i.

Following the acknowledge cycle the CPU automatically pushes the Program Status onto the system stack and loads another Program Status from the Program Status Area. The Segment Trap line is reset during the segment trap acknowledge cycle. Suppress is not generated during the stack push. If the store creates a write warning condition, a Segment Trap Request is generated and is serviced at the end of the Program Status swap. The SWW flag is also set. Servicing this second Segment Trap Request also creates a write warning condition, but because the SWW flag is set, no Segment Trap Request is generated. If a violation rather than a write warning occurs during the Program Status swap, the FATL flag is set rather than the SWW flag. Subsequent violations cause Suppress to be asserted but not Segment Trap Request. Without the SWW and FATL flags, trap processing routines that generate memory violations would repeatedly be interrupted and called to process the trap they created.

The CPU routine to process a trap request should first check the FATL flag to determine if a fatal system error has occurred. If not, the SWW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the Violation Type Register reset.

Virtual Memory

Several features of the MMU can be used in conjunction with external circuitry to support virtual memory for the Z8001. Segment Trap Request can be used to signal the CPU in the event that a segment is not in primary memory. The CPU-Inhibit Flag can be used to indicate whether a segment is in the memory or in secondary storage. The Changed and Altered Flags in the attribute field for each segment can aid in implementing efficient segment management policies. The Status Registers can be used in recovering from virtual memory access faults.

Multiple MMUs

MMU architecture directly supports two methods for multiple MMU configurations. The first approach extends single-MMU capability for handling 64 segments to a dual-MMU configuration that manages the 128 different segments the Z8001 can address. This scheme uses the URS flag in the Mode Register in connection with the high-order bit of the segment number (SN_6).

The second approach uses several MMUs to implement multiple translation tables. Multiple tables can be used to reduce the time required to switch tasks by assigning separate tables to each task. Multiple translation tables for multi-task environments can use the Master Enable Flag to enable the appropriate MMUs through software. Multiple translation tables may also be used to extend the physical memory size beyond 16 megabytes by separating system from normal memory and/or program from data memory. The MST and NMS flags in the Mode Register can be used in conjunction with the $N/\overline{S}$ line to select the MMU that contains the appropriate table. Special external circuitry that monitors the CPU Status lines can manipulate the MMU $N/\overline{S}$ line to perform this selection.

DMA Operation

Direct memory access operations may occur between Z8001 instruction cycles and can be handled through the MMU. The MMU permits DMA in either the System or Normal mode of operation. For each memory access, the segment attributes are checked and if a violation is detected, Suppress is activated. Unlike a CPU violation that automatically causes Suppress signals to be generated on subsequent memory accesses until the next instruction, DMA violations generate a Suppress only on a per memory access basis.

The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. No Segment Trap Request is ever generated during DMA, hence warning conditions are not signaled. Trap Requests are not issued because the CPU cannot acknowledge such a request.

At the start of a DMA cycle, DMASYNC must go Low for at least two clock cycles, indicating to the MMU the beginning of a DMA cycle. A Low DMASYNC inhibits the MMU from using an indeterminate segment number on lines SN_0-SN_6. When the DMA logical memory address is valid, the DMASYNC line must be High before a rising edge of Clock and the MMU then performs its address translation and access protection functions. Upon the release of the bus at the termination of the DMA cycle the DMASYNC line must again be High. After two clock cycles of DMASYNC High, the MMU assumes that the CPU has control of the bus and that subsequent memory references are CPU accesses. The first instruction fetch occurs at least two cycles after the CPU regains control of the bus. During CPU cycles, DMASYNC should always be High.

MMU Commands

The various registers in the MMU can be read and written using Z8001 CPU special I/O commands. These commands have machine cycles that cause the Status lines to indicate an SIO operation is in progress. During these machine cycles the MMU enters command mode. In this mode, the rising edge of the Address Strobe indicates a command is present on the AD_8-AD_{15}. If Chip Select is asserted and if this command indicates that data is to be written into one of the MMU registers, the data is read from AD_8-AD_{15} while Data Strobe is Low. If the command indicates that data is to be read from one of the MMU registers, the data is placed on AD_8-AD_{15} while Data Strobe is Low.

There are ten commands that read or write various fields in the Segment Descriptor Register. The status of the Read/Write line indicates whether the command is a read or a write.

The auto-incrementing feature of the Segment Address Register (SAR) can be used to block load segment descriptors using the repeat forms of the Special I/O instructions. The SAR is autoincremented at the end of the field. In accessing the base field, first the high-order byte is selected and then the low-order byte. The command accessing the entire Descriptor Register references the fields in the order of base address, limit and attribute.

Opcode (Hex)	Instruction
08	Read/Write Base Field
09	Read/Write Limit Field
0A	Read/Write Attribute Field
0B	Read/Write Descriptor (all fields)
0C	Read/Write Base Field; Increment SAR
0D	Read/Write Limit Field; Increment SAR
0E	Read/Write Attribute Field; Increment SAR
0F	Read/Write Descriptor; Increment SAR
15	Set All CPU-Inhibit Attribute Flags
16	Set All DMA-Inhibit Attribute Flags

Three commands are used to read and write the control registers.

Opcode (Hex)	Instruction
00	Read/Write Mode Register
01	Read/Write Segment Address Register
20	Read/Write Descriptor Selector Counter Register

The Status Registers are read-only registers, although the Violation Type Register (VTR) can be reset. Nine instructions access these registers.

Opcode (Hex)	Instruction
02	Read Violation Type Register
03	Read Violation Segment Number Register
04	Read Violation Offset (High-byte) Register
05	Read Bus Status Register
06	Read Instruction Segment Number Register
07	Read Instruction Offset (High-byte) Register
11	Reset Violation Type Register
13	Reset SWW Flag in VTR
14	Reset FATL Flag in VTR

MMU Timing

The Z8010 translates addresses and checks for access violations by stepping through sequences of basic clock cycles corresponding to the cycle structure of the Z8001 CPU. The following timing diagrams show the relative timing relationships of MMU signals during the basic operations of memory read/write and MMU control commands. For exact timing information, refer to the composite timing diagram.

Memory Read and Write. Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 inputs. During a memory read cycle (Figure 9) the 7-bit segment number is input on SN_0-SN_6 one clock period earlier than the address offset; a High on DMASYNC during T_3 indicates that the segment offset data is valid. The most significant eight bits of the address offset are placed on the AD_0-AD_{15} inputs early in the

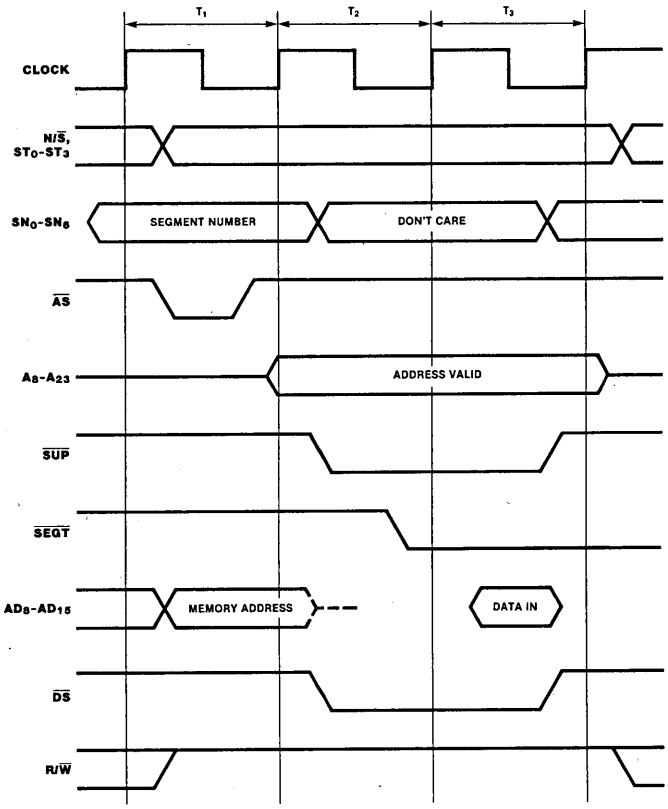


Figure 9. Memory Read Timing

MMU Timing (Continued)

first clock period. Valid address offset data is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The most significant 16-bits of the address (physical memory location) remain valid until the end of T_3. Segment Trap Request and Suppress are asserted in T_2. Segment Trap Request remains Low until Segment Trap Acknowledge is received. Suppress is asserted during the current machine cycle and terminates during T_3. Suppress is repeatedly asserted during CPU instruction execution cycles until the current instruction has terminated.

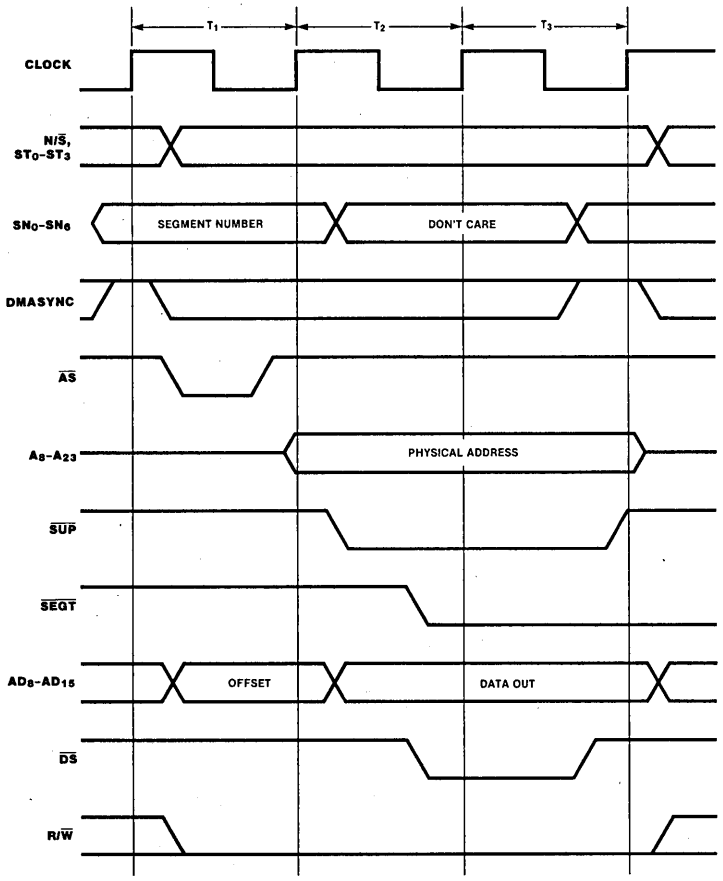


Figure 10. Memory Write Timing

MMU Timing (Continued)

MMU Command Cycle. During the command cycle of the MMU (Figure 11), commands are placed on the Address/Data lines during T_1. The Status lines indicate that a Special I/O instruction is in progress, and the Chip Select line enables the appropriate MMU for that command. Data to be written to a register in the MMU must be valid on the Address/Data lines late in T_2. Data read from the MMU is placed on the Address/Data lines late in the T_{WA} cycle.

Input/Output and Refresh. Input/Output and Refresh operations are indicated by the status lines ST_0-ST_3. During these operations, the MMU refrains from any address translation or protection checking. The address lines A_8-A_{23} remain 3-stated.

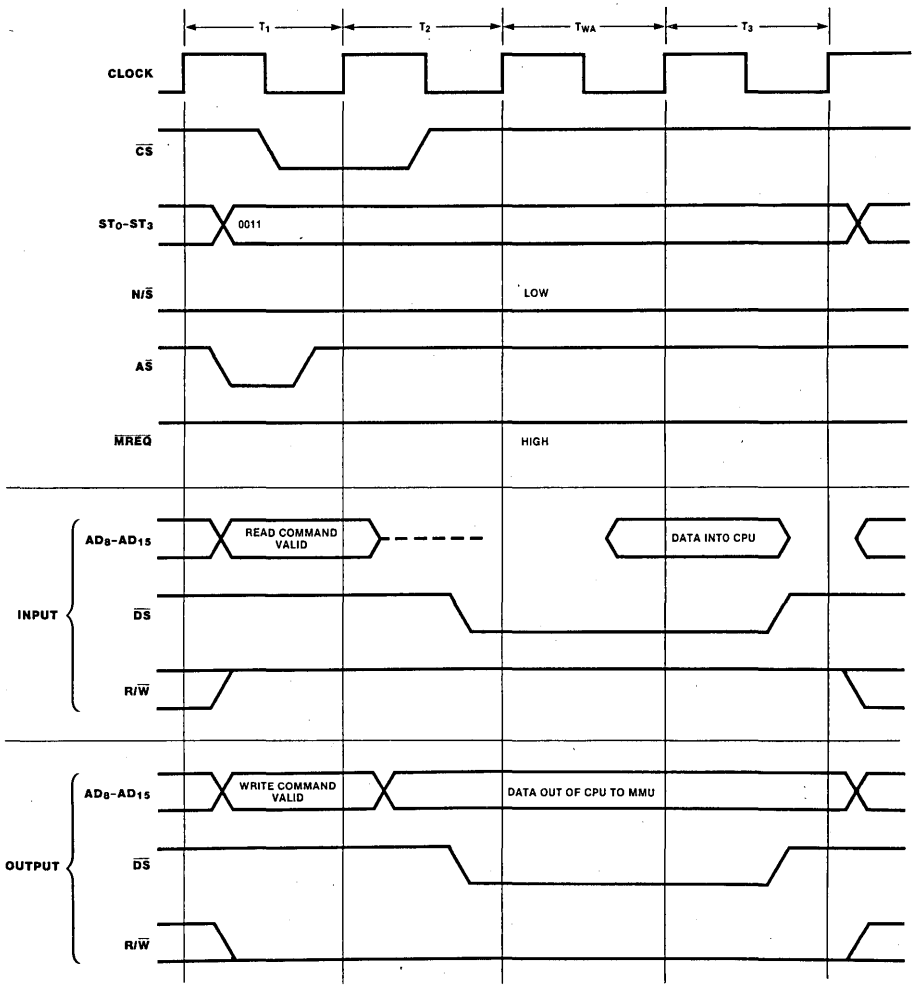


Figure 11. I/O Command Timing

MMU Timing (Continued)

Reset. The MMU can be reset by either hard ware or software mechanisms. A hardware reset occurs on the falling edge of the Reset signal; a software reset is performed by a Z8000 Special I/O command. A hardware reset clears the Mode Register, Violation Type Register and Descriptor Selection Counter. If the Chip Select line is Low, the Master Enable Flag in the Mode Register is set to 1. All other registers are undefined. After reset, the $\underline{AD_8}$-$\underline{AD_{15}}$ and A_8-A_{23} lines are 3-stated. The $\overline{SUP}$and $\overline{SEGT}$ open-drain outputs are not driven. If the Master Enable flag is not set during reset, the MMU does not respond to subsequent addresses on its A/D lines. To enable an MMU after a hardware reset, an MMU command must be used in conjunction with the Chip Select line.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Violation Type Register and returns the MMU to its initial state (as if no violations or warnings had occurred). Note that the hardware and software resets have different effects.

Segment Trap and Acknowledge. The Z8010 MMU generates a segment trap whenever it detects an access violation or a write into the lowest block of a segment with the DIRW flag set. In the case of an access violation, the MMU also activates Suppress. This Suppress signal can be used to inhibit memory writes. The Segment Trap remains Low until a Trap Acknowledge signal is received. If a violation occurs, Suppress is asserted for that cycle and all subsequent CPU cycles until the end of the instruction; intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only, but no Trap Request is generated.

When the MMU issues a Segment Trap Request it awaits a Segment Trap Acknowledge. Subsequent violations occurring before the Trap Acknowledge is received are still detected and handled appropriately. During the Segment Trap Acknowledge cycle, the MMU drives one of its Address/Data lines High; the particular line selected is a function of the identification field of the mode register. After the Segment Trap has been acknowledged by the Z8001 CPU, the Violation Status Register should be read via the Special I/O commands in order to determine the cause of the trap. The Trap Type Register should also be reset so that subsequent traps will be recorded correctly.

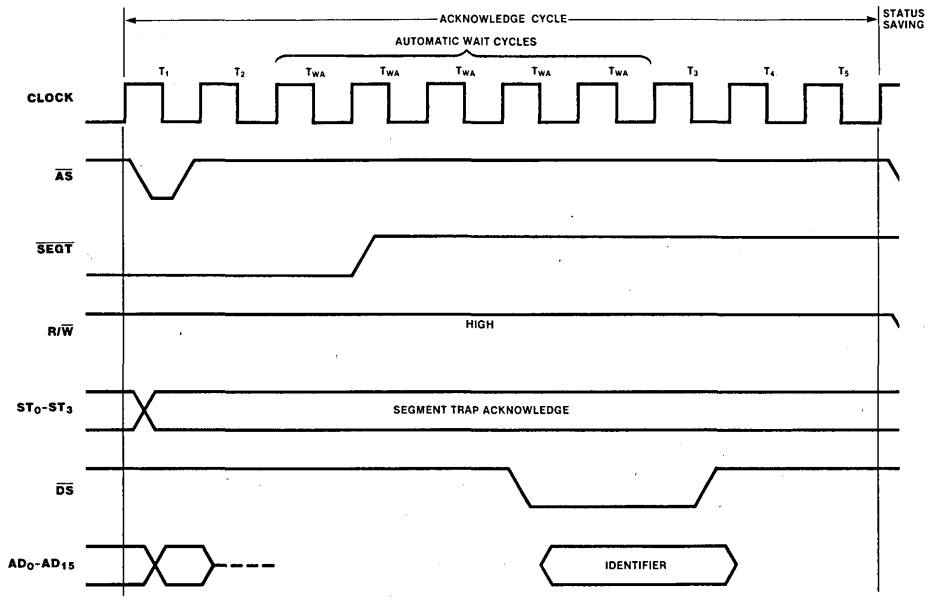


Figure 12. Segment Trap and Acknowledge Timing

Pin Description

A_8-A_{23}. *Address Bus* (outputs, active High, 3-state). These address lines are the 16 most-significant bits of the physical memory location.

AD_8-AD_{15}. *Address/Data Bus* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for commands and for logical addresses intended for translation.

$\overline{AS}$. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ indicates that AD_0-AD_{15}, ST_0-ST_3, R/$\overline{W}$ and N/$\overline{S}$ are valid.

CLK. *System Clock* (input). CLK is the 5 V single-phase time-base input used for both the CPU and MMU.

$\overline{CS}$. *Chip Select* (input, active Low). This line selects an MMU for a control command.

DMASYNC. *DMA/Segment Number Synchronization Strobe* (input, active High). A Low on this line indicates that the segment number lines are 3-state; a High indicates that the segment number is valid. It must always be High during CPU cycles. If a DMA device does not use the MMU for address translation, the $\overline{BUSACK}$ signal from the CPU may be used as an input to DMASYNC.

$\overline{DS}$. *Data Strobe* (input, active Low). This line provides timing for the data transfer between the MMU and the Z8001 CPU.

N/$\overline{S}$. *Normal/System Mode* (input, Low = System Mode). N/$\overline{S}$ indicates the Z8001 CPU or Z8016 DMA is in the Normal or System Mode. The signal can also be used to switch between MMUs during different phases of an instruction.

Reserved. Do not connect.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the MMU.

R/$\overline{W}$. *Read/Write* (input, Low = write). R/$\overline{W}$ indicates the Z8001 CPU or Z8016 DTC is reading from or writing to memory or the MMU.

$\overline{SEGT}$. *Segment Trap Request* (output, active Low, open drain). The MMU interrupts the Z8001 CPU with a Low on this line when the MMU detects an access violation or write warning.

SN_0-SN_6. *Segment Number* (inputs, active High). The SN_0-SN_5 lines are used to address one of 64 segments in the MMU; SN_6 is used to selectively enable the MMU.

ST_0-ST_3. *Status* (inputs, active High). These lines specify the Z8001 CPU status.

ST_3-ST_0	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference (e.g., to an MMU)
0 1 0 0	Segment trap acknowledge
0 1 0 1	Nonmaskable interrupt acknowledge
0 1 1 0	Nonvectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (EPU)
1 0 1 1	Stack memory request (EPU)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	Extension processor transfer

$\overline{SUP}$. *Suppress* (output, active Low, open drain). This signal is asserted during the current bus cycle when any access violation except write warning occurs.

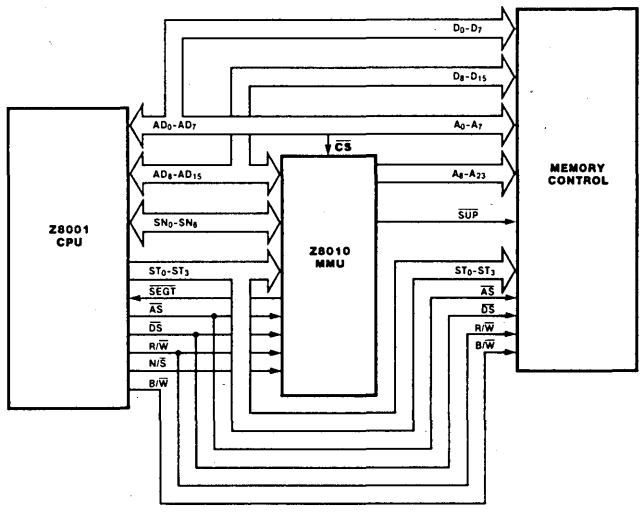
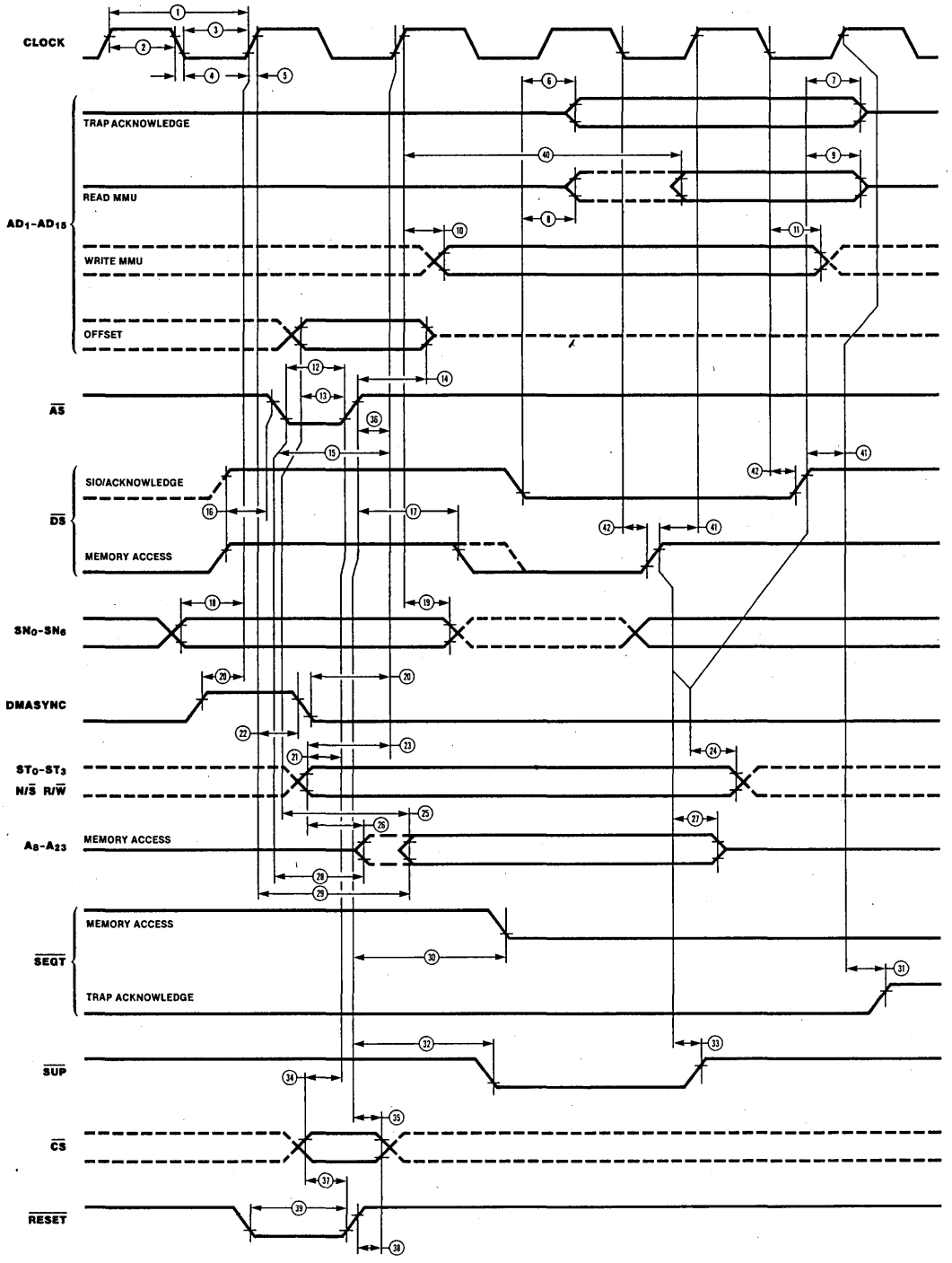


Figure 13. The MMU in a Z8001 System


CLOCK
TRAP ACKNOWLEDGE
READ MMU
AD1-AD15
WRITE MMU
OFFSET
AS
SIO/ACKNOWLEDGE
DS
MEMORY ACCESS
SN0-SN8
DMASYNC
ST0-ST3
N/S R/W
A8-A23
MEMORY ACCESS
SEQT
MEMORY ACCESS
TRAP ACKNOWLEDGE
SUP
CS
RESET

AC Characteristics

No.	Symbol	Parameter	Z8010 6 MHz Min	Z8010 6 MHz Max	Z8010 10 MHz Min	Z8010 10 MHz Max	Notes*
1	TcC	Clock Cycle Time	165		100		
2	TwCh	Clock Width (High)	70		40		
3	TwCl	Clock Width (Low)	70		40		
4	TfC	Clock Fall Time		10		10	
5	TrC	Clock Rise Time		15		10	
6	TdDSA(RDv)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		80		60	1
7	TdDSA(RDf)	$\overline{DS}$ ↑ (Acknowledge) to Read Data Float Delay		60		45	1
8	TdDSR(RDv)	$\overline{DS}$ ↓ (Read) to AD Output Driven Delay		80		60	1
9	TdDSR(RDf)	$\overline{DS}$ ↑ (Read) to Read Data Float Delay	60	60		45	1
10	TdC(WDv)	CLK ↑ to Write Data Valid Delay		80		50	
11	ThC(WDn)	CLK ↓ to Write Data Not Valid Hold Time	20		10		
12	TwAS	Address Strobe Width	50		30		
13	TsOFF(AS)	Offset Valid to $\overline{AS}$ ↑ Setup Time	35		20		
14	ThAS(OFFn)	$\overline{AS}$ ↑ to Offset Not Valid Hold Time	40		20		
15	TdAS(C)	AS ↓ to CLK ↑ Delay		90		50	
16	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	30		15		
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	40		30		
18	TsSN(C)	SN Data Valid to CLK ↑ Setup Time	40		30		
19	ThC(SNn)	CLK ↑ to SN Data not Valid Hold Time	0		0		
20	TdDMAS(C)	DMASYNC Valid to CLK ↑ Delay		80		60	
21	TdSTNR(AS)	Status (ST_0-ST_3, N/$\overline{S}$, R/$\overline{W}$) Valid to $\overline{AS}$ ↑ Delay	30		10		
22	TdC(DMA)	CLK ↑ to DMASYNC ↓ Delay	15		10		
23	TdST(C)	Status (ST_0-ST_3) Valid to CLK ↑ Delay	60		30		
24	TdDS(STn)	$\overline{DS}$ ↑ to Status Not Valid Delay	0		0		
25	TdOFF(Av)	Offset Valid to Address Output Valid Delay		90		60	1
26	TdST(Ad)	Status Valid to Address Output Driven Delay		75		45	1
27	TdDS(Af)	$\overline{DS}$ ↑ to Address Output Float Delay		130		100	1
28	TdAS(Ad)	$\overline{AS}$ ↓ to Addres Output Driven Delay		70		40	1
29	TdC(Av)	CLK ↑ to Address Output Valid Delay		155		100	1
30	TdAS(SEGT)	$\overline{AS}$ ↑ to $\overline{SEGT}$ ↓ Delay		100		60	1,2
31	TdC(SEGT)	CLK ↑ to $\overline{SEGT}$ ↑ Delay		200		100	1,2
32	TdAS(SUP)	$\overline{AS}$ ↑ to $\overline{SUP}$ ↓ Delay		90		55	1,2
33	TdDS(SUP	$\overline{DS}$ ↑ to $\overline{SUP}$ ↑ Delay		100		60	1,2
34	TsCS(AS)	Chip Select Input Valid to $\overline{AS}$ ↑ Setup Time	10		10		
35	ThAS(CSn)	$\overline{AS}$ ↑ to Chip Select Input Not Valid Hold Time	40		20		
36	TdAS(C)	$\overline{AS}$ ↑ to CLK ↑ Delay	10		10		
37	TsCS(RST)	Chip Select Input Valid to $\overline{RESET}$ ↑ Setup Time	100		60		
38	ThRST(CSn)	$\overline{RESET}$ ↑ to Chip Select Input Not Valid Hold Time	0		0		
39	TwRST	$\overline{RESET}$ Width (Low)	2TcC		2TcC		
40	TdC(RDv)	CLK ↑ to Read Data Valid Delay	460	300		190	
41	TdDS(C)	$\overline{DS}$ ↑ to CLK ↑ Delay	20		10		
42	TdC(DS)	CLK ↓ to $\overline{DS}$ ↑ Delay	0		0		

NOTES:
1. 50 pf Load.
2. 2.2K Pull-up.

* Units in nanoseconds (ns).

Timing measurements are made at the following voltages:

	High	*Low*
Clock	4.0 V	0.8 V
Output	2.0 V	0.8 V
Input	2.0 V	0.8 V
Float	△V	±0.5 V

Absolute Maximum Ratings

Voltages on all pins with respect to GND ..... −0.3V to +7.0V
Operating Ambient Temperature .....See Ordering Information
Storage Temperature..... −65°C to +150°C

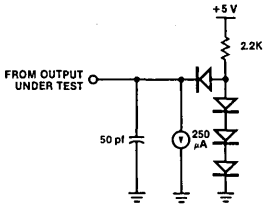
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The DC characteristics and capacitance section below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T_A ≤ +70°C



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{OL}	Output Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		300	mA	

NOTE: The on-chip back-bias voltage generator takes approximately 20 ms to pump the back-bias voltage to -2.5 V after the power has been turned on. The performance of the Z8010 Z-MMU is not guaranteed during this period.

Zilog

Product Specification

Z8016 Z8000® Z-DTC Direct Memory Access Transfer Controller

October 1988

FEATURES

- Memory-to-peripheral transfers up to 2.66M bytes per second at 4 MHz.
- Memory-to-memory transfers up to 1.33M bytes per second at 4 MHz.
- Two fully independent, multi-function channels.
- Masked data pattern matching for Search and Transfer-and-Search operations.
- Funneling option that permits mixing of byte and word data during transfer operations.
- Can operate in logical address space with Zilog Memory Management Units, providing an 8M byte logical addressing range and 16M byte physical addressing range.
- Programmable chaining operation provides automatic loading of control parameters from memory into each channel.
- Software- or hardware-controlled Wait state insertion.
- Z-BUS™ daisy-chain interrupt hierarchy and bus-request structure.

GENERAL DESCRIPTION

The Z8016 DMA Transfer Controller (DTC) is a high performance data transfer device designed to match the power and addressing capability of the Z8000 CPUs. In addition to providing block data transfer capability between memory and peripherals, each of the two DTC channels can perform peripheral-to-peripheral and memory-to-memory transfers. A special Search mode of operation compares data read from memory or peripherals with the contents of a pattern register. A search can be performed concurrently with transfers or as an operation in itself.

In all operations (Search, Transfer, and Transfer-and-Search), the DTC can operate in either Flowthrough or Flyby transfer mode. In the Flowthrough mode, data is stored temporarily within the DTC on its way from source to destination. In this mode transfers can be made between a word-oriented memory and a byte-oriented peripheral through the bidirectional byte/word funneling option. In Flyby mode, data is transferred in a single step (from source to destination), thus providing twice the throughput.

The Z8016 DTC takes full advantage of the Z8000 memory management scheme by interfacing directly to the Z8010 Memory Management Unit (MMU) or the Z8015 Paged Memory Management Unit (PMMU). In this configuration, 8M bytes of logical address range are provided for each CPU address space. Alternatively, the Z8016 DTC can operate independently of an MMU, directly addressing up to 16M bytes of physical address space.

In addition to providing a hardware $\overline{\text{WAIT}}$ input to accommodate different memory or peripheral speeds, the Z8016 DTC allows the user to program the automatic insertion of either zero, one, two, or four Wait states for either source or destination addresses. Alternatively, the $\overline{\text{WAIT}}$ input pin function can be disabled and these software-programmed Wait states used exclusively.

The Z8016 DTC minimizes CPU involvement by allowing each channel to load its control registers from memory automatically when a DMA operation is complete. By loading the address of the next block of control parameters as part of this operation, command chaining is accomplished. The only action required of the CPU is to load the address of the control parameter table into the channel's Chain Address register and then issue a Start Chain command.

In some DMA applications, data is transferred continuously between the same two locations. To service these repetitive DMA operations, base registers are provided on each channel to reinitialize the current source and destination address registers. This re-initialization eliminates the need for reloading registers from memory tables.

The Z8016 DTC is directly Z-BUS compatible, and operates within the Z8000 daisy-chain vectored-priority interrupt scheme. The Demand Interleave operation allows the DTC to surrender the bus to the external system, or to alternate between internal channels. This capability allows for parallel operations between dual channels or between a DTC channel and the CPU.

The DTC can be used to provide a central DMA function for the CPU or to provide dispersed DMA operations in conjunction with a wide variety of Z8000 Family peripheral controllers.

The Z8016 DTC is packaged in a 48-pin DIP and uses a single +5 V power supply.

The Z8016 DTC pin functions and assignments are shown in Figures 1 and 2, respectively.

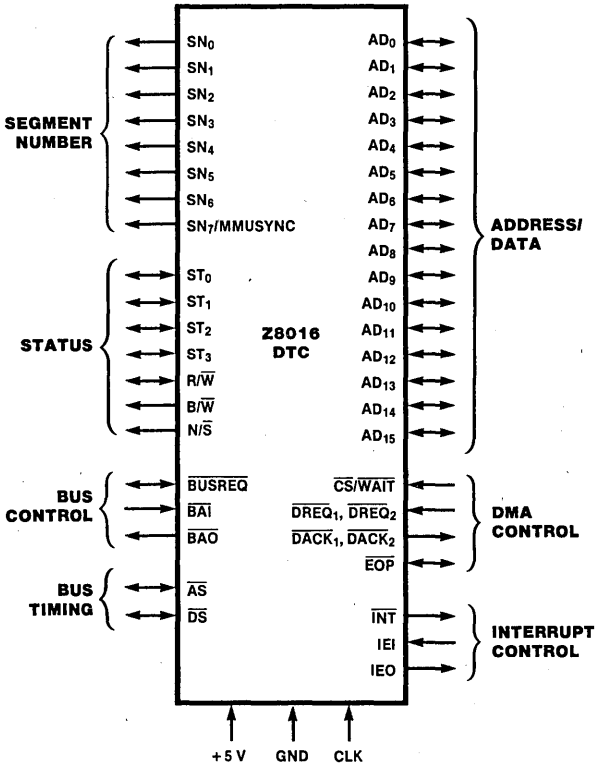


Figure 1. Pin Functions

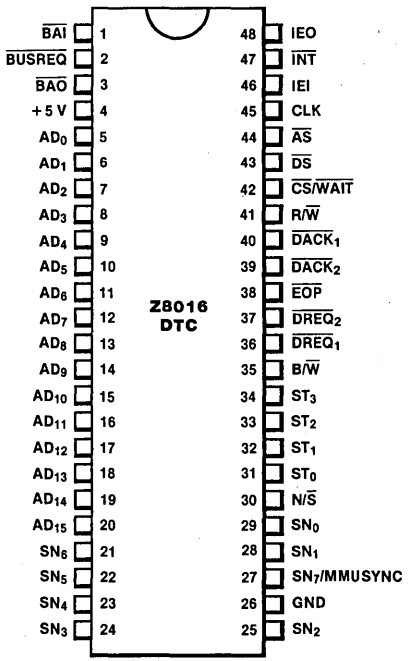


Figure 2. 48-pin Dual-In-Line Package (DIP), Pin Assignments

SIGNAL DESCRIPTIONS

AD_0-AD_{15}. *Address/Data Bus* (bidirectional, active High, 3-state) pins 5-20. These multiplexed Address/Data lines are used for all I/O and memory transactions.

$\overline{AS}$. *Address Strobe* (bidirectional, active Low, 3-state) pin 44. When the DTC is bus master the rising edge of $\overline{AS}$ (while $\overline{DS}$ is High) indicates that addresses are valid. When the DTC is not bus master, the address lines are sampled on the rising edge of $\overline{AS}$. There are no timing requirements between $\overline{AS}$ as an input and the DTC clock, because the Z-BUS does not use a bused clock. If $\overline{AS}$ and $\overline{DS}$ are simultaneously Low, the DTC will be reset.

$\overline{BAI}$. *Bus Acknowledge In* (input, active Low) pin 1. Signals that the bus has been released for DTC control. In multiple-DTC configurations, the $\overline{BAI}$ pin of the highest-priority DTC is normally connected to the Bus Acknowledge pin of the CPU. Each lower-priority DTC has its $\overline{BAI}$ connected to the $\overline{BAO}$ of the next higher-priority DTC.

$\overline{BAO}$. *Bus Acknowledge Out* (output, active Low) pin 3. In a multiple-DMA configuration, this pin signals that no higher-priority DTC has requested the bus. $\overline{BAI}$ and $\overline{BAO}$ form a daisy chain for multiple-DTC priority resolution.

$\overline{\text{BUSREQ}}$. *Bus Request* (bidirectional, active Low, open-drain) pin 2. $\overline{\text{BUSREQ}}$ is used by the DTC to obtain control of the bus from the CPU. Before driving $\overline{\text{BUSREQ}}$ active, the DTC samples this line to ensure that another request is not already being made by another device. Since the DTC internally synchronizes the sampled $\overline{\text{BUSREQ}}$ signal, transitions on $\overline{\text{BUSREQ}}$ can be asynchronous with respect to the DTC clock.

B/$\overline{\text{W}}$. *Byte/Word* (output, 3-state) pin 35. This output indicates the type of data transferred on the Address/Data (A/D) bus. A High on this line indicates a byte (8-bit) transfer and a Low indicates a word (16-bit) transfer. This signal is activated when $\overline{\text{AS}}$ goes Low and remains valid for the duration of the transaction.

CLK. *DTC Clock* (input) pin 45. The Clock signal controls internal operations and the rates of data transfer. It is usually derived from a master system clock or an associated CPU clock. When the DTC is used with an MMU, both must be driven from the same clock signal. While many DTC input signals are asynchronous, transitions for other signals (such as $\overline{\text{WAIT}}$ inputs) must meet setup and hold requirements relative to the DTC clock. (See the timing diagrams for details.)

$\overline{\text{CS}}$/$\overline{\text{WAIT}}$. *Chip Select/Wait* (input, active Low) pin 42. When the DTC is not in control of the system bus, this pin serves as a Chip Select ($\overline{\text{CS}}$) input. A CPU or other external device uses $\overline{\text{CS}}$ to activate the DTC for reading and writing the DTC's internal registers. ($\overline{\text{CS}}$ can be held Low for multiple transfers to and from the DTC, provided that $\overline{\text{AS}}$ and $\overline{\text{DS}}$ are enabled for each transfer.) There are no timing requirements between the $\overline{\text{CS}}$ input and the DTC clock; the $\overline{\text{CS}}$ input timing requirements are only defined relative to $\overline{\text{AS}}$.

When the DTC is in control of the system bus, this pin serves as the $\overline{\text{WAIT}}$ input. Slow memories and peripheral devices can use $\overline{\text{WAIT}}$ to extend $\overline{\text{DS}}$ during bus transfers. Unlike the $\overline{\text{CS}}$ input, transitions on the $\overline{\text{WAIT}}$ input must meet certain timing requirements relative to the DTC clock (see the Active State timing diagram for details). The $\overline{\text{WAIT}}$ function can be disabled using a control bit in the Master Mode register, in which case this input is treated as a Chip Select only and is ignored when the DTC is in control of the system bus.

$\overline{\text{DACK}}_1$, $\overline{\text{DACK}}_2$. *DMA Acknowledge* (output, active Low) pins 39 and 40. There is one DMA Acknowledge line associated with each channel. The $\overline{\text{DACK}}$ lines are programmed in the Channel Mode register to be pulsed, held active, or held inactive during DMA transfers. During Flyby operations the $\overline{\text{DACK}}$ line is used for two purposes. It selects the peripheral involved in the transfer, and it provides timing information on when to access the bus. During flowthrough operations the $\overline{\text{DACK}}$ line can be programmed to be active or inactive during a DMA transfer. $\overline{\text{DACK}}$ is not output during chaining operations.

$\overline{\text{DREQ}}_1$, $\overline{\text{DREQ}}_2$. *DMA Request* (input, active Low) pins 36 and 37. There is a DMA Request line associated with each channel. These lines can make transitions independent of the DTC clock. They are used by external logic to initiate and control DMA operations performed by the DTC.

$\overline{\text{DS}}$. *Data Strobe* (bidirectional, active Low, 3-state) pin 43. A Low on this signal while $\overline{\text{AS}}$ is High indicates that the A/D bus is being used to transfer data. When the CPU is bus master and is transferring information to or from the DTC, $\overline{\text{DS}}$ is a timing input used by the DTC to move data to or from the A/D bus.

$\overline{\text{EOP}}$. *End of Process* (bidirectional, active Low, open-drain, asynchronous) pin 38. This line is output when a Terminal Count (TC) or Match Condition (MC) termination occurs (see Termination section). An external source can terminate a DMA operation in progress by driving $\overline{\text{EOP}}$ Low. $\overline{\text{EOP}}$ always applies to the active channel; if no channel is active, $\overline{\text{EOP}}$ is ignored. The Suppress output of the MMU can be connected to $\overline{\text{EOP}}$ to terminate DMA accesses that violate the MMU protection settings. To provide full access protection, an external $\overline{\text{EOP}}$ is accepted even during chaining.

IEI. *Interrupt Enable In* (input, active High) pin 46. IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher-priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High) pin 48. IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting DTC. IEO is connected to the next lower-priority device's IEI input and thus inhibits interrupts from lower-priority devices.

$\overline{\text{INT}}$. *Interrupt Request* (output, open-drain, active Low) pin 47. This signal is pulled Low when the DTC requests an interrupt.

N/$\overline{\text{S}}$. *Normal/System* (output, 3-state) pin 30. The N/$\overline{\text{S}}$ signal is activated when the DTC is bus master. The N/$\overline{\text{S}}$ signal indicates which memory space is being accessed by going High for normal memory and Low for system memory.

R/$\overline{\text{W}}$. *Read/Write* (bidirectional, 3-state, Low = write) pin 41. When the DTC is not bus master, R/$\overline{\text{W}}$ is a status input used to indicate whether data is being read from (High) or written to (Low) the DTC. When the DTC is bus master, R/$\overline{\text{W}}$ is an output used to indicate whether the DTC is reading or writing the addressed location. During Flyby DMA operations, the "Flyby peripheral" (Figure 3) inverts the R/$\overline{\text{W}}$ signal to determine whether it must read or write.

SN_0-SN_6. *Segment Number* (output, 3-state) pins 21-25 and 28-29. In logical address configuration, these lines provide the segment number field of a 23-bit segmented address. The SN_0–SN_6 I/O address information can be used to increase the DTC's logical I/O address space beyond that of the CPU. In physical address configuration, these lines provide bits 23 through 17 of a 24-bit linear address. The 24th bit (MSB) is output on SN_7/MMU Sync.

SN_7 or MMU Sync. *Segment Number 7 or MMU Sync* (output, 3-state) pin 27. In a logical address space configuration (with MMU), this line outputs an active High pulse prior to each machine cycle. The MMU uses this signal to synchronize access to its translation table and to differentiate between CPU and DTC control. The MMU ignores MMUSYNC if the status lines (ST_0–ST_3) indicate that an I/O transaction is being performed. This output is Low when the DTC is not bus master and the MM1 bit in the Master Mode register is set.

In a physical address space configuration (without MMU), this line outputs SN_7, which becomes the 24th address bit in a linear address space. The 24-bit linear address configuration allows the DTC to access 16M bytes of memory. This pin floats to the high impedance state when the DTC is not bus master and the MM1 bit is cleared.

ST_0-ST_3. *Status* (bidirectional, 3-state) pins 31-34. When the DTC is bus master, these lines are outputs indicating the type of memory or I/O transaction being performed. When the DTC is not bus master, the status lines are inputs used to detect Interrupt and Segment Trap Acknowledge cycles (Table 1).

Table 1. Status Codes

ST_3	ST_2	ST_1	ST_0	Transaction/Operation	Status Code Generated/Decoded
0	0	0	0	Internal Operation	
0	0	0	1	Memory Refresh	
0	0	1	0	I/O Transaction	Generated
0	0	1	1	Special I/O Transaction	Generated
0	1	0	0	Segment Trap Acknowledge	Decoded
0	1	0	1	Nonmaskable Interrupt Acknowledge	Decoded
0	1	1	0	Nonvectored Interrupt Acknowledge	Decoded
0	1	1	1	Vectored Interrupt Acknowledge	Decoded
1	0	0	0	Memory Transaction for Data/DTC Chaining	Generated
1	0	0	1	Memory Transaction for Stack	Generated
1	0	1	0	Reserved	
1	0	1	1	Reserved	
1	1	0	0	Memory Transaction for Program Fetch (Subsequent Word)	Generated
1	1	0	1	Memory Transaction for Program Fetch (First Word)	
1	1	1	0	Reserved	
1	1	1	1	Reserved	

FUNCTIONAL DESCRIPTION

Channel Initialization

The Z8016 DTC operates with a minimum of interaction with the host CPU. Each channel's operation is determined by the settings of its own set of control registers. Each channel is initialized when the DTC loads its control parameters from memory into its control registers during the chaining operation. To initiate the chaining operation, the CPU is required to program the Master Mode register and each channel's Chain Address register. Then each channel's control registers are automatically loaded by the DTC with control parameters stored in a chain control table in memory, located at the address pointed to by that channel's Chain Address register. Once the channel registers are loaded, the DTC is ready to perform DMA operations.

Initiating DMA Operations. DMA operations can be initiated in three ways:

- Software Request. The CPU can issue Software Request commands to start DMA operations on a specific channel. This channel must then request control of the bus and perform transfers.
- Hardware Request. DMA operations can be started by forcing a channel's $\overline{DREQ}$ input Low, as described in the Transfer Modes section.
- Starting After Chaining. If the Software Request bit of the Channel Mode register is loaded with a 1 during chaining, the channel will perform the programmed DMA operation at the end of chaining. If the channel is

programmed for Single Operation or Demand mode, it will perform the operation immediately. The channel will give up the bus after chaining and before the operation if the CPU Interleave bit in the Master Mode register is set. Note that once a channel starts a chaining operation by fetching a reload word, it retains bus control at least until all of the registers specified in the reload word have been loaded from memory.

Transfers

The Z8016 DTC uses three basic types of operation: Transfer, Search, and Transfer-and-Search.

During a Transfer operation, the DTC obtains control of the system A/D bus from the CPU. Data is read from one addressable port (source) and is written to another addressable port (destination) in words or bytes. This applies to both Flyby and Flowthrough transfers.

Flyby transfers use a single addressing/transfer cycle, in which data is transferred directly from the source to the destination with no intermediate storage (Figure 3). This method of transfer provides higher throughput than Flowthrough transfers but cannot be used for memory-to-memory transfer.

Flowthrough transfers are used for all combinations of addressable memory and I/O spaces. These transfers use independent double Addressing/Transfer cycles, in which data is stored temporarily in the DTC while being transferred from source to destination (Figure 4). Flowthrough transfers can use the funneling option, which allows mixing of data sizes between source and destination. For example, a byte-oriented peripheral can conveniently supply data to a word-oriented memory. This option requires no added circuitry for either memory or peripherals.

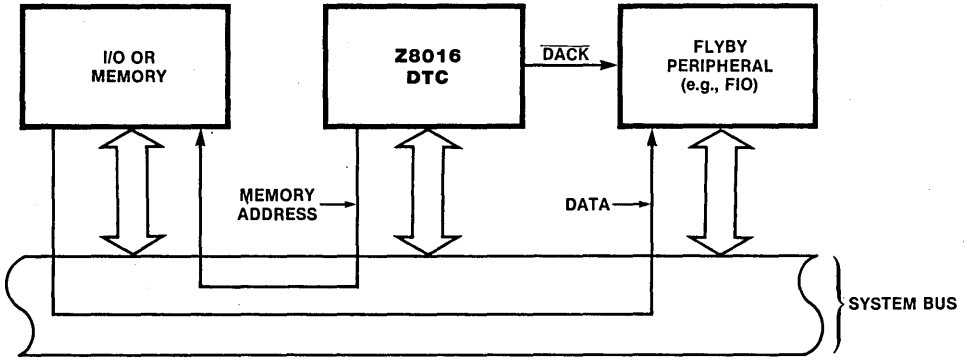


Figure 3. Configuration of a Flyby Transaction

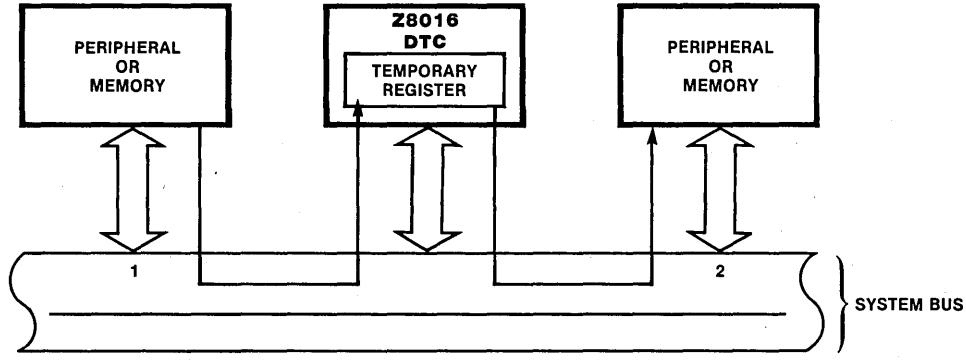


Figure 4. Configuration of a Flowthrough Transaction

During a Search operation, data is read from the source port and compared byte-by-byte with a pattern register containing a programmable match byte. The Search operation can be programmed to stop either when the read data matches (Stop-on-Match) or when it fails to match the masked pattern (Stop-on-No-Match). For word reads, the Channel Mode register can be used to select either 8- or 16-bit compares.

Transfer-and-Search operations combine the transfer and search functions to facilitate the transfer of variable-length data blocks. While data is being transferred between two ports, a simultaneous search is made for a bit-maskable byte match. Transfer-and-Search can be performed in either Flowthrough or Flyby mode. A Flyby Transfer-and-Search can be used to increase throughput for transfers between peripherals or between memory and a peripheral; it cannot be used for memory-to-memory transfers.

Transfer Modes. The Z8016 DTC operates in either of two transfer modes: Single or Demand. The Demand mode is further divided into the Demand Dedicated with Bus Hold, Demand Dedicated with Bus Release, and Demand Interleave modes.

The Single mode is used with peripherals that transfer single bytes or words at irregular intervals. Each Software Request command causes the channel to perform a single DMA operation and each application of a High-to-Low transition on the $\overline{DREQ}$ input also initiates a DMA operation. Each time a Single mode DMA operation ends, the channel relinquishes the bus unless a new transition has occurred on $\overline{DREQ}$.

In the Demand mode, when the $\overline{DREQ}$ input is active, transfer cycles are executed repeatedly until the transfer is completed. In the Demand Dedicated with Bus Hold mode, the active channel retains control of the bus until the transfer is complete, even after the $\overline{DREQ}$ input has gone inactive. In the Demand Dedicated with Bus Release mode, the active channel releases control of the bus when the $\overline{DREQ}$ input goes inactive. When the $\overline{DREQ}$ input becomes active again, control of the bus is re-acquired and the transfer operation continues.

The Demand Interleave mode has two options, programmable in the Master Mode register bit MM2. If MM2 is set, the DTC relinquishes and re-requests bus control after every DMA operation.

This permits the CPU and other devices to gain bus control. If both channels receive active $\overline{DREQ}$ inputs, each channel relinquishes control to the CPU after each operation. In the second option (MM2 is 0), control can pass from one channel to the other without requiring the DTC to release bus control. If both channels receive active $\overline{DREQ}$ inputs, control alternates between channels and the DTC retains bus control until all channel operations are complete.

Wait States. The Z8016 DTC can insert Wait cycles into the DMA Transaction cycle under hardware or software control. The $\overline{CS}/\overline{WAIT}$ input can be multiplexed to function as a Chip Select for the DTC when it does not have control of the bus, and as a $\overline{WAIT}$ input when the DTC is the bus controller. Multiplexing $\overline{CS}$ and $\overline{WAIT}$ requires external logic, but the DTC can be programmed to insert Wait states automatically without external logic when accessing either I/O or memory addresses. Either zero, one, two, or four Wait states can be added. Wait states can be programmed separately for the Current Address registers and for the Chain Address register. Programmable Wait cycle insertion allows memories and peripherals of different speeds to be associated with I/O and memory addresses.

Interrupts. On the Z8016 DTC, each channel is an interrupt source and has its own vector register for identifying the source of the interrupt during a CPU/DTC Interrupt Acknowledge transaction. An interrupt can result from a Match Condition (MC), End-Of-Process (EOP), or Terminal Count (TC) on either channel. The user selects the action to be performed by setting bits in the Channel Mode register.

Three bits in each channel's Status register control interrupts. These are the Channel Interrupt Enable (CIE) bit, the Interrupt Pending (IP) bit, and the Interrupt Under Service (IUS) bit.

Devices connected to any of the CPU's three interrupt inputs resolve priority conflicts with an interrupt daisy chain, as shown in Figure 5. The daisy chain has two functions. During an Interrupt Acknowledge transaction, it determines which interrupt source is being acknowledged. At all other times, it determines which interrupt sources can initiate an interrupt request.

The Z8016 DTC has an interrupt queuing capability, which includes a two-deep interrupt queue on each channel. This allows the DTC to continue normal operation between the time an interrupt is issued and the time the Interrupt Acknowledge is received.

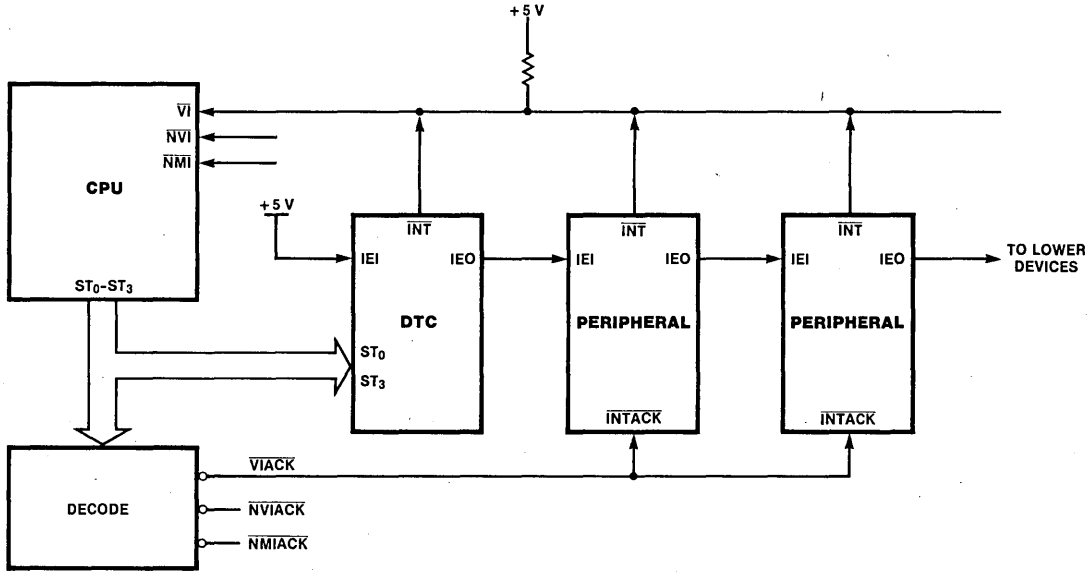


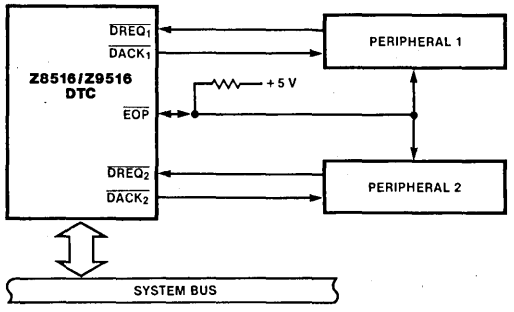
Figure 5. Interrupt Daisy Chain

Termination

There are three ways a Transfer-and-Search or Search operation can end and two ways a Transfer operation can end. When a channel's Current Operation Count goes to 0, the DMA operation ends; this is called a Terminal Count (TC) termination. A DMA operation can also be stopped by driving the $\overline{EOP}$ pin Low with external logic; this is called an $\overline{EOP}$ termination. Match Condition (MC) is the last method of termination which occurs when the data being Transferred-and-Searched or Searched meets the match condition programmed in Channel Mode register bits CM_{17}-CM_{16}. These bits allow the user to stop when a match occurs between the unmasked Pattern register bits and the data read from the source, or when a no-match occurs. Both byte and word matches are supported. MC terminations do not apply to Transfer operations since the pattern matching logic is disabled in Transfer mode.

End of Process

The End-of-Process ($\overline{EOP}$) interface pin is a bidirectional signal. Whenever a TC, MC, or $\overline{EOP}$ termination occurs, the DTC drives the $\overline{EOP}$ pin Low. During DMA operations, the $\overline{EOP}$ pin is sampled by the DTC to determine if $\overline{EOP}$ is being driven Low by external logic. Figure **19** shows when internal $\overline{EOP}$s are generated marking termination of all Transfers and when the $\overline{EOP}$ pin is sampled during the DMA iteration. The generation of internal $\overline{EOP}$s and sampling of external $\overline{EOP}$s for Transfer-and-Searches follows the same timing used for Transfers. Since there is a single $\overline{EOP}$ pin for both channels, $\overline{EOP}$s should only be driven Low by a channel while that channel is being serviced. This can be accomplished by selecting a level $\overline{DACK}$ output (CMR_{18} = 0) and gating each channel's $\overline{EOP}$ request with $\overline{DACK}$, as shown in Figure 5A



Notes:
1. External $\overline{EOP}$ stops channel.
2. DTC drives $\overline{EOP}$ Active on TC and MC.
3. Channel should apply $\overline{EOP}$ only if its $\overline{DACK}$ is active.

Figure 5A. $\overline{EOP}$ Connection

If an $\overline{EOP}$ is detected while the channel is trying to reload the Chain Address register, the new Chain Address Offset and Segment are discarded and the old address +2 is preserved to allow inspection of the erroneous address.

Programming Completion Options. When a channel ends a DMA operation, the reason for ending is stored in the Completion Status Field of the channel's Status register (Figure 7). This information is retained until the next DMA operation ends at which time the Status register is updated to reflect the reason(s) for the latest termination. More than one bit in the Completion Field could be set to 1. All three of the channel's Status register completion bits would be set to 1 under the following conditions: If a channel decremented its Current Operation Count to 0 causing a TC termination, input data from the source generated a match causing an MC termination, and a Low on the $\overline{EOP}$ pin resulted in an $\overline{EOP}$ termination.

When a DMA operation ends, the channel can:

(a) Issue an Interrupt request (i.e., setting the IP or SIP bit of the channel's Status register)

(b) Perform Base-to-Current reloading

(c) Chain reload the next DMA operation

(d) Perform any combination of the above or

(e) None of the above

The user selects the action to be performed by the channel in the Completion option field of the Channel Mode register. For each type of termination (TC, MC or EOP) the user can choose which action or actions are to be taken. If no reloading is selected for the type of termination that occurred, the NAC bit in the Status register is set.

More than one action can occur when a DMA operation ends. This may arise because more than one action was programmed for the applicable termination. The priorities of those actions are Interrupt request, Base-to-Current reloading, and chaining. The Interrupt cannot be serviced unless the DTC has relinquished the bus.

Interrupts

To permit the DTC to begin a new DMA operation after issuing an interrupt but before the CPU acknowledges that interrupt, a two-deep interrupt queue is provided on each channel of the DTC. Interrupt handling by the Z8000 microprocessor is summarized in this section, followed by a brief discussion of the DTC's queueing capability and its implications for the system.

A complete Interrupt cycle on the Z8000 CPU consists of an Interrupt Request followed by an Interrupt Acknowledge transaction. The request, which consists of the CPU's Interrupt pin being pulled Low by a peripheral, notifies the processor that an interrupt is pending. The Interrupt Acknowledge cycle, initiated by the CPU as a result of the interrupt request, performs two functions: it selects the peripheral whose interrupt is to be acknowledged and it obtains a vector that identifies the device involved and the reason for the interrupt.

The DTC has two sources of interrupt. Each source has three bits that control its interrupt generation. These bits are the Channel Interrupt Enable (CIE), Interrupt Pending (IP), and Interrupt Under Service (IUS) bits. Since each channel on the DTC contains all three of these bits (bits CM_{15}-CM_{13}), they are seen by the CPU as two separate interrupt sources. Each channel also has its own vector register for identifying the source of the interrupt during an Interrupt Acknowledge interchange with the CPU. The Disable Lower Chain (DLC) and No Vector (NV) bits in the DTC's Master Mode register control this behavior for the entire chip.

Once a channel issues an interrupt, it is desirable to allow the channel to proceed with the next DMA operation before the interrupt is acknowledged. This could lead to problems if the DTC channel attempted to chain reload the Vector register contents. In such a situation, it may not be clear whether the old or new vector would be returned during the acknowledge. This dilemma is resolved in the DTC by providing each channel with an Interrupt Save register. When the channel sets IP as part of the procedure followed to issue an interrupt, the contents of the vector register and some of the Status register bits are saved in an Interrupt Save register (Figure 9). When an Interrupt Acknowledge cycle is performed, the contents of the Interrupt Save register are driven onto the bus. Although the use of an Interrupt Save register allows the channel to proceed with a new task, problems can still arise if a second interrupt is to be issued by the channel before the first interrupt is acknowledged. To avoid conflicts between the first and second interrupt, each channel has a Second Interrupt Pending (SIP) bit in its Status register. When a second interrupt is issued before the first interrupt is acknowledged, the SIP bit is set and the channel relinquishes the bus until an acknowledge occurs. For compatibility with polled interrupt schemes, the Interrupt Save register can be read without wait states by the host CPU. As an aid to debugging a system's interrupt logic, whenever IP is set, the Interrupt Save register is loaded from the Vector and Status registers.

Note that the SIP bit is transferred to the IP bit when IP is cleared by the host CPU. Whenever CIE is set, INT goes Low when IP is set.

Base-to-Current Reloading. When a channel finishes a DMA operation, the user may select to perform a Base-to-Current Reload. (Base-to-Current reloading is also referred to as Auto-reloading in this document.) In this type of reload, the Current Address registers A and B are loaded with the data in the Base Address registers A and B respectively, and the Current Operation Count register is loaded with the data in the Base Operation Count. The Base-to-Current reload operation facilitates repetitive DMA operations without the multiple memory accesses required by chaining. Although the channel must have bus control to perform Base-to-Current reloading, the complete reloading operation occurs in four clock cycles (TAU_1 through TAU_4). If the channel has to relinquish the bus because two unacknowledged interrupts are queued, it has to regain bus control to perform any Base-to-Current reloading (or chaining). In this case it acquires the system bus once an interrupt acknowledge is received, even if it immediately afterward relinquishes the bus because no hardware or software request is present.

Chaining. If the channel is programmed to chain at the end of a DMA operation, it uses the Chain Address register to point to a Chain Control Table in memory. The first word in the table is a Reload word, specifying the register(s) to be loaded. Following the Reload word are the data values to be transferred into the register(s). Chaining is described in detail in the Channel Initialization section.

Because chaining occurs after Base-to-Current reloading, it is possible to reset the Current Address registers A and B and the Current Operation Count register to the values used for previous DMA operations and then chain reload one or two of these registers to some special value. If the Base values are not reloaded during chaining, the channel can revert back to the Base values at a later cycle.

If an all zero Reload word is fetched during chaining, the chain operation does not reload any registers but performs like any other chaining operation. Thus, the Chain Address is incremented by 2 to point to the next word in memory and, at the end of the all Zero-Reload word chain operation, the channel is ready to perform a DMA operation. All zero Reload words are useful as "Stubs" to start or terminate linked lists of DMA operations traversed by chaining. Care must be taken in their use since the channel may perform an erroneous operation if it is unintentionally started after the chaining operation.

MEMORY MANAGEMENT

The DTC can be configured to operate in physical address space or logical address space. When the DTC is operated in logical address space, the segment and offset portions of the address registers combine to form 23-bit logical addresses. In conjunction with a CPU, DMA operations can be handled through the Z8010 MMU. MMUs offer dynamic segment relocation, segment protection, and other memory management features.

In the physical address space configuration, the segment and offset portions of the DTC's address registers are combined with the SN_7 output to form a single 24-bit linear address. The extended I/O addressing capability of the DTC can be used to increase the DTC's physical I/O address space beyond that of the CPU. Figure 6 illustrates various DTC configuration options with the Z8000 CPUs and MMUs.

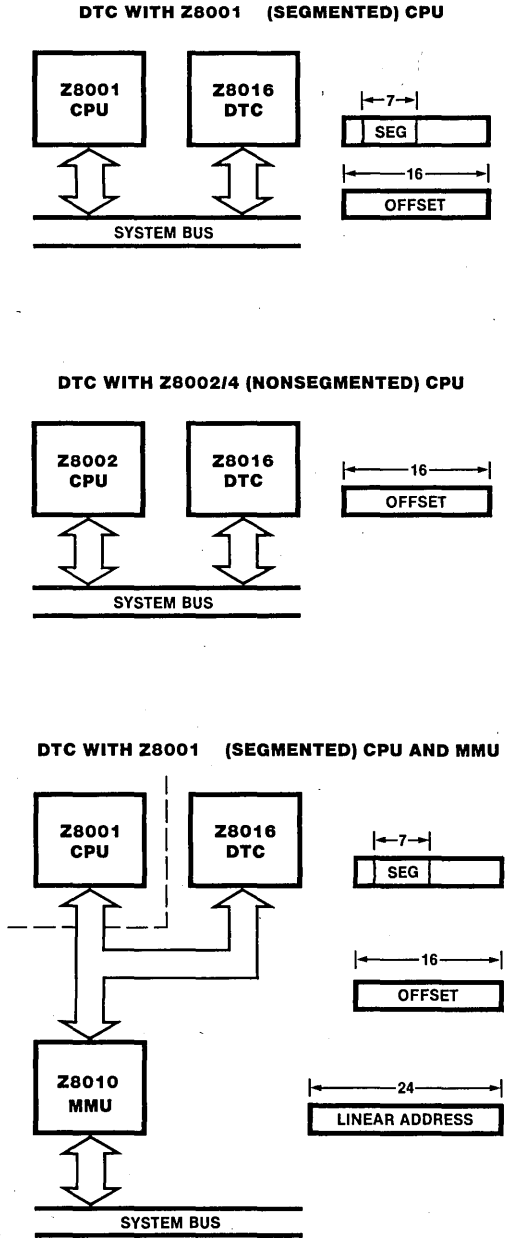


Figure 6. DTC Configurations

INTERNAL STRUCTURE

The internal structure of the Z8016 DTC includes driver and receiver circuitry for interfacing with Zilog's Z-BUS. The DTC's internal bus interfaces with the Z-BUS and services all internal logic and registers, as illustrated in the DTC block diagram (Figure 7).

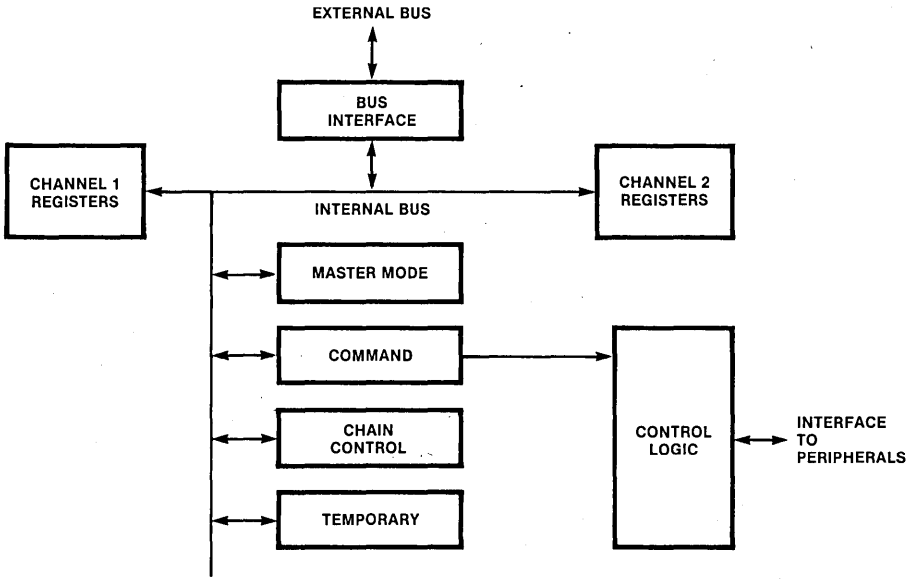


Figure 7. DTC Block Diagram

REGISTER DESCRIPTION

The DTC contains chip-level control registers as well as channel-level registers that are duplicated for each channel. Registers on the DTC that can be read by the CPU are either fast- or slow-readable. CPU I/O instructions can read fast-readable registers without Wait states. Slow-readable registers can be read by the CPU only if Wait states are inserted. This requires external logic to generate and time the application of Low signals on the CPUs $\overline{WAIT}$ input if the slow-readable registers are to be read.

Control Registers

The four control registers direct the functioning of the DTC. (Figure 8.)

Master Mode Register. This register selects the way in which the DTC interfaces to the system. The following descriptions indicate how the individual bits in the Master Mode register are used. The Master Mode register is fast-readable.

Chip Enable (CE). The setting of this bit enables the DTC to request the bus, perform DMA operations and reload registers.

Logical/Physical Address Space (LPA). The setting of this bit determines how the system will view the segment and offset portions of the Current ARA and ARB registers. When LPA is set to 1 (Logical Address Space), the segment and offset portions of the Current ARA and ARB registers are treated as separate portions of the address. The 16-bit offset portion of the address will appear on pins AD_0–AD_{15} when $\overline{AS}$ is Low. The 7-bit segment number appears on pins SN_0–SN_6 for the duration of the transaction.

When this bit is set to 0 (Physical Address Space), the segment and offset portions of the Current ARA and ARB registers are treated as a single address and all eight segment bits in the register are used. Both the I/O and the memory addresses in Physical Memory Space are generated by loading the offset portion of the Current Address register onto the AD_0–AD_{15} bus and the segment portion of that register onto the SN_0–SN_7 bus. (In conjunction with the nonsegmented Z8000 CPUs, either Logical or Physical Address Space setting may be used.)

Wait Line Enable (WLE). This bit is set to enable sampling of the $\overline{CS/WAIT}$ line during memory and I/O transactions.

Disable Lower Chain (DLC). This bit is set to inhibit all lower priority devices on the interrupt daisy chain. While DLC is 0, the DTC generates Low and High signals on the IEO output in response to IEI.

No Vector on Interrupt (NVI). This bit determines whether the DTC channel or a peripheral returns a vector during Interrupt Acknowledge cycles. While the bit is cleared, a channel receiving an Interrupt Acknowledge will drive the contents of its Interrupt Save register onto the A/D bus while $\overline{DS}$ is Low. While this bit is set, interrupts are serviced in an identical manner, but the A/D bus remains in a high impedance state throughout the Acknowledge cycle.

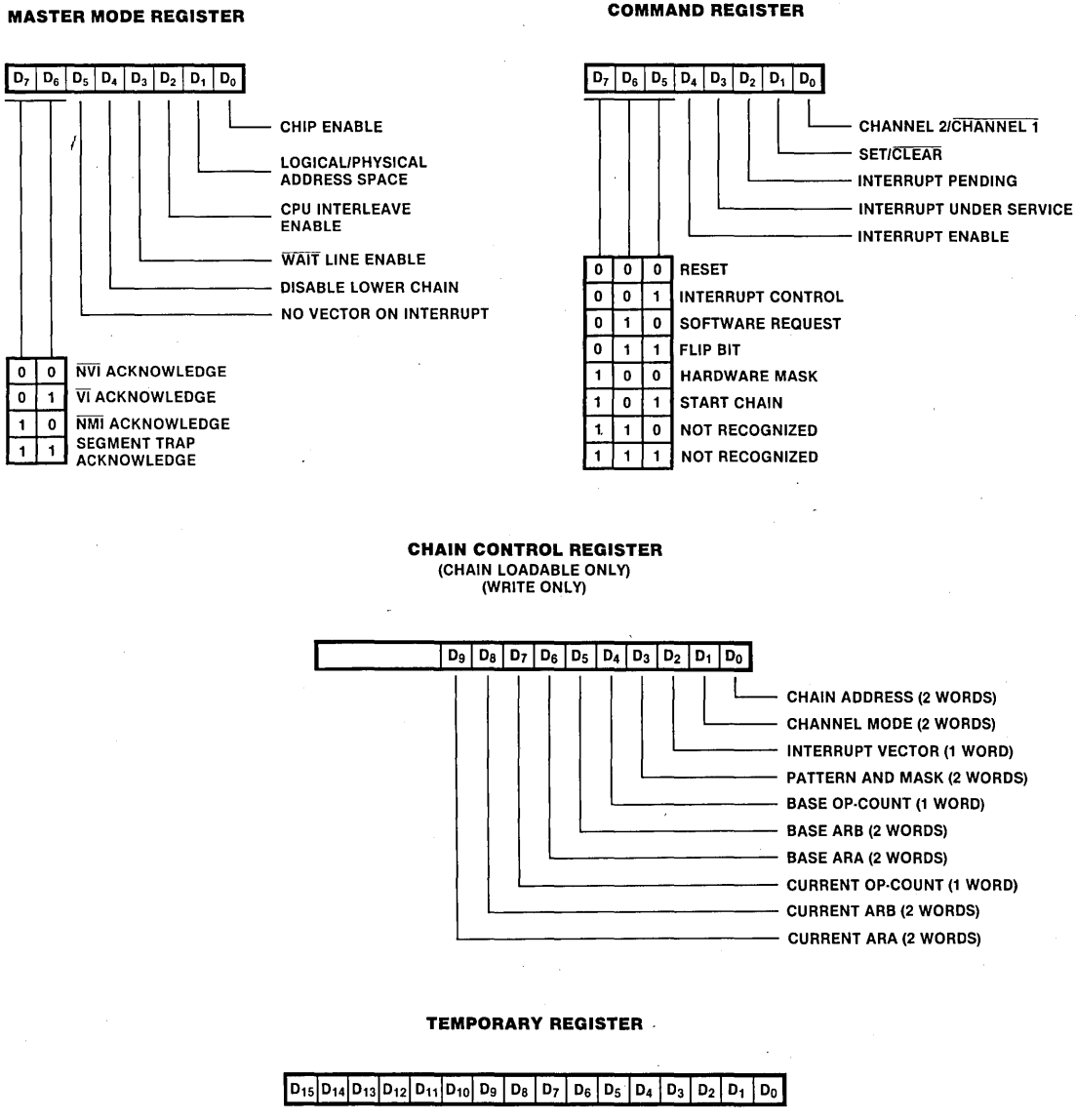


Figure 8. Control Registers

Interrupt Acknowledge Field (two bits). This field is used to select the type of Interrupt Acknowledge cycle the DTC is to respond to. The setting of this field must correspond to the IEI/IEO daisy chain on which the DTC is located. The DTC can respond to Nonmaskable Interrupt (NMI), Nonvectored Interrupt (NVI), or Segment Trap Acknowledge cycles.

CPU Interleave Enable. When this bit is set, interleaving of bus use between the CPU and the DTC is enabled.

Chain Control Register. This 16-bit register specifies which registers are to be loaded from memory during a chaining operation. The Chain Control register is loaded from the memory location pointed to by the Chain Address register. The Chain Control register is chain loadable only and cannot be accessed by the CPU.

Command Register. The Command register is an 8-bit write-only register written to by the host CPU to execute commands. The Command register is loaded from the data on AD_7–AD_0; the data on AD_{15}–AD_8 is disregarded.

Temporary Register. This 16-bit register is used to hold data during Flowthrough transfers, Search operations, and Transfer-and-Search operations. The Temporary register cannot be written or read by the CPU.

Channel-Level Registers

Each of the DTC's two channels has a complete set of channel-level registers. This set consists of both General-Purpose and Special-Purpose registers, as illustrated in Figure 9. The General-Purpose registers are commonly found on DMA devices and can be read or written by the CPU. The Special-Purpose registers provide additional features specific to the Z8016 DTC.

General-Purpose Registers. The General-Purpose register set on each channel consists of the Current Address registers A and B, the Base Address registers A and B, the Base and Current Operation Count registers, and the Channel Mode register (Figure 10).

Current and Base Address Registers A and B. The Current Address registers A and B are used to point to the source and destination for DMA operations. The contents of the Base Address registers A and B are transferred into the Current Address registers A and B at the end of a DMA operation if the user enables base-to-current reloading in the Completion field of the Channel Mode register. The base-to-current reload operation facilitates repetitive DMA operations without the multiple memory accesses required by chaining.

Each of the Base and Current Address registers A and B consist of two words. The first word contains a 7-bit Tag field and an 8-bit Segment Number field. The second word contains a 16-bit offset. The use of the Tag field is described below. The use of the Segment Number field depends upon the setting of the LPA bit in the Master Mode register. The Base and Current Address registers are fast-readable and can be loaded by chaining.

Programmable Wait Field. This field allows the insertion of zero, one, two, or four Wait states into memory or I/O accesses addressed by the offset and segment fields.

Address Control Field. At the end of each iteration of a DMA operation, the address can be incremented, decremented, or left unchanged. Memory addresses are changed by one if the address points to a byte operand or by two if the address points to a word operand.

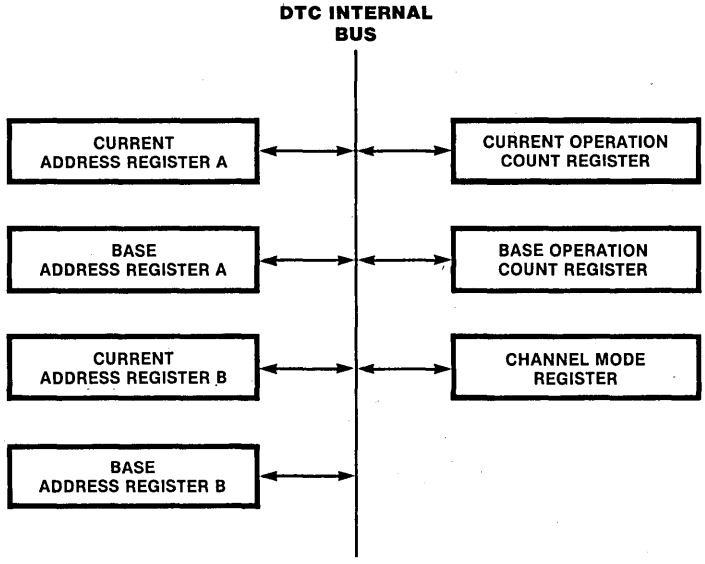
Address Reference Field. This portion of the Tag field is used to select whether the address pertains to memory space or I/O space. The N/$\overline{S}$ output line is always Low (indicating System) for I/O space but can be either High (Normal) or Low (System) for memory space.

Current and Base Operation Count Registers. The 16-bit Current Operation Count register specifies the number of words or bytes to be transferred, searched, or transferred-and-searched. For word-to-word operations and byte-word funneling, this register must be programmed with the number of words to be transferred or searched.

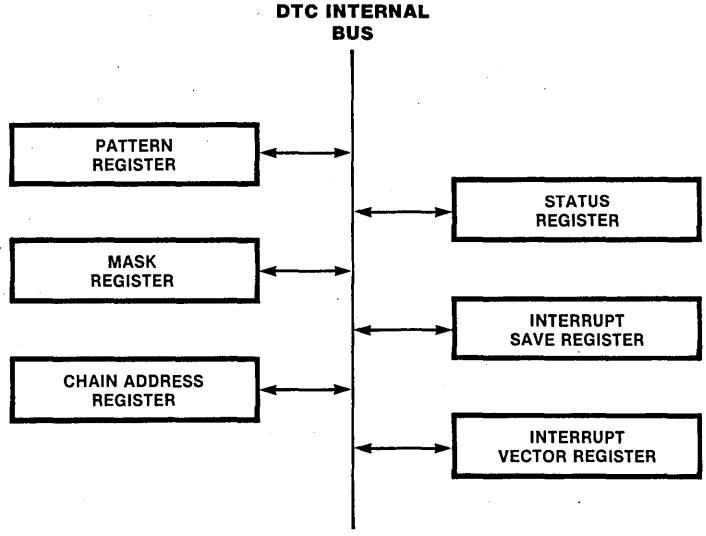
The Base Operation Count register reinitializes the current source and destination in the Current Operation Count register. Each time data is transferred or searched, the Current Operation Count register is decremented by one. Once all of the data is transferred or searched, the Current Operation Count register will contain zero. If the transfer on search stops before the Current Operation Count register reaches zero, the contents of the register indicate the number of bytes or words remaining to be transferred or searched. This allows a channel to be restarted from where it left off without requiring reloading of the Current Operation Count register. The Current and Base Operation Count registers are slow-readable and can be loaded by chaining.

Channel Mode Register. This register selects the type of DMA operation the channel is to perform, how the operation is to be executed, and what action is to be taken when the operation finishes. The Channel Mode register is slow-readable and can be loaded by chaining.

Data Operation and Transfer Type Field. These fields are used to select the type of operation the channel is to perform along with the operand size. The specific codes are listed in Tables 2 and 3. The Flip bit is used to select which of the Current Address Registers A (ARA), or B (ARB), points to the source and which points to the destination address.



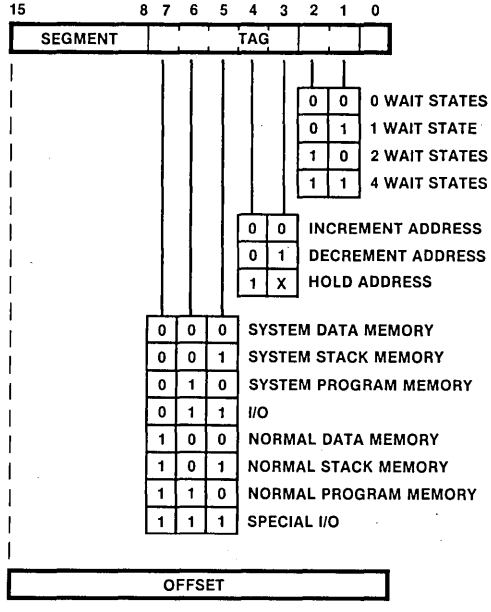
GENERAL-PURPOSE CHANNEL REGISTERS



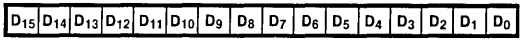
SPECIAL-PURPOSE CHANNEL REGISTERS

Figure 9. Channel-Level Registers

BASE AND CURRENT ADDRESS REGISTERS A AND B



BASE AND CURRENT OPERATION COUNT REGISTERS



CHANNEL MODE REGISTER

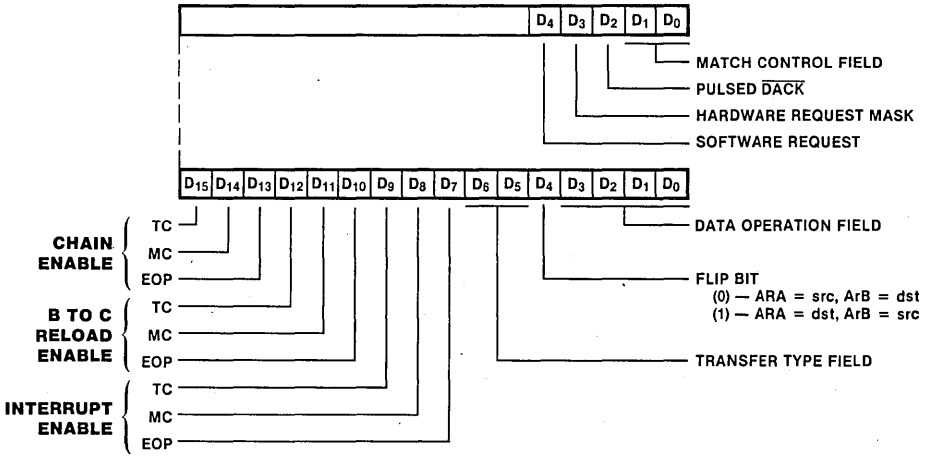


Figure 10. General-Purpose Channel Registers

Table 2. Data Operation Field

Code/Operation	Operand Size ARA	Operand Size ARB	Transaction Type
Transfer			
0001	Byte	Byte	Flowthrough
100X	Byte	Word	Flowthrough
0000	Word	Word	Flowthrough
0011	Byte	Byte	Flyby
0010	Word	Word	Flyby
Transfer-and-Search			
0101	Byte	Byte	Flowthrough
110X	Byte	Word	Flowthrough
0100	Word	Word	Flowthrough
0111	Byte	Byte	Flyby
0110	Word	Word	Flyby
Search			
1111	Byte	Byte	N/A
1110	Word	Word	N/A
101X	Illegal		

Completion Field. This field is used to program the action taken by the channel at the end of a DMA operation. When a DMA operation ends, the channel can perform any combination of the following options:

- Interrupt the CPU (Interrupt Enable field)
- Base-to-Current reload (B to C Reload field)
- Chain reload the next DMA operation (Chain Enable field)

The options are performed according to the bits set in the Interrupt Enable, B to C Reload, and Chain Enable fields for each type of termination that occurs; the NAC bit in the Status register is automatically set on completion of a DMA operation.

Match Control Field. This 2-bit field determines whether matches use an 8-bit or 16-bit pattern and whether the channel is to Stop-On-Match or Stop-On-No-Match. The specific codes for the Match Control field are listed in Table 3.

Table 3. Transfer Type Field and Match Control Field

Code	Transfer Type	Match Control
00	Single Transfer	Stop on No Match
01	Demand Dedicated/Bus Hold	Stop on No Match
10	Demand Dedicated/Bus Release	Stop on Word Match
11	Demand Interleave	Stop on Byte Match

Pulse $\overline{\text{DACK}}$ (PD). This bit determines when the $\overline{\text{DACK}}$ line is active. While cleared, the channel's $\overline{\text{DACK}}$ line is active whenever the channel is performing a DMA operation, regardless of the type of transaction. While the PD bit is set, the $\overline{\text{DACK}}$ pin is inactive during chaining, Flowthrough Transfers, Flowthrough Transfer-and-Searches, and Searches. $\overline{\text{DACK}}$ is pulsed active during Flyby Transfers and Flyby Transfer-and-Searches at the time necessary to strobe data into, or out of, the Flyby peripheral.

Hardware Request Mask (HRM). If this bit is set, a DMA operation can be started by applying a Low on the channel's $\overline{\text{DREQ}}$ input.

Software Request (SR). If this bit is set during chaining, the channel performs the programmed DMA operation at the end of the chaining operation.

Special Purpose Registers. The Special-Purpose registers on each channel are the Pattern and Mask registers, the Status register, the Interrupt Vector register, the Interrupt Save registers, and the Chain Address register (Figure 11).

Pattern and Mask Registers. These registers are used in Search and Transfer-and-Search operations. The Pattern register contains the pattern that the read data is compared to. The Mask register allows the user to exclude or mask selected Temporary register bits from comparison by setting the corresponding Mask register bit to 1. The Pattern and Mask registers are slow-readable and can be loaded by chaining.

Status Register. The Status register on each channel reports the status of that channel. The functions of the individual bits are indicated in the following field descriptions. The Status register is fast-readable.

Completion Status Field. Three bits indicate whether the DMA operation ended as a result of TC, MC, or EOP. The TC bit is set if the Operation Count (reaching zero) ends the DMA operation. The MC bit is set if a pattern match termination occurs. The EOP bit is set when an EOP termination ends a DMA transfer. The appropriate combination of the TC, MC, and EOP bits is set if multiple reasons exist for ending a DMA operation. The Match Condition High byte (MCH) and Match Condition Low byte (MCL) bits report the match states of the upper and lower comparator bytes of the last word transferred. The MCH and MCL bits are updated with each transfer.

These bits are set when the associated comparator bytes are matched, regardless of whether Stop-on-Match or Stop-on-no-Match is programmed.

Hardware Interface Status Field. The Hardware Request (HRQ) bit provides a means of monitoring the channel's $\overline{\text{DREQ}}$ input line. While $\overline{\text{DREQ}}$ is Low, the HRQ bit is set. While the Hardware Mask (HM) bit is set, the DTC is prevented from responding to a Low on the $\overline{\text{DREQ}}$ line. However, the HRQ bit always reports the status of $\overline{\text{DREQ}}$ regardless of the status of the HM bit.

DTC Status Field. This field reports the current channel status to the CPU. The "channel initialized and waiting for request" status is implicitly indicated if bits ST_{12} through ST_9 are clear.

Second Interrupt Pending (SIP). When a second interrupt is to be issued before the first interrupt is acknowledged, this bit is set and the channel relinquishes the bus until an Acknowledge occurs.

Waiting for Bus (WFB). This bit is set when the channel is waiting for bus control to perform a DMA operation.

No Auto-Reload or Chaining (NAC). This bit is set under the following conditions:

- A channel completes a DMA operation and neither Base-to-Current reloading nor auto-chaining is enabled.
- A channel is issued an $\overline{EOP}$ during chaining.
- A Reset is issued to the DTC.

Chaining Abort (CA). This bit is set when a channel is issued an $\overline{EOP}$ during chaining or a Reset is issued to the DTC. The Chain Abort (CA) bit holds the No Auto-Reload or Chaining (NAC) bit in the set state until the $\overline{EOP}$ bit is cleared. The CA bit is cleared when a new Chain Address Segment and Tag word or Offset word is loaded into the channel.

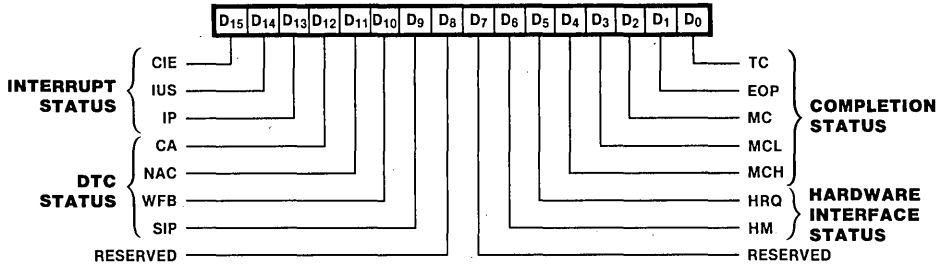
Interrupt Status Field. The Channel Interrupt Enable (CIE), Interrupt Pending (IP), and Interrupt Under Service (IUS) bits are used to control the way a channel generates an interrupt. An interrupt source with its IP bit set makes an interrupt request if all of the following conditions are met: Interrupts are enabled, (CIE bit = 1), there is no Interrupt Under Service (IUS bit = 0), no higher priority interrupt is being serviced, and no Interrupt Acknowledge transaction is in progress. When an interrupt source has an Interrupt Under Service (IUS = 1), all lower priority interrupt sources are prevented from requesting interrupts.

Interrupt Vector and Interrupt Save Registers. The 8-bit Interrupt Vector register contains the vector or identifier to be output during an Interrupt Acknowledge cycle. When an interrupt occurs, the contents of the Interrupt Vector register and bits ST_9–ST_{15} of the Status register are stored in the 16-bit Interrupt Save register. Because the vector and status are stored, a new vector can be loaded during chaining and a new DMA operation can be performed before an Interrupt Acknowledge cycle occurs. If another interrupt occurs on the channel before the first is acknowledged, further channel activity is suspended. When a clear IP command is issued, the status and vector for the second interrupt are loaded into the Interrupt Save register and channel operation resumes. The DTC can retain only two interrupts for each channel. The Interrupt Save register is fast-readable.

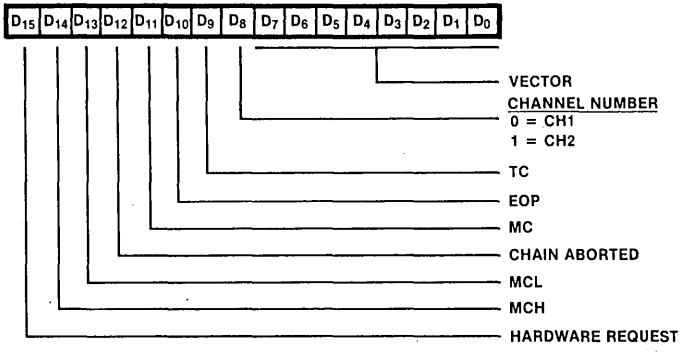
Chain Address Register. This register points to the chain control table in memory containing data to be loaded into the channel's registers. The Chain Address register consists of two words (Figure 11). The first word consists of a Segment and Tag field. The second word contains the 16-bit offset portion of the memory address. Bit 15 in the Segment field is ignored when the DTC is configured for logical address space (LPA = 1). The Tag field contains two bits used to designate the number of Wait states to be inserted during accesses to the Chain Control table. The Chain Address register is fast-readable and is loadable by chaining.

Table 4 provides a list of register addresses.

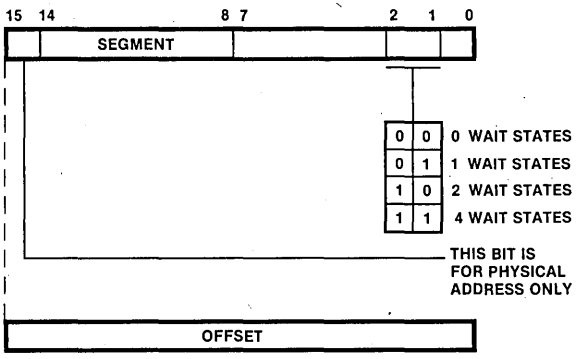
STATUS REGISTER



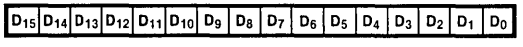
INTERRUPT SAVE REGISTER



CHAIN ADDRESS REGISTER



PATTERN AND MASK REGISTERS



INTERRUPT VECTOR REGISTER

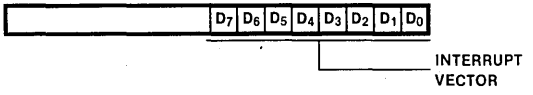


Figure 11. Special-Purpose Channel Registers

Table 4. Register Address Summary

Address (AD_7-AD_0)	(Hex)	Control Registers
X 0 1 1 1 0 0 X	38	Master Mode
X 0 1 0 1 1 1 X	2E	Command Channel 1
X 0 1 0 1 1 0 X	2C	Command Channel 2
		General-Purpose Channel Registers
X 0 0 1 1 0 1 X	1A	Current Address Register A-Channel 1, Segment/Tag
X 0 0 0 1 0 1 X	0A	Current Address Register A-Channel 1, Offset
X 0 0 1 1 0 0 X	18	Current Address Register A-Channel 2, Segment/Tag
X 0 0 0 1 0 0 X	08	Current Address Register A-Channel 2, Offset
X 0 0 1 0 0 1 X	12	Current Address Register 8-Channel 1, Segment/Tag
X 0 0 0 0 0 1 X	02	Current Address Register B-Channel 1, Offset
X 0 0 1 0 0 0 X	10	Current Address Register B-Channel 2, Segment/Tag
X 0 0 0 0 0 0 X	00	Current Address Register B-Channel 2, Offset
X 0 0 1 1 1 1 X	1E	Base Address Register A-Channel 1, Segment/Tag
X 0 0 0 1 1 1 X	0E	Base Address Register A-Channel 1, Offset
X 0 0 1 1 1 0 X	1C	Base Address Register A-Channel 2, Segment/Tag
X 0 0 0 1 1 0 X	0C	Base Address Register A-Channel 2, Offset
X 0 0 1 0 1 1 X	16	Base Address Register B-Channel 1, Segment/Tag
X 0 0 0 0 1 1 X	06	Base Address Register B-Channel 1, Offset
X 0 0 1 0 1 0 X	14	Base Address Register B-Channel 2, Segment/Tag
X 0 0 0 0 1 0 X	04	Base Address Register B-Channel 2, Offset
X 0 1 1 0 0 1 X	32	Current Operation Count Channel 1
X 0 1 1 0 0 0 X	30	Current Operation Count Channel 2
X 0 1 1 0 1 1 X	36	Base Operation Count Channel 1
X 0 1 1 0 1 0 X	34	Base Operation Count Channel 2
		Special-Purpose Channel Registers
X 1 0 0 1 0 1 X	4A	Pattern Channel 1
X 1 0 0 1 0 0 X	48	Pattern Channel 2
X 1 0 0 1 1 1 X	4E	Mask Channel 1
X 1 0 0 1 1 0 X	4C	Mask Channel 2
X 0 1 0 1 1 1 X	2E	Status Channel 1
X 0 1 0 1 1 0 X	2C	Status Channel 2
X 0 1 0 1 0 1 X	2A	Interrupt Save Channel 1
X 0 1 0 1 0 0 X	28	Interrupt Save Channel 2
X 1 0 1 1 0 1 X	5A	Interrupt Vector Channel 1
X 1 0 1 1 0 0 X	58	Interrupt Vector Channel 2
X 0 1 0 0 1 1 X	26	Chain Address, Channel 1 Segment/Tag
X 0 1 0 0 0 1 X	22	Chain Address, Channel 4 Offset
X 0 1 0 0 1 0 X	24	Chain Address, Channel 2 Segment/Tag
X 0 1 0 0 0 0 X	20	Chain Address, Channel 2 Offset
X 1 0 1 0 1 1 X	56	Channel Mode Channel 1 High
X 1 0 1 0 0 1 X	52	Channel Mode Channel 1 Low
X 1 0 1 0 1 0 X	54	Channel Mode Channel 2 High
X 1 0 1 0 0 0 X	50	Channel Mode Channel 2 Low

NOTE: X = ignored.

ADDRESSING

The address generated by the DTC is always a byte address, even though the memory is organized as 16-bit words. All word-sized data is word-aligned and must be addressed by even addresses (A_0 = 0). With byte transfers, the least significant address bit determines which half of the A/D bus is used for the transfer. An even address specifies the most significant byte (AD_8–AD_{15}), and an odd address specifies the least significant byte (AD_0–AD_7). This addressing mechanism applies to memory accesses as well as to I/O and Special I/O accesses.

COMMANDS

The Z8016 DTC responds to several commands that give the CPU direct control over operating parameters. The commands described below are executed immediately after being written by the CPU into the DTC's Command register. A summary of the DTC commands is given in Table 5.

Reset

The Reset command forces the DTC into an idle state, in which it waits for a Start Chain command. The Start Chain command initiates a chain operation on either channel.

Software Request

A channel's Software Request command initiates a previously programmed transfer. If both channels are active, Channel 1 has priority.

Set/Clear Hardware Mask

The Set/Clear Hardware Mask command sets or clears the Hardware Mask bit in the selected channel's Mode register.

Table 5. DTC Command Summary

Command	Opcode Bits 7654	Opcode Bits 3210	Example Code (HEX)
Reset	000X	XXXX	00
Start Chain Channel 1	101X	XXX0	A0
Start Chain Channel 2	101X	XXX1	A1
Clear Software Request Channel 1	010X	XX00	40
Clear Software Request Channel 2	010X	XX01	41
Set Software Request Channel 1	010X	XX10	42
Set Software Request Channel 2	010X	XX11	43
Clear Hardware Mask Channel 1	100X	XX00	80
Clear Hardware Mask Channel 2	100X	XX01	81
Set Hardware Mask Channel 1	100X	XX10	82
Set Hardware Mask Channel 2	100X	XX11	83
Clear CIE, IUS, IP Channel 1	001E	SP00	*
Clear CIE, IUS, IP Channel 2	001E	SP01	*
Set CIE, IUS, IP Channel 1	001E	SP10	*
Set CIE, IUS, IP Channel 2	001E	SP11	*
Clear Flip Bit Channel 1	011X	XX00	60
Clear Flip Bit Channel 2	011X	XX01	61
Set Flip Bit Channel 1	011X	XX10	62
Set Flip Bit Channel 2	011X	XX11	63

*NOTES:
1. E = Set to 1 to perform set/clear on CIE, Clear to 0 for no effect on CIE.
2. S = Set to 1 to perform set/clear on IUS, Clear to 0 for no effect on IUS.
3. P = Set to 1 to perform set/clear on IP, Clear to 0 for no effect on IP.
4. X = "don't care" bit. This bit is not decoded and may be 0 or 1.
5. Flip bit = reset to 0 for ARA = src, ARB = dst. Set to 1 for ARA = dst, ARB = src.

Set/Clear IP, IUS, and CIE

The Set/Clear IP, IUS, and CIE commands manipulate the Interrupt Control bits located in each channel's Status register. These bits implement the interrupt daisy-chain control. The IP, IUS, and CIE bits for each channel can be set and cleared individually or in combination.

Set/Clear Flip Bit

The Set/Clear Flip Bit command reverses the source and destination, thereby reversing the direction of data transfer without reprogramming the channel.

TIMING

The following descriptions and timing diagrams refer to the relative timing relationships of DTC signals during basic operations. For exact timing information, refer to the composite timing diagrams.

Bus Request And Acknowledge

Before the DTC can perform a DMA operation, it must gain control of the system bus. The $\overline{BUSREQ}$, $\overline{BAI}$, and $\overline{BAO}$ interface pins provide connections between the DTC and the host CPU and other DMA devices to arbitrate which device has control of the system bus. When the DTC wants to gain bus control, it drives $\overline{BUSREQ}$ Low. Bus Request and Acknowledge timing is shown in Figure 12.

Flowthrough Transactions

Timing for Flowthrough I/O and Flowthrough Memory transactions (Figures 13 and 14, respectively) is identical. There are two types of I/O space on the Z8016: I/O and Special I/O. Status lines ST_0–ST_3 specify when an I/O operation is being performed and which of the two I/O spaces is being accessed. During an I/O transaction, status signal N/$\overline{S}$ will be Low to indicate a System Level operation.

The timing for I/O operations is identical to the timing of Flowthrough memory transactions. An I/O cycle consists of three states: T_1, T_2, and T_3. The TWA state is a Wait state that can be inserted into the transaction cycle. The $\overline{AS}$ output is pulsed Low to mark the beginning of a T-cycle. The N/$\overline{S}$ line is set Low (System) and the R/$\overline{W}$ and B/$\overline{W}$ lines select Read or Write operations for bytes or words. The N/$\overline{S}$, R/$\overline{W}$ and B/$\overline{W}$ lines become stable during T_1 and remain stable until the end of T_3.

I/O address space is byte-addressed but both 8- and 16-bit data sizes are supported. During I/O transactions, the B/$\overline{W}$ output is High for byte transactions and Low for word transactions.

The R/$\overline{W}$ output is High during Read operations and Low during Write operations. $\overline{DS}$ is driven Low to signal the peripherals that data can be gated onto, or received from, the bus. $\overline{DS}$ is driven High to signal the end of the I/O transaction.

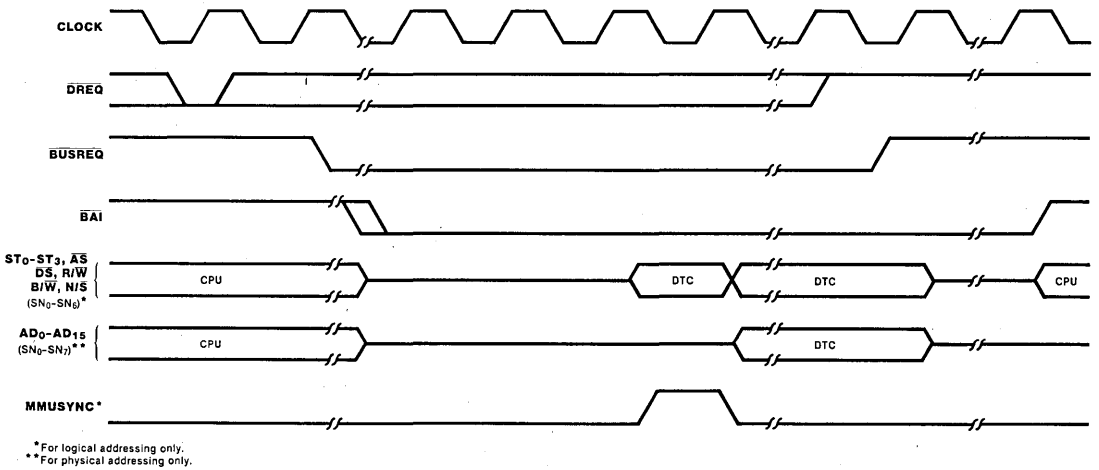
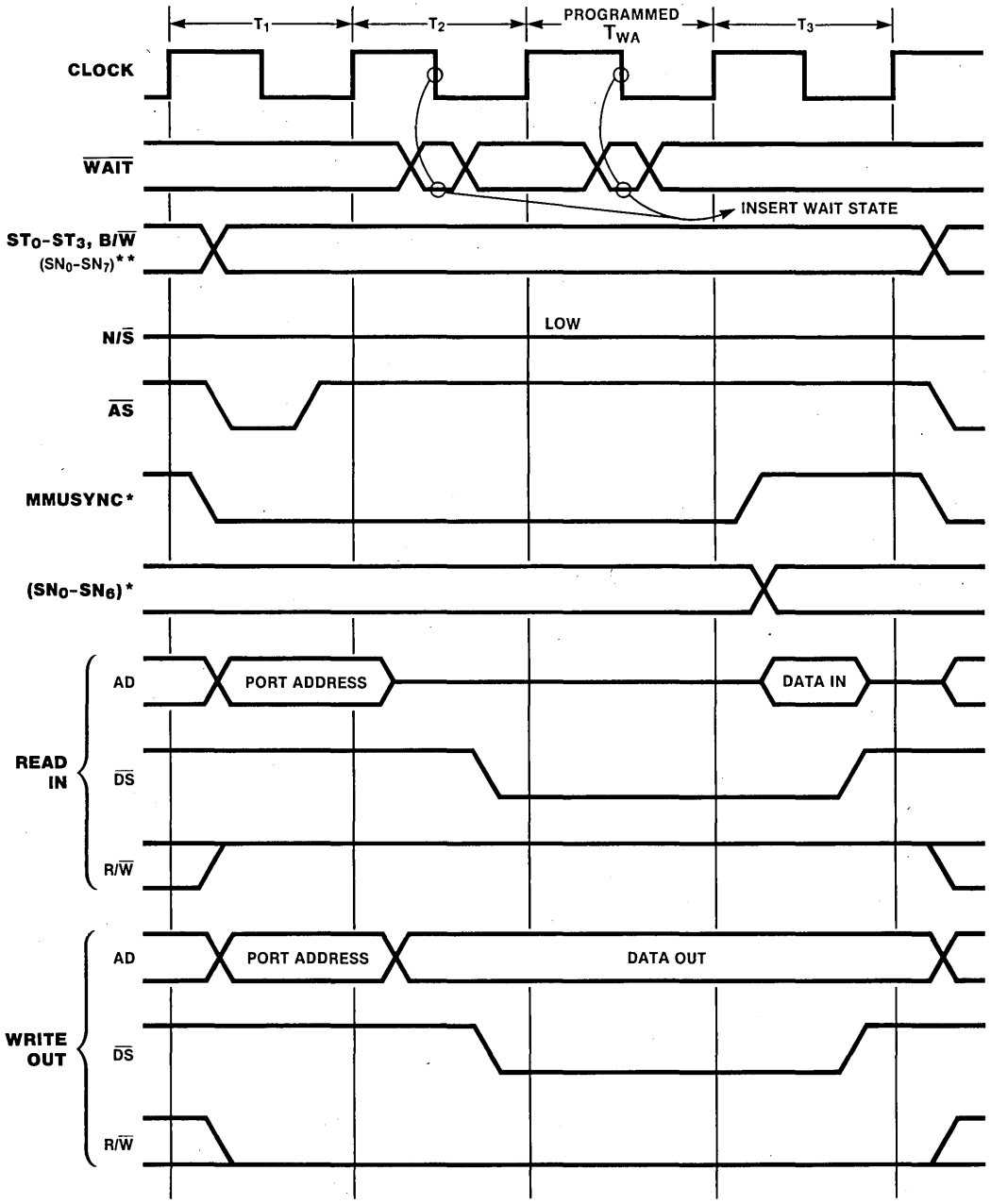
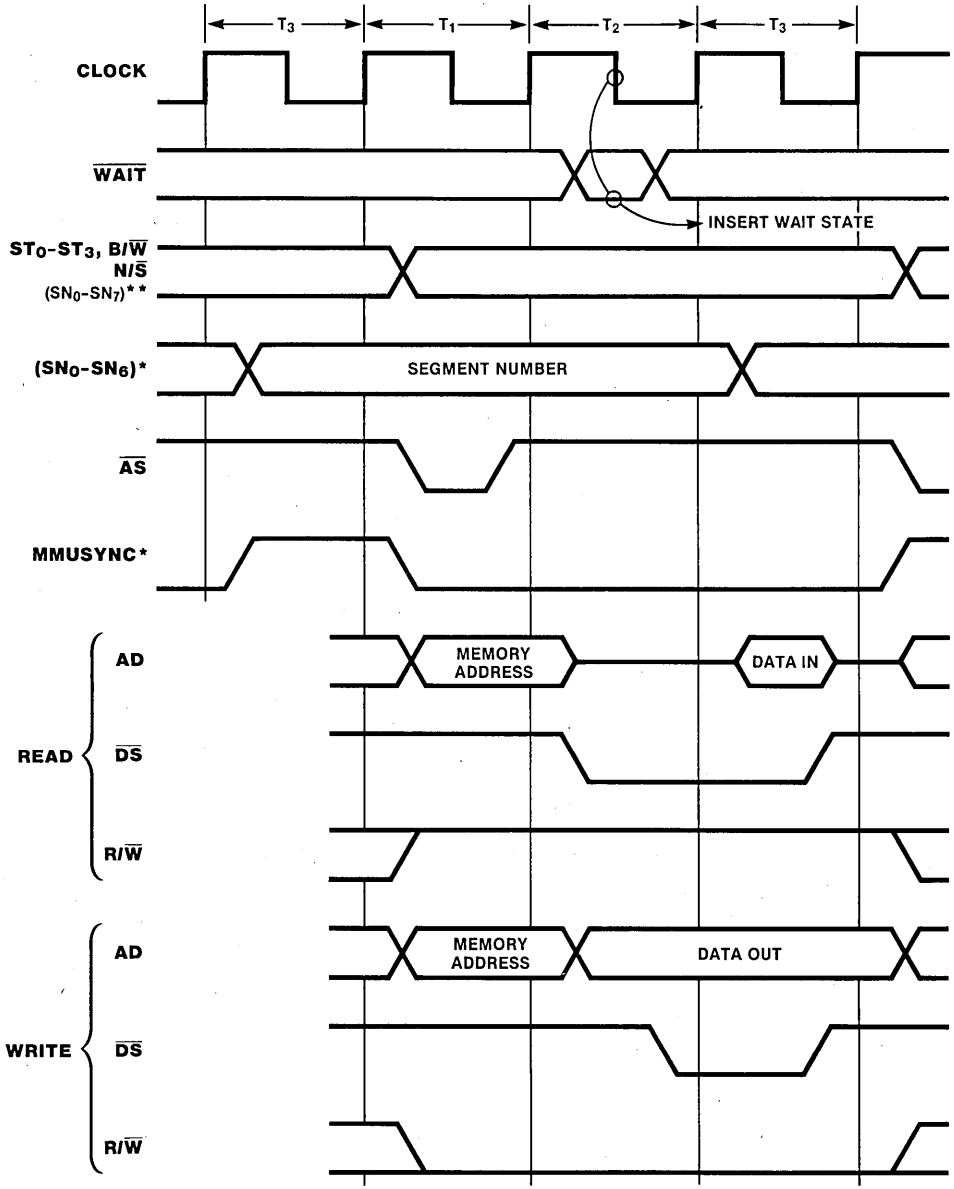


Figure 12. Bus Request and Acknowledge Timing



*For logical addressing only.
**For physical addressing only.

Figure 13. Flowthrough I/O Transaction Timing

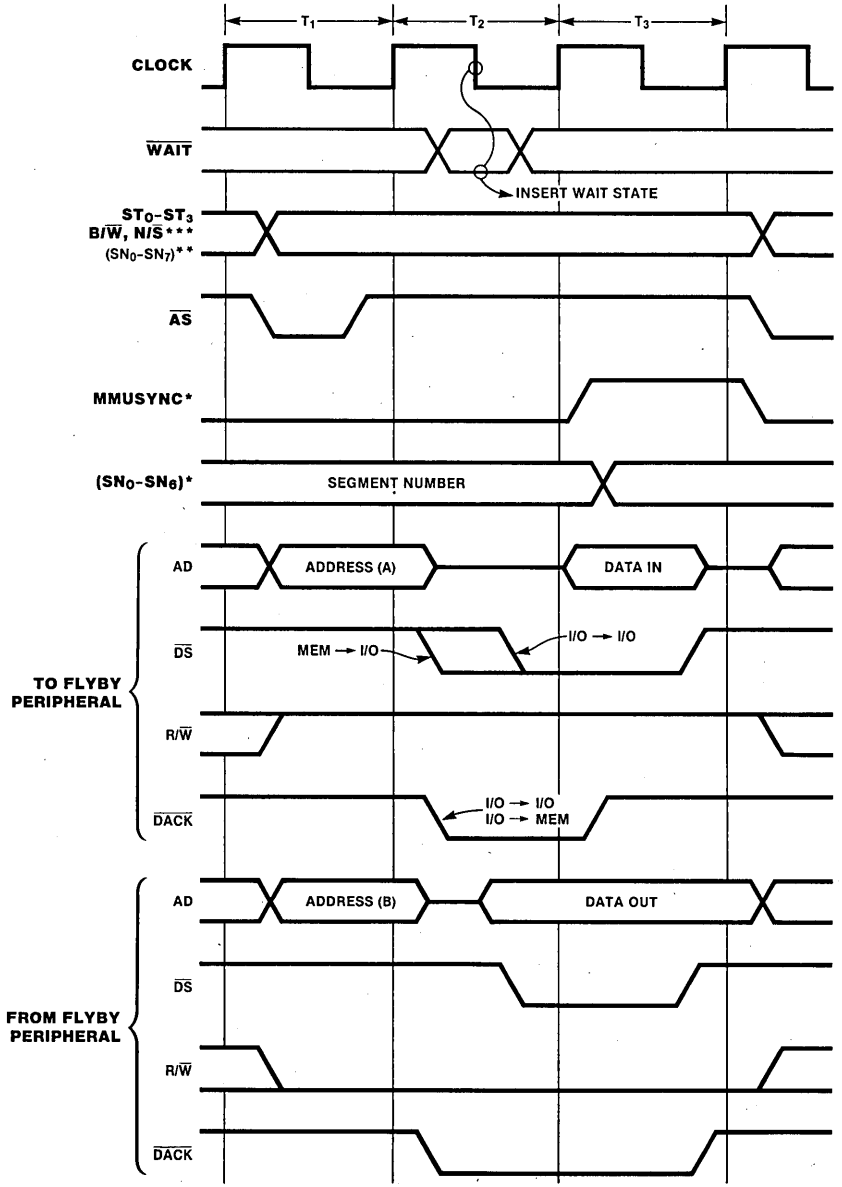


*For logical addressing only.
**For physical addressing only.

Figure 14. Flowthrough Memory Transaction Timing

Flyby Transactions

A Flyby operation is performed during three T-states. $\overline{AS}$ is pulsed during T_1 to signal the output of address information. R/$\overline{W}$ is High if the current ARA specifies source, and Low if the current ARB specifies destination. $\overline{DS}$ and $\overline{DACK}$ are driven active during T_2 to initiate the transfer, and driven inactive during T_3 to conclude the transfer. Wait states can be inserted between T_2 and T_3 to extend the active time to $\overline{DS}$ and $\overline{DACK}$. Flyby transaction timing is shown in Figure 15.



*Toggles for memory access in logical address space only.
**For physical addressing only.
***N/S will be low for I/O transactions.

(A) Address is current ARA
(B) Address is current ARB

Figure 15. Flyby Transaction Timing

$\overline{\text{DREQ}}$ Timing

The following section describes $\overline{\text{DREQ}}$ timing for various operations.

A High-to-Low transition of $\overline{\text{DREQ}}$ causes a single iteration of a DMA operation. A new transition can occur after the Low-to-High $\overline{\text{AS}}$ transition on the first memory or I/O access of the DMA iteration. Figure 16 shows the timing for a new transition to be applied and recognized to avoid giving up the bus at the end of the current iteration.

In Bus Hold mode, $\overline{\text{DREQ}}$ is sampled when a channel gains bus control. If $\overline{\text{DREQ}}$ is Low, an iteration of a DMA operation is performed. If $\overline{\text{DREQ}}$ is High, the channel retains bus control and continues to drive all bus control signals active or inactive, but performs no DMA operation.

In Demand mode during DMA operation, $\overline{\text{DREQ}}$ is sampled to determine whether the channel should perform another cycle or release the bus (Figure 17).

$\overline{\text{DREQ}}$ is sampled after each End of Chaining or Base-to-Current Reloading operation. If $\overline{\text{DREQ}}$ is active, the channel begins performing DMA operations immediately, without releasing the bus.

$\overline{\text{DACK}}$ Timing

During I/O and memory transactions, $\overline{\text{WAIT}}$ is sampled in the middle of T_2. If $\overline{\text{WAIT}}$ is High, and no programmable Wait states are selected, the DTC proceeds to T_3. Otherwise, one or more Wait states are inserted. $\overline{\text{WAIT}}$ is also sampled during T_{WA}. If $\overline{\text{WAIT}}$ is High the DTC proceeds to T_3, otherwise, additional Wait states are inserted. When both hardware and software Wait states are inserted, each $\overline{\text{WAIT}}$ time is sampled. A Low causes a hardware Wait state to be inserted in the next cycle. Software Wait state insertion is suspended until $\overline{\text{WAIT}}$ is High. Hardware Wait states can be inserted any time during the software Wait state sequence. $\overline{\text{DACK}}$ timing is shown in Figure 18.

$\overline{\text{EOP}}$ Timing

$\overline{\text{EOP}}$ is driven Low when a TC, MC, or $\overline{\text{EOP}}$ termination occurs. When a DMA operation has terminated, $\overline{\text{EOP}}$ is sampled on the falling edge of T_3 to determine if $\overline{\text{EOP}}$ has been driven Low. The generation of internal $\overline{\text{EOP}}$s and sampling of external $\overline{\text{EOP}}$s for Transfers-and-Searches follows the same timing used for Transfers. $\overline{\text{EOP}}$ timing is shown in Figure 19.

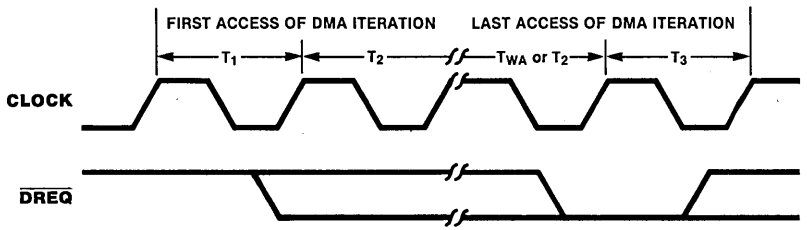
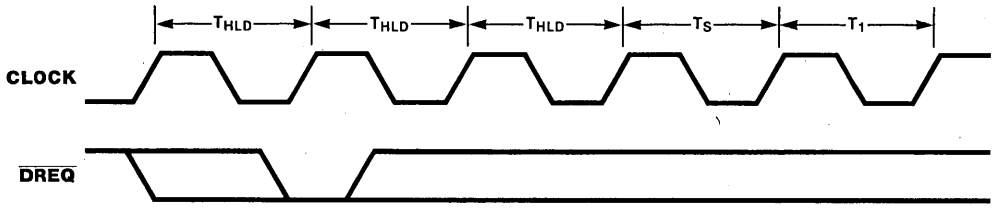
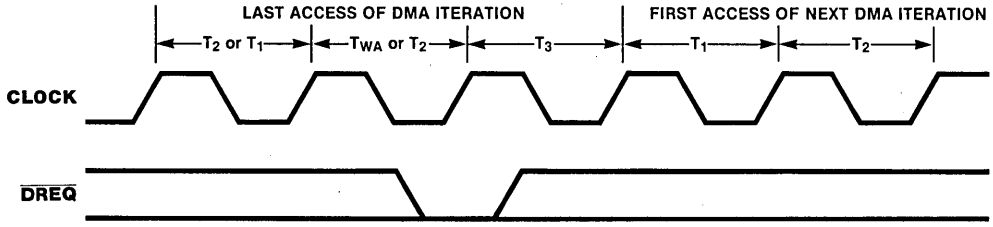


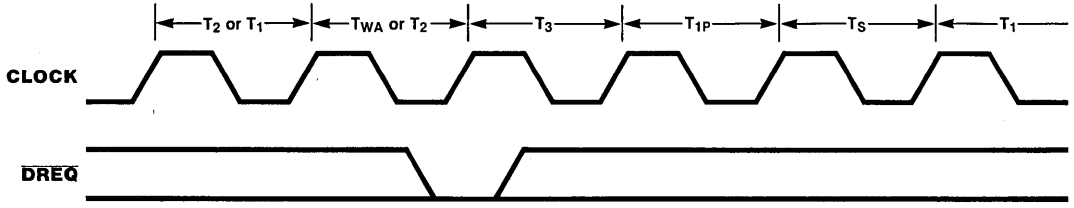
Figure 16. Sample $\overline{\text{DREQ}}$ During Single Transfer DMA Operations



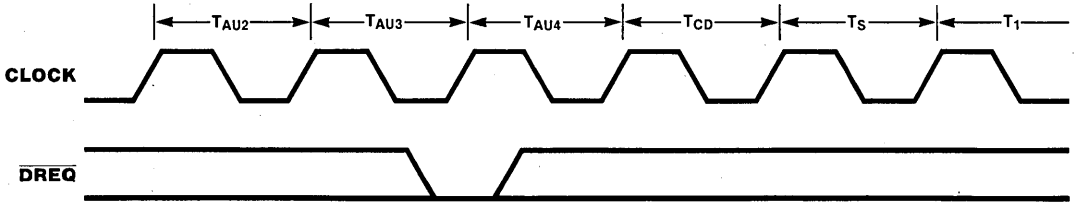
(A) Sampling $\overline{DREQ}$ While in Bus Hold Mode



(B) $\overline{DREQ}$ Sampling in Demand Mode During DMA Operations



(C) Sampling $\overline{DREQ}$ at the End of Chaining



(D) Sampling $\overline{DREQ}$ at End of Base-to-Current Reloading

Figure 17. $\overline{DREQ}$ Sampling in Demand Mode

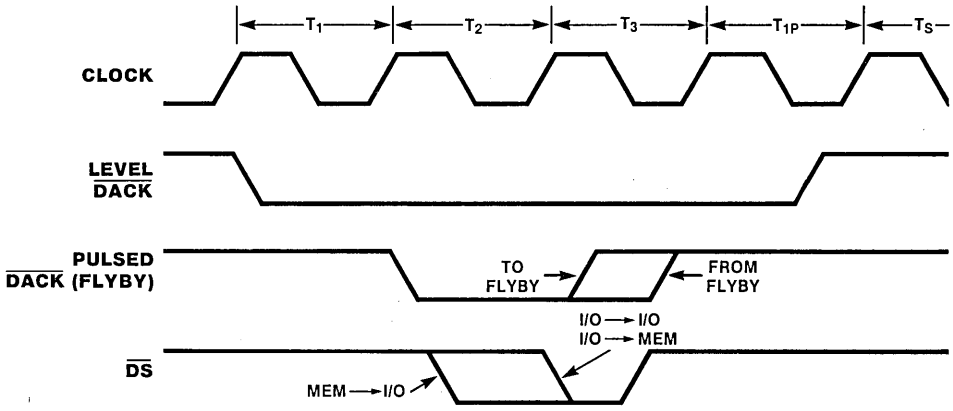


Figure 18. DACK Timing

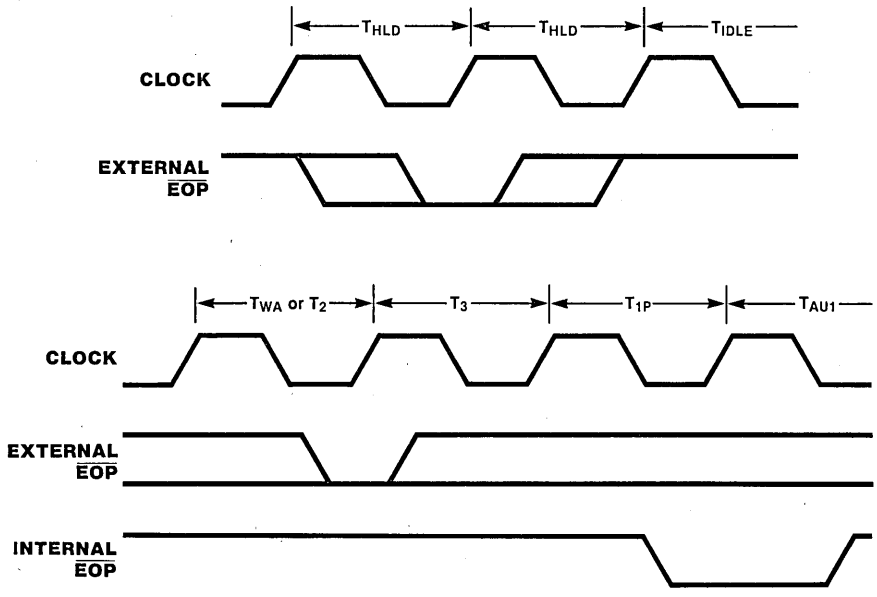
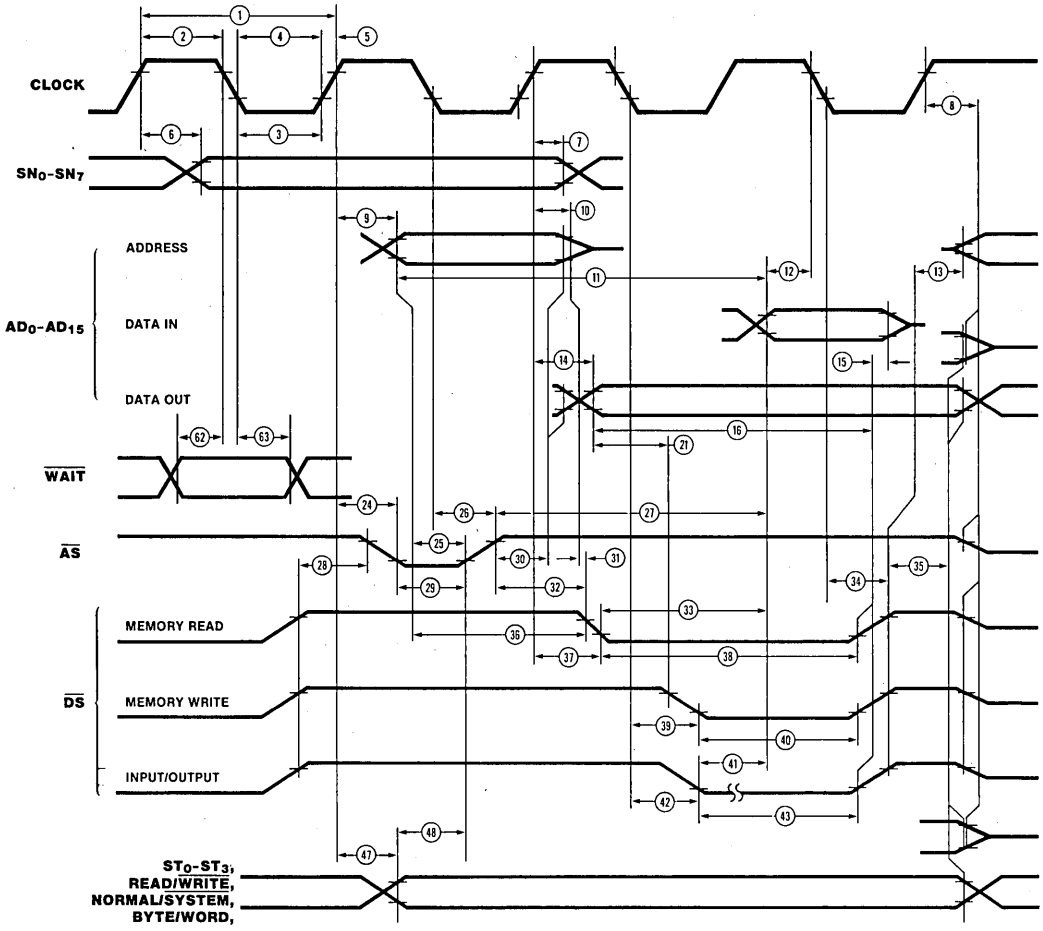


Figure 19. EOP Timing

ACTIVE STATE TIMING



AC CHARACTERISTICS†

Timing for DTC as Bus Master

Number	Symbol	Parameters	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max
1	TcC	Clock Cycle Time	250	2000	165	
2	TwCh	Clock Width (High)	105		70	
3	TwCl	Clock Width (Low)	105		70	
4	TfC	Clock Fall Time		20		10
5	TrC	Clock Rise Time		20		15
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50pf Load) Delay***		110		90
7	TdC(SNn)	Clock ↑ to Segment Number Valid Delay	20		10	
8	TdC(Bz)	Clock ↑ to Bus Float Delay		65		50
9	TdC(A)	Clock ↑ to Address Valid Delay		100		90
10	TdC(Az)	Clock ↑ to Address Float Delay		65		50
11	TdA(Dl)	Address Valid to Data In Required Valid Delay	400		305	
12	TsDI(C)	Data In to Clock ↓ Setup Time	20		15	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	80		45	
14	TdC(DO)	Clock ↑ to Data Out Valid Delay		100		90
15	ThDI(DS)	$\overline{DS}$ ↑ to Data In Hold Time	0		0	
16	TdDO(DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	230		200	
21	TdDO(SW)	Data Out Valid to $\overline{DS}$ ↓ (Write) Delay	55		35	
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		70		60
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		80		60
27	TdAS(Dl)	$\overline{AS}$ ↑ to Data In Required Valid Delay		300		220
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	75		35	
29	TwAS	$\overline{AS}$ Width (Low)	80		60	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Valid Delay	60		45	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0	
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Read) Delay	75		40	
33	TdDSR(Dl)	$\overline{DS}$ (Read) ↓ to Data In Required Valid Delay	165		155	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65
35	TdDS(DO)	$\overline{DS}$ ↑ to Data Out (Write Only) and Status Valid (Read and Write) Delay	85		45	
36	TdA(DSR)	Address Valid $\overline{DS}$ (Read) ↓ Delay	120		110	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		60		60
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275		185	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		60		60
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	160		150	
41	TdDSI(Dl)	$\overline{DS}$ (Input) ↓ to Data In Required Valid Delay		325		210
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		60		60
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	150*		150	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		80
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	60		35	
62	TsWT(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	20		20	
63	ThWT(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	30		30	
96	TdC(SNr)	Clock ↑ to SN7/MMUSYNC ↑ Delay**		110		110
97	TdC(SNf)	Clock ↑ to SN7/MMUSYNC ↓ Delay**	20	110		110

NOTES:

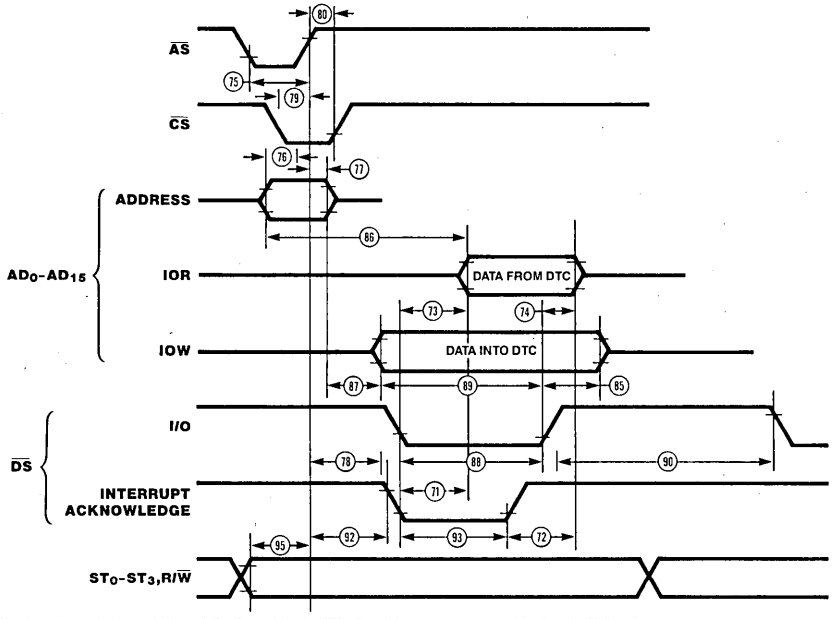
*Wait states should be inserted by programming a hardware when accessing slow peripherals.

**Logical Addressing only.

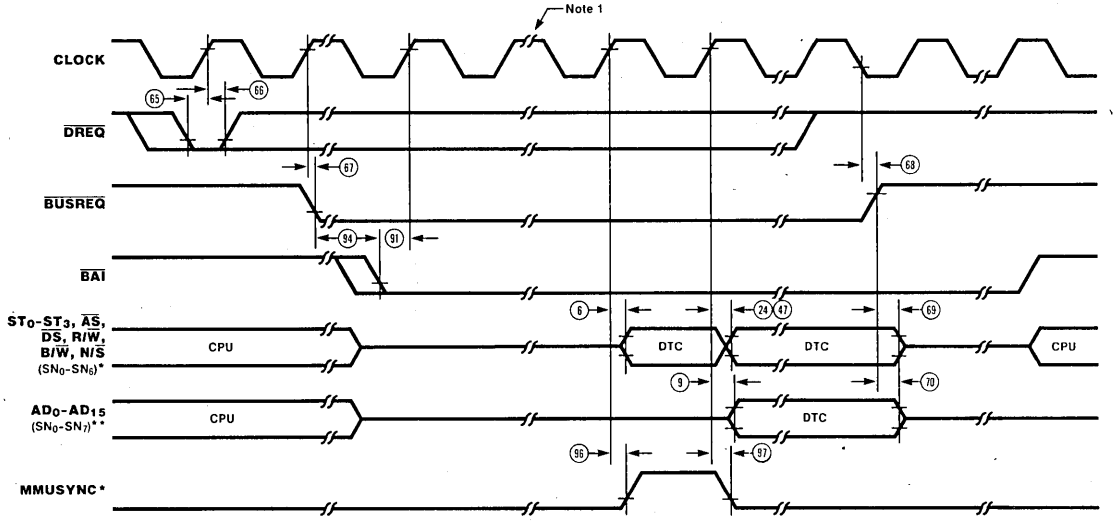
***130 ns max with Logical Addressing.

†Units in nanoseconds (ns).

INACTIVE STATE TIMING



BUS EXCHANGE TIMING



*For logical addressing only.
**For physical addressing only.
Note 1: The DTC will begin driving the bus on the clock cycle following the clock cycle in which the set-up parameters are met.

AC CHARACTERISTICS†

Timing for DTC as Bus Slave and CPU-DTC Bus Exchange

Number	Symbol	Parameters	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max
64	TwDRQ	$\overline{DREQ}$ Pulse Width (Single Transfer Mode)	20		20	
65	TsDRQ(C)	$\overline{DREQ}$ Valid to Clock ↑ Setup Time	60		50	
66	ThDRQ(C)	Clock ↑ to $\overline{DREQ}$ Valid Hold Time	20		20	
67	TdC(BRQf)	Clock ↑ to $\overline{BUSREQ}$ ↓ Delay		150		120
68	TdC(BRQr)	Clock ↓ to $\overline{BUSREQ}$ ↑ Delay		165		150
69	TdBRQ(BUSc)	$\overline{BUSREQ}$ ↑ to Control Bus Float Delay		140		110
70	TdBRQ(BUSd)	$\overline{BUSREQ}$ ↑ to AD Bus Float Delay		140		110
71	TdDSA(RDV)	$\overline{DS}$ ↓ (Acknowledge) to Data Output Valid Delay		135		120
72	TdDSA(RDZ)	$\overline{DS}$ ↑ (Acknowledge) to Data Output Float Delay		80		75
73	TdDSR(DOD)	$\overline{DS}$ ↓ (IOR) to Data Output Driven Delay		135		120
74	TdDSR(RDZ)	$\overline{DS}$ ↑ (IOR) to Data Output Float Delay		80		75
75	TwAS	$\overline{AS}$ Low Width	70		50	
76	TsA(AS)	Address Valid to $\overline{AS}$ ↑ Setup Time	30		10	
77	ThAS(Av)	$\overline{AS}$ ↑ to Address Valid Hold Time	50		40	
78	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay (I/O)	50		40	
79	TsCS(AS)	$\overline{CS}$ Valid to $\overline{AS}$ ↑ Setup Time	0		0	
80	ThCS(AS)	$\overline{AS}$ ↑ to $\overline{CS}$ Valid Hold Time	40		30	
81	TwAS(DS)	$\overline{AS}$ and $\overline{DS}$ Simultaneously Low Time (Reset)	3TcC		3TcC	
82	TdBAI(Az)	$\overline{BAI}$ ↑ to SN_0-SN_7, AD_0-AD_{15} Float Delay (Reset)		135		120
83	TdBAI(ST)	$\overline{BAI}$ ↑ to ST_0-ST_3, R/$\overline{W}$, B/$\overline{W}$, N/$\overline{S}$ Float Delay (Reset)		100		80
84	TdBAI(DS)	$\overline{BAI}$ ↑ to $\overline{DS}$, $\overline{AS}$ Float Delay (Reset)		100		85
85	TdDS(Dn)	$\overline{DS}$ ↑ (IOW) to Data Valid Hold Time	40		40	
86	TdAC(DRV)	Address Valid to Data (IOR) Required Valid Delay		540		345
87	TdAZ(DS)	Address Float to DS ↓ (IOR) Delay	0		0	
88	TwDS(IO)	$\overline{DS}$ (IO) Low Width	150*		150	
89	TsD(DS)	Data (IOW) Valid to $\overline{DS}$ ↑ Setup Time	40		40	
90	TrDS(W)	$\overline{DS}$ ↑ (IOW) to $\overline{DS}$ ↓ (IOW) (Write Recovery Time applies only for issuing Command)	4TcC		4TcC	
91	TsBAK(C)	$\overline{BAI}$ Valid to Clock ↑ Setup Time	60		50	
92	TdAS(DS)	$\overline{AS}$ ↑ to DS ↓ (ACK) Delay	100		100	
93	TwDS(AK)	$\overline{DS}$ (ACK) Low Width	150		150	
94	TdBRQ(BAI)	$\overline{BUSREQ}$ ↓ to $\overline{BAI}$ ↓ Required Delay	0		0	
95	TsS(AS)	Status Valid to $\overline{AS}$ ↑ Setup Time	40		0	
98	TdBAI (BAO)	$\overline{BAI}$ ↑, ↓ to $\overline{BAO}$ ↑, ↓ Delay		80		70
99	TdIEI(IEO)	IEI ↑, ↓ to IEO ↑, ↓ Delay		80		60

NOTES:
*2000 ns for reading slow-readable registers (worst case)
†Units in nanoseconds (ns).

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

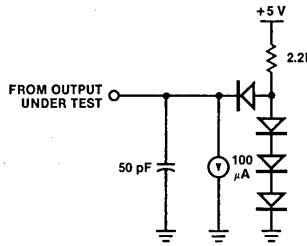
STANDARD TEST CONDITIONS

The DC characteristics and capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

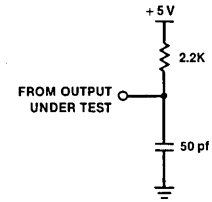
Standard conditions are as follows:

- $+4.75V \leq V_{CC} \leq +5.25V$
- GND = 0V
- T_A as specified in Ordering Information

All AC parameters assume a load capacitance of 50 pf max.



Standard Test Load



Open-Drain Test Load

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 μA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	μA	$0.4 \leq V_{IN} \leq V_{CC}$
I_{OL}	Output Leakage		±10	μA	$0.4 \leq V_{IN} \leq +V_{CC}$
I_{CC}	V_{CC} Supply Current		350	mA	T_A = 0°C

NOTE: V_{CC} = 5 V ± 5% unless otherwise specified.

CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C_{CLOCK}	Clock Capacitance		40	pf
C_{IN}	Input Capacitance		5	pf
C_{OUT}	Output Capacitance		10	pf

T_A = 25°C, f = 1 MHz.
Unmeasured pins returned to ground.

Zilog

PRELIMINARY
Product Specification

October 1988

Z16C20 CMOS Z-BUS ®GLU General Logic Unit

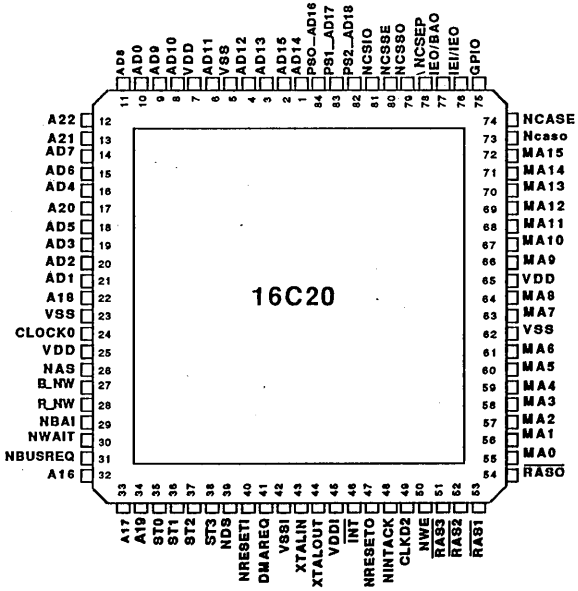
FEATURES:

- Directly Interfaces Z8000 CPU'S To Their Peripherals
- 8M Byte Address Range
- Eprom Interface
- Static RAM Interface
- Dynamic RAM Interface / Timing
- Eprom Address Accelerator
- DMA Controller
- Clock Generator
- Reset Circuitry
- Programmable Wait State Generators
- Input/Output Latch Controls
- General Purpose Timers
- Watchdog Timer
- Prioritized Interrupts
- Fully Programmable
- 10 MHz and 16 MHz Versions

GENERAL DESCRIPTION:

The Z16C20 CMOS GLU integrates into a single device the SSI and MSI logic typically required to interface a 16-bit Z8000 CPU in a system environment. It provides, to the user, an optimum system design solution in many areas. These areas include; cost, parts count, board area, reliability, and performance. This is achieved while simplifying hardware design and shortening design cycles. The Z16C20GLU supports up to 8M Byte of address space. It interfaces directly and simutaneously to EPROMs, SRAMs and DRAMs. By programming in their individual address space boundaries, the Z16C20 recognizes the type of memory being accessed and generates the appropriate controls and handshake signals required.

By anticipating code fetches, EPROM addresses may be generated by the GLU ahead of time. This allows slower and less expensive EPROMs to be used while maintaining high performance within the system. A DMA channel is provided for easy bootstrapping on power-up as well as for other DMA applications. In addition to that, the GLU has elaborate timing circuitry: on-chip clock generator with system clock and 1/2 system clock outputs (for slow peripherals), two general purpose, fully programmable, 16-bit timers, and a watchdog timer. It is also capable of automatically inserting wait-states when needed, prioritizing the interrupts and issuing synchronized resets.



PIN DESCRIPTIONS:

AD0-AD15. *Address/Data Bus* (I/O). Bidirectional multiplexed address and data bus from the CPU.

$\overline{AS}$. *Address strobe* (Bidirectional, Active Low).

B/$\overline{W}$. *Byte/Word* (Input, active High Byte). Selects byte or word transfers.

$\overline{BAI}$. *Bus Available Input* (input, active Low). $\overline{BAI}$ is an input received from the CPU or peripherals. Indicates to Z16C20 that bus has been 3-stated so that on-chip DMA can take control of the CPU bus.

$\overline{BUSREQ}$. *Bus Request* (Output, Active Low). Output to CPU requesting bus control for Z16C20 DMA.

$\overline{CASe-o}$. *Column Address Strobe even/odd* (Outputs, Active Low). Used for the DRAM interface. CASe selects the even bank and CASo selects the odd bank.

CLK. *System clock* (Output, active High). Capable of being programmed so that when certain peripherals are accessed, it will shift frequency to 1/2 of normal rate. This is necessary for peripherals using CLKD2 as their clock.

CLKD2. *Clock divide-by-2* (Output, active High). The system clock divided by 2. Used to drive peripherals that cannot operate at full system clock speed (CLK).

$\overline{CSE}$. *Chip Select Enable* (Output, active Low). Used for selecting the EPROMs.

$\overline{CSIO}$. *Chip Select I/O* (Output, Active Low). Used for selecting I/O space F3XX to FEXX.

$\overline{CSSe-o}$. *SRAM Chip Selects even/odd* (Outputs, active Low). Used to select even or odd SRAM banks.

$\overline{DMAREQ}$. *DMA Request* (Input, active Low). $\overline{DMAREQ}$ is an input received from a flyby peripheral requesting service from the on-chip DMA.

$\overline{DS}$. *Data strobe* (Bidirectional, Active Low)

IEI/IEO. *Interrupt Enable In* (Input, active High). Interrupt daisy chain control input. Optionally may be programmed to IEO.

IEO/BAO. *Interrupt Enable Output* (Output, Active High) Interrupt daisy chain control output. May be optionally programmed to BAO-*Bus Acknowledge Output.*

$\overline{INT}$. *Interrupt* (Bidirectional, Active Low) Interrupt to Z8000 CPU is generated by internal Z16C20 timers and DMA. The DMA may also optionally use this signal as a request to relinquish the bus back to the CPU so that the pending interrupt may be serviced. For example, the internal Z16C20 interrupt timer could be used to provide DRAM refresh during a DMA cycle if that was required.

$\overline{INTACK}$. *Interrupt Acknowledge* (Output, Active Low). $\overline{INTACK}$ is used for peripheral hand-shake during the interrupt acknowledge machine cycle. It is decoded from the four status inputs.

MA0-MA15. *Memory Address Bus* (Outputs, active High). Latched address lines for static memory interfaces. For DRAM interface, it will output 9 or 10 (256K or 1M) multiplexed address bits.

PS0-2/A16-18. *Peripheral Selects or Upper memory address lines* (Outputs, PS0-2 progammable active High or Low). Upper memory address lines are transfered SN0-2.PS0-2 may be programmed as peripheral selects or latch strobes. These selects will also serve as chip select for DMA peripherals during DMA operations. PS2 will be the default DMA chip select for the Power-On Bootstrap operation.

R/$\overline{W}$. Read/Write input.

$\overline{RAS0-3}$. *Row Address Strobes* (Outputs, Active Low). Used for the DRAM interface. Each RAS is capable of addressing 2 Mbyte of DRAM.

$\overline{RSTO}$. *Reset Output* (Output, Active low). Provides a synchronized system reset.

$\overline{RSTI}$. *Reset Input* (input, active Low). Reset input from external logic.

SN0-SN6. *Segment Number* (Inputs, active High). Upper address lines from the Z8000 CPU.

ST3-ST0. *Status* (Input, active High). Status lines from Z8000 CPU.

$\overline{WAIT}$. *Wait* signal to CPU. (Bidirectional, active Low.)

$\overline{WD/GPO}$. *Watch Dog or General Purpose Output* (Output, active Low). This output will be either the terminal count of the internal watchdog timer or a general purpose output, bit selectable.

$\overline{WE}$. *Write Enable* (Output, active Low). This signal is used to control memory and I/O writes. It is the combination of the "WRITE" signal and the "DATA STROBE".

XTALIN. *Crystal input.*

XTALOUT. *Crystal output.*

FUNCTIONAL DESCRIPTION:

Memory Address Configuration
Address lines are demultiplexed from the CPU address/data bus and latched internally, except for DRAM accesses, where they are appropriately multiplexed for the programmed DRAM size. The Z16C20 with the use of internal boundary registers, is capable of allocating memory space for EPROMs, SRAMS, 256K byte DRAMS and 1M byte DRAMS. These memory types may co-exist and are restricted to location in the 8M byte memory space as follows:

Eprom
EPROM must start at location 0 and is allocated in 32k byte blocks limited only by the memory size up to the boundary address. When other memory types overlap with EPROM, EPROM accesses will dominate unless EPROM is enabled for "code execution only" and in that case, DRAM or SRAM accesses would be performed for "DATA" reads and writes. The boundary register for EPROM is 8 bits. Multiple banks of EPROM require external decoding of the upper address bits and the EPROM chip select. The power-up default is that the full memory space is allocated to EPROM with no RAM overlap. If EPROM is not used, other memory devices can take its space.

Static Ram
SRAM space is defined by two 8 bit boundary registers. Minimum allocation, when SRAM is used, is 32K bytes. When the upper 8 address bits, A[23-16], fall within the range of the values set in the boundary registers, the SRAM will be automatically selected.

SRAM is completely disabled by programming bit 7 (most significant) of the upper boundary with "0" and bit 7 (most significant) of the lower boundary with "1". Since in normal operation the upper boundary would never be less that the lower boundary it is possible to use this condition as the disable. With the afore mentioned restrictions, SRAM may reside anywhere in the 8M byte memory space. Maximum allocation is 8M bytes. If SRAM is disabled, no memory space is allocated to it.

SRAM accesses take precedence over DRAM accesses when both are programmed to reside in the same space. However DRAM could be accessed by programming away the SRAM space. In this case SRAM would be ignored even though it physically exists. Multiple banks of SRAM require external decoding of the upper address bits and the SRAM chip selects.

Dram
Control signals and timing allow the GLU to connect directly to 256K bit or 1M bit DRAM devices. Both sizes may co-exist. "RAS ONLY" refresh for both memory sizes is provided.

DRAM space is allocated in banks of either 512K bytes or 2M bytes. Four boundary registers of 4 bits each allow for 4 banks without external decoding. Banks of 512K bytes may co-exist with banks of 2M bytes. A second 4 bit register exists to indicate which banks are 2M byte banks. Other than these boundary restrictions, DRAM may reside anywhere in the 8M byte space.

Additional banks of 512K bytes may be accessed by external decoding. The technique is to program a bank for 2M byte. SN4 and SN3 may then be externally decoded with the appropriate bank select, RAS0, RAS1, RAS2, or RAS3. In this manner up to four 512K byte RAS signals may be produced for each 2M byte RAS signal provided by the Z16C20. If two banks are programmed for the same space then the high order bank will be used for code accesses while the low order bank will be used for data accesses. This feature is only valid for bank pairs 0 with 1 and 2 with 3. Only DRAMS of the same size may be paired.

The DRAM RAS strobes are disabled by programming the block size to 2M byte and programming the least significant bit of the RAS boundary to a "1". It was possible to use this bit as a disable since it was not used for determination of the 2M byte space.

Memory Interface

Eprom Interface
The EPROM interface consists of a single active low chip select /CSE and latched address lines MA0-MA15. A bit is provided to completely disable EPROM accesses.

Accesses are made to EPROM whenever the block address for a code transaction is less than that contained in the EPROM boundary register. Likewise "memory data reads" access EPROM in this space unless programmed to access RAM.

EPROM mode is enabled on power-up/reset and the boundary register defaults to hex FF.

Eprom Address Accelerator
The EPROM address accelerator may improve EPROM performance by as much as 25% depending on number of inserted wait states. This means slower and lower cost EPROMS can be used in higher performance applications.

The system takes advantage of the accelerator when operating in a sequential addressing mode of EPROM. The accelerator works by incrementing the active address (all 23 bits) so that the next address is available within the Z16C20before the next address has arrived from the CPU and before the present memory cycle is complete. This computed address can be placed on the memory address bus at the middle of T3 as soon as the previous transaction has ended.

The next address from the CPU will not arrive until the middle of T1 of the next cycle at which time it will be compared

(again 23 bits) against the incremented address that is already present on the bus. Should the comparison fail, the address bus will be updated with the correct address and a wait-state will be generated. The new address will be latched and incremented in anticipation of the next sequential EPROM access.

The EPROM accelerator maintains its advantage even when accessing different types of memory or I/O. All that is required is that the next executed address be sequential. This feature is bit enabled, with default being the off state.

Sram Interface
The static RAM interface consists of three signals $\overline{CSSe}$, $\overline{CSSo}$, and $\overline{WE}$. The two chip selects (even and odd) allow the high and low byte to be individually accessed.

The $\overline{WE}$ output is a combination of $\overline{DS}$ AND R/$\overline{W}$.

$\overline{CSSe}$ decodes B/$\overline{W}$ and A0.

$\overline{CSSo}$ decodes B/$\overline{W}$ and $\overline{A0}$.

The static interface is selected by programming the appropriate boundary registers.

Dram Interface
Six control outputs $\overline{RAS0}$, $\overline{RAS1}$, $\overline{RAS2}$, $\overline{RAS3}$, $\overline{CASe}$, and $\overline{CASo}$ and multiplexed address lines MA0-MA9 are used to directly drive DRAM memories. These signals provide sufficient control to allow direct addressing for 8 megabytes of DRAM memory.

The Row/Column address generation is as follows:

Z16C20 MA Signal	256K Bit Row/Col	1M Bit Row/Col
MA0	A1/A10	A1/A11
MA1	A2/A11	A2/A12
MA2	A3/A12	A3/A13
MA3	A4/A13	A4/A14
MA4	A5/A14	A5/A15
MA5	A6/A15	A6/SN0
MA6	A7/SN0	A7/SN1
MA7	A8/SN1	A8/SN2
MA8	A9/SN2	A9/SN3
MA9	NA	A10/SN4

"RAS ONLY" mode of refresh is supported. The refresh is CPU driven and the refresh address is simply passed through to the memory bus appropriately timed to the RAS signal. Since the Z8000 provides only 9 bits of refresh address the Z16C20 generates the 10th bit.

Peripheral Interface
The Z16C20 contains provision for allowing a Z8000 or Z280 to operate with slower peripherals operating at one-half the system frequency.

A clock output, CLKD2, that is the system clock devided by 2, is provided. This clock will drive the clock input of low performance peripherals. An access to any of these peripherals will cause the Z16C20 to slow the system clock to 1/2 clock frequency at the leading edge of T2. T1 is not stretched because status is not available soon enough in T1 to make the decision to hold T1. This should not be a problem if the peripheral was able to operate with the full frequency address strobe during times when it was not being accessed. On power up, CLK is defaulted to 1/2 the frequency (CLKD2).

It should be noted that peripherals do have to recognize address strobe even when they are not being accessed. This is because the peripherals use the $\overline{AS}$ to clock the Interrupt Pending bits with.

If the full frequency "address strobe width" is not sufficient, then T1 can be stretched continuously by 1/2 clock period. This should provide sufficient width for the address strobe. If T1 stretching is enabled, then the first state of any wait state sequence will be shortened to 1/2 clock period. Any additional wait states after the first one, will be the normal full clock period in length.

All interrupt acknowledge cycles will also be slowed to 1/2 if when the low performance peripheral interface is enabled.

If $\overline{DMAREQ}$ is held active during the power-up sequence, then $\overline{PS2}$ will be DMA chip select.

Dma Controller
The DMA circuit contains a 23 bit up/down address counter and a 16 bit transaction length counter. The address counter may be incremented/decremented by 1 or 2 or 4 while the transaction counter is decremented by 1. This allows the DMA to transfer 65K bytes, words, or longwords (with external hardware support) to anywhere in the 8 Mbyte memory space. In the case of long word transfer the B/$\overline{W}$ pin will indicate L/$\overline{W}$.

Transfers are flyby mode only and bidirectional between a flyby peripheral and a memory device. The memory device can be DRAM, SRAM, or EPROM (read only). There is the restriction that the DMA addresses be within range of the programmed Z16C20 memory boundary registers.

The DMA will support byte or word transfers. When the transfer is byte to memory the transfer should be on the lower address/data bus. The Z16C20 will replicate this lower byte to the upper byte so that it may be loaded into

the proper memory location. When the byte transfer is from the memory to I/O the Z16C20 will replicate the upper bus on the lower bus when the address is even. This function is bit programmable and is normally defaulted to "off" after power-up unless the bootstrap option is to be selected In this case it will be defaulted to "on" and the direction of transfer will be I/O to memory.

The external DMA interface requires the following signals. Outputs are MA[15:0]. Inputs are $\overline{\text{DMAREQ}}$ and $\overline{\text{BUSACK}}$. Bidirectionals are AD[15:0}, SN[6:0], $\overline{\text{BUSREQ}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, R/$\overline{\text{W}}$, B/$\overline{\text{W}}$, and ST0:3.

The DMA circuit will acknowledge the DMA peripheral through the Peripheral select outputs, PS0_A16, PS1/A17, and PS2/A18. Any one of these three outputs may be programmed to select the DMA peripheral. If the bootstrap load mode is enabled on power-up PS2/A18 will select the DMA peripheral. During DMA operations, the DMA requesting device must invert its R/$\overline{\text{W}}$ input after detecting that that $\overline{\text{BUSACK}}$ from the processor has gone active.

DMA transactions can be made in either "BURST", "ALTERNATE CYCLE", or "BUS RELINQUISH ON VECTORED INTERRUPT" modes. This third mode in conjunction with the internal interrupt timer is useful in servicing DRAM refresh during long bursts.

Burst mode means that the DMA will not release $\overline{\text{BUSREQ}}$ until a full block has been transferred. unless, of course, the interrupt mode has been enabled. When in "Alternate Cycle" mode, the DMA releases bus request upon completion of each DMA byte or word transfer and then immediately regenerates BUSREQ upon detecting /AS, the CPU address strobe.

Bit enabled interrupts on "DMA START" and "DMA FINISH" are available. Interrupts are prioritized internally and externally through the ZBUS daisy chain arrangement and provide a vector upon receiving interrupt acknowledge.

Zero to three wait states are programmable. When the two devices involved in a DMA transfer differ in the number of wait states programmed, the transaction will accommodate the slower device.

Zero to three delay states for inter-transaction padding are also programmable. This allows for delaying the next access to the flyby peripheral if there has not been sufficient time for recovery from the previous transaction.

Software reset is provided.

The DMA circuit can also be made to operate in a bootstrap mode if $\overline{\text{DMAREQ}}$ is held active during the power-on-reset timing cycle. This mode enables the DMA to program itself directly from the flyby peripheral. This is accomplished by the DMA generating $\overline{\text{BUSREQ}}$ to the CPU during reset so that the CPU will stay off the bus. It then addresses its internal registers and generates the appropriate controls so that the flyby peripheral can write the bus. After the registers have all been loaded the "start address register" and the "transaction count register" are both loaded to their respective counters and a normal DMA cycle is started. This allows the Z16C20 to perform a bootstrap load operation on power-up.

Timers

The Z16C20 has two 16 bit timers each with its own eight bit prescaler.

Timer A serves the function of an interrupt timer. The counter's 16 bit time constant is written to a latch that is also readable by the CPU. The latch contents are loaded to the counter on the "GO" command and then down counted to a terminal count that sets the interrupt if enabled. Timer A has priority over Timer B. The counter also has a auto-reload mode where the time constant is reloaded to the counter upon terminal count. Interrupts in this mode will be missed if interrupt service is not provided within the timer period. The contents of the 16 bit counter are also readable by the CPU. Timer A is disabled on power-up/reset.

Timer B serves the function of an interrupt timer or a watchdog timer. The counter is physically identical to Timer A and as an interrupt timer also behaves identically to timer A except that the Timer A interrupt has priority over Timer B. Once the time constant latch has been loaded, a "GO" command along with a "Timer B Enable" will load the counter and immediately start a count down to zero. If the terminal count is reached, either a $\overline{\text{WDOG}}$ output or a system reset is generated. The option is bit programmable. The requirement here is to service the counter within the timer period so the counter is not able to terminate. The $\overline{\text{WDOG}}$ output is bit enabled. If disabled, the output then becomes a programmable general purpose output. Timer B is disabled on power-up/reset so that the general purpose output is enabled with default value of logic "1".

Interrupts And Daisy Chain

Interrupts are generated by the interrupt timers, the DMA, external pin and by writing to a predefined I/O address. The DMA provides for two interrupts: one on DMA start and the other on DMA finish. The HARD interrupt shares function with the "WATCHDOG TERMINAL OUTPUT PIN" and is bit programmed as an input or an output. A high to low transition on this pin, when enabled as an interrupt, will generate an interrupt and provide the appropriate vector on interrupt acknowledge. Likewise, the Soft interrupt will generate an interrupt whenever a write is made to Z16C20 I/O address of FFE0. These interrupts are highest priority internal Z16C20 interrupts and neither can be interrupted

by another internal interrupt while under service. All interrupts are bit enabled and are programmable as to their respective priority. Each interrupt also has an associated vector that is put on the Address/Data bus during interrupt acknowledge. The first 13 bits of the vector are common to all four interrupts and are user programmable. Bit 0 is always zero. Bits 1 and 2 are determined by the source of the interrupt.

DEVICE	VECTOR
Timer A	XXXXXXXXXXX000
Timer B	XXXXXXXXXXX010
DMA start	XXXXXXXXXXX100
DMA	XXXXXXXXXXX110
HARD	XXXXXXXXXX1000
SOFT	XXXXXXXXXX1010

"XXXXXXXXXXX" is the common programmed value. Any value is appropriate but is the same value for all four interrupts. The least significant four bits are supplied by the Z16C20.

The Z8000 daisy chain is also supported through three pins, IEI, IEO, and $\overline{\text{INTACK}}$ (decoded from status lines and output as $\overline{\text{INTACK}}$ from Z16C20).

Interrupt vectors will be enabled on to data bus only if the IEI is active. If an interrupt is active IEO is driven Low disabling any lower priority interrupts from generating a vector during interrupt acknowledge.

Clock Generator
This circuit consists of signals XTALIN, XTALOUT, and CLK and CLKD2. The oscillator input will accept either a series resonant crystal, a ceramic resonator, or a TTL level signal. The CLK output, which is the system clock, has sufficient drive capability for the CPU clock requirements.

CLKD2 is a clock output that is 1/2 the system clock (CLK) frequency. CLKD2 is used to clock peripherals that can not operate at full system clock speed. The system clock can be slowed to 1/2 the normal frequency. This occurs when accessing peripherals that are operating on CLKD2, after programming into the Z16C20 their address space.This feature is bit programmable. The power-up/reset default is that no peripherals are clocked by CLKD2. After the Z16C20 is initialized, the clock is corrected to the programmed frequency for the various peripherals.

The oscillator's maximum frequency is 32 Mhz. It has an internal divide by two for the CPU clock (CLK) and an additional divide by two for the CLKD2.

Reset Synchronization Circuitry
The reset circuit has two signals. $\overline{\text{RSTI}}$ (input) and $\overline{\text{RSTO}}$ (output). The $\overline{\text{RSTO}}$ is synchronized with CLK to meet the CPU requirements.

RSTO is activated four ways:

a)Power On Reset,When power is first applied, circuitry will hold $\overline{\text{RSTO}}$ active for 30ms.

b)Watchdog timer times out, a reset will be issued if that function is enabled.

c)$\overline{\text{RSTI}}$ will activate $\overline{\text{RSTO}}$ for at least 16 clock cycles or for as long as $\overline{\text{RSTI}}$ is held active. This input is edge triggered but has normal TTL levels.

d) There are two levels of software reset: One does not activate $\overline{\text{RSTO}}$, thus resetting only the internal Z16C20 and the other activates the internal reset as well as $\overline{\text{RSTO}}$.

Wait State Generators
Programmable zero to three additional wait states can be generated for EPROM, RAM, I/O, DMA and Interrupt Acknowledge. The DRAM wait state generator adds 1/4 clock period to the address hold time after RAS and adds 1/4 clock period from the column address until CAS is generated. This allows additional time for the address bus to be driven valid as well as allow more liberal timing for the RAS before CAS specification. The additional 1/2 clock period will be added to the CAS access time.

The $\overline{\text{WAIT}}$ I/O consists of an inverter circuit of which the pull-up device has high impedance, approximately 2K. this feature yields additional power savings since it can directly drive a cmos gateand by elliminating the need for a power consuming pull-up resistors.

Wait states are defaulted to maximum delay of three wait states upon reset.

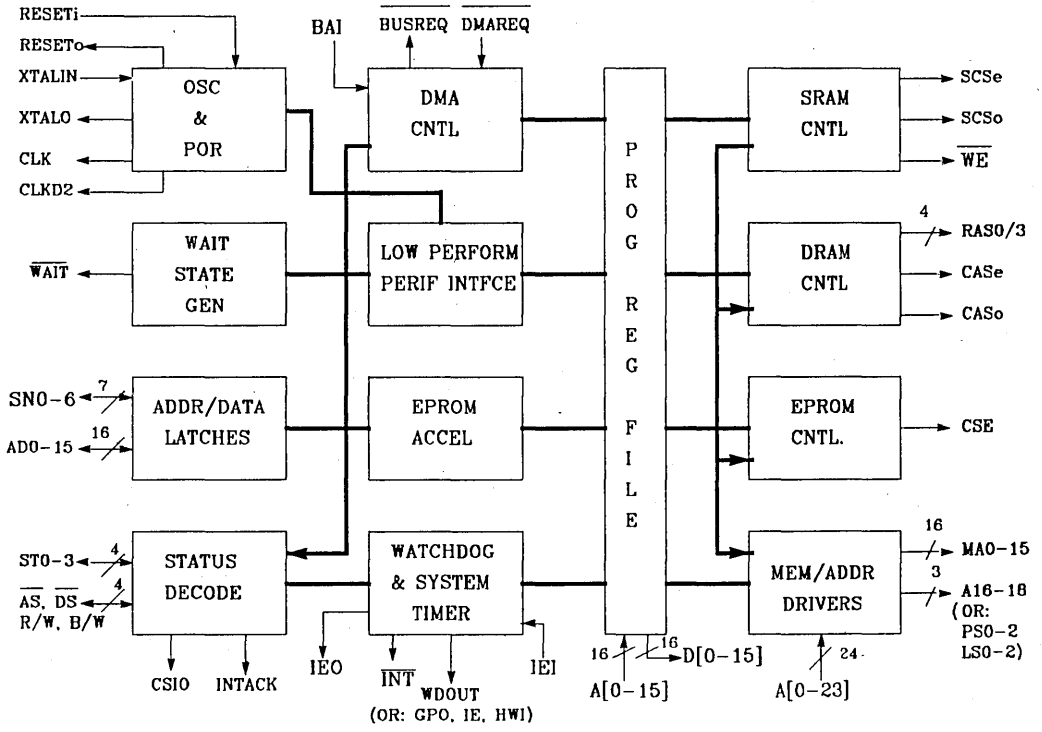
Peripheral Interface
Peripheral chip selects, $\overline{\text{CSIO}}$ and $\overline{\text{INTACK}}$ signals are decoded from the status lines. Any one of the three peripheral chip selects may be programmed to select the DMA peripheral during DMA operation. $\overline{\text{CSIO}}$ is active in the normal I/O space and pertains to that space not specified by other Z16C20 I/O chip selects.

The peripheral selects PSn are programmable for active high or low. These selects are multiplexed with the upper address outputs.

I/O SELECT	ADDRESS
PCS0	F0XX
PCS1	F1XX
PCS2	F2XX
CSIO	F3XX to FEXX
Z16C20 Register Space	FFXX

Internal registers are word wide and read/write with the exception of the count values of Timer A, Timer B, DMA transaction counter, and DMA address counter which are read only.

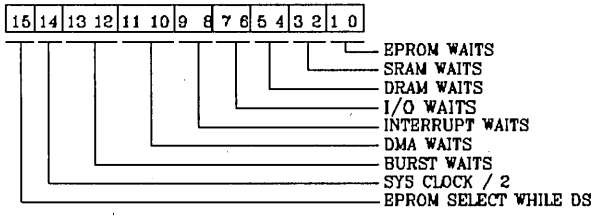
Z16C20 (GLU) BLOCK DIAGRAM



REGISTER DESCRIPTIONS

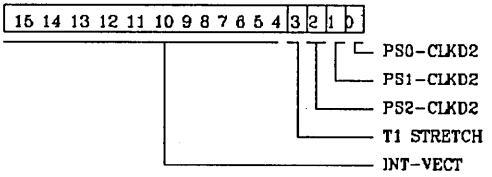
WAIT STATE SELECTS REGISTER

FFEE



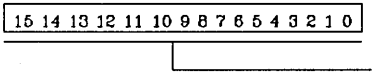
INTERRUPT VECTOR/PORT TIMING REGISTER

FFF3



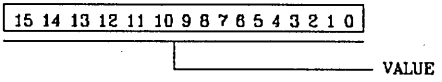
DMA TRANSACTION COUNTER – READ ONLY REGISTER

FFEF



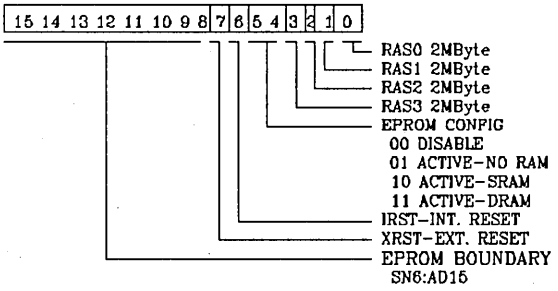
COUNTER B VALUE – READ ONLY REGISTER

FFF4



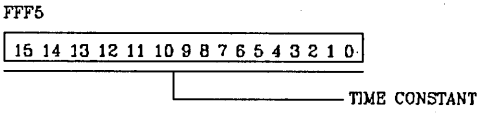
BOUNDARY REGISTER 1

FFF0



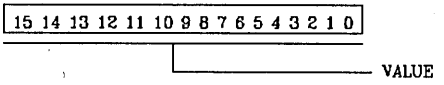
TIMER A – TIME CONSTANT REGISTER

FFF5



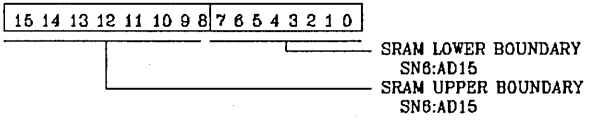
COUNTER A VALUE – READ ONLY REGISTER

FFF6



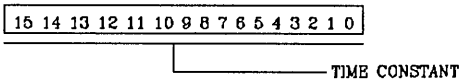
BOUNDARY REGISTER 2

FFF1



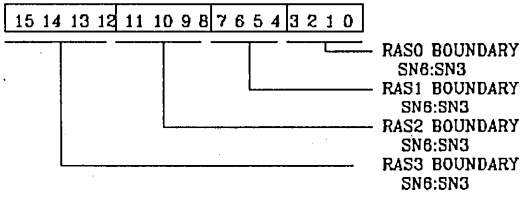
TIMER B – TIME CONSTANT REGISTER

FFF7



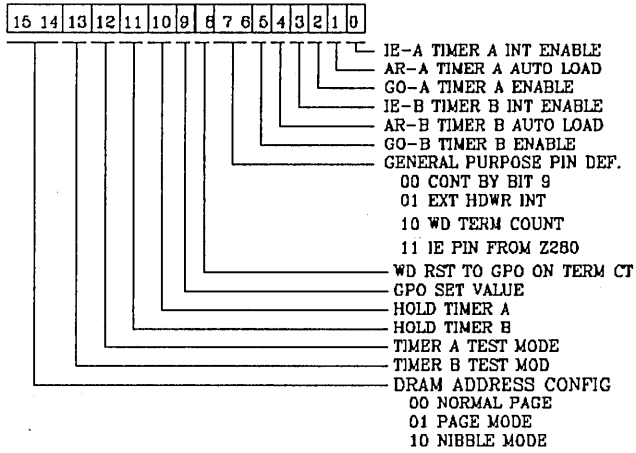
BOUNDARY REGISTER 3

FFF2



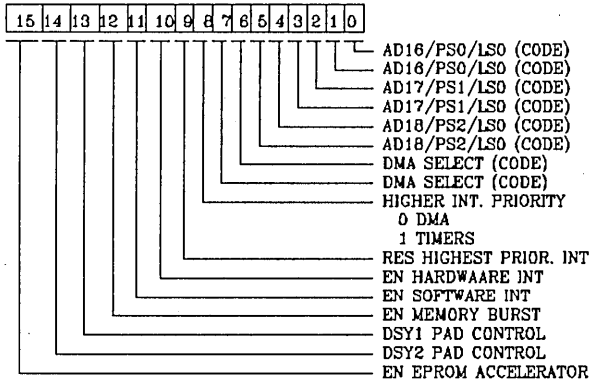
TIMER CONTROL REGISTER

FFF8



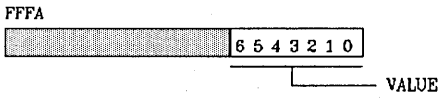
EXTENDED ADDRESS AND PERIPHERAL SELECT REGISTER

FFF9



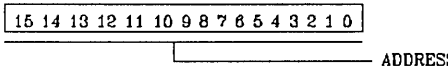
DMA SEGMENT NUMBER – READ ONLY REGISTER

FFFA



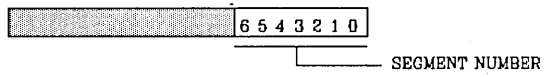
DMA ADDRESS CONSTANT

FFFB



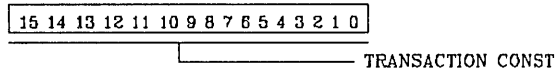
DMA SEGMENT NUMBER CONSTANT

FFFC



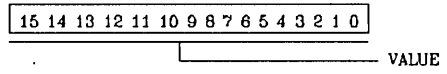
DMA TRANSACTION COUNTER CONSTANT

FFFD



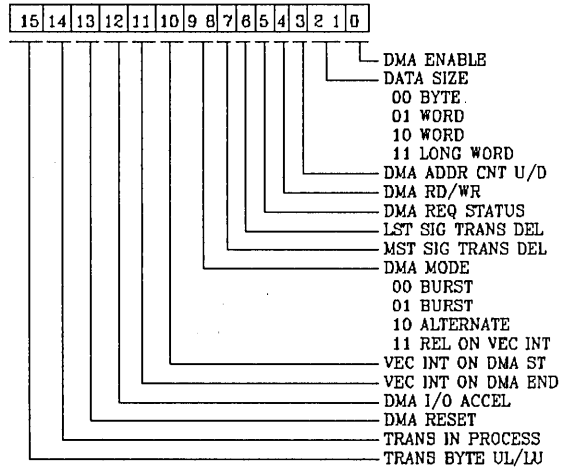
DMA ADDRESS COUNT – READ ONLY REGISTER

FFFE



DMA CONTROL

FFFF



ABSOLUTE MAXIMUM RATINGS:

Voltage on Vcc with respect to Vss
. . . . . -0.3V to +7.0V
Voltages on all inputs with respect to Vss
. . . . . -0.3V to Vcc+0.3V
Operating Ambient Temperature
. . . . . See Ordering Information
Storage Temperature
. . . . . -65°C to + 150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

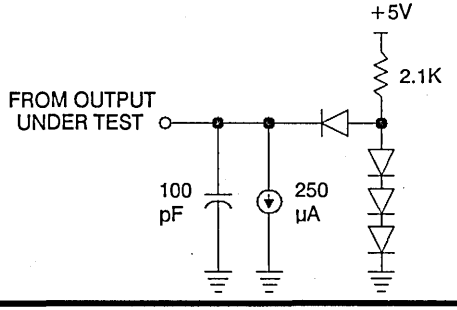
STANDARD TEST CONDITIONS:

The DC Characteristics and Capacitance sections below apply to the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows in to the referenced pin.

Available operating temperatures ranges are:
S = 0°C to + 70°C
E = -40°C to + 85°C
M = -55°C to + 125°C
Voltage Supply Range: + 5.0V ± 10%

All AC parameters assume a load capacitance of 100 pF. Add 10 ns delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus and 100 pF for the address and control lines. AC timing measurements are referenced to 1.5 volts (except for CLOCK, which is referenced to the 10% and 90% points).

The Ordering Information section lists temperature ranges and product numbers. Package drawings are in the Package Information section. Refer to the Literature List for additional documentation.



DC CHARACTERISTICS:

Symbol	Parameter	min	max	Unit	Condition
V_{IHC}	Input Clock High Voltage	V_{CC}-1.0	V_{CC}+0.3	V	Driven by Ext. Clock
V_{ILC}	Input Clock Low Voltage	-0.3	1.0	V	Driven by Ext. Clock
V_{IH}	Input High Voltage	2.2	V_{CC}+0.3	V	
V_{IL}	Input Low Votlage	-0.3	0.8	V	
V_{OHC}	Output Clock High Voltage	V_{CC}-0.6		V	
V_{OH}	Output High Voltage	2.4			I_{OH}=-250µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL}=2.0mA
V_{OLW}	Output Low Voltage (Wait)		0.5	V	I_{OL}=5.0mA
I_{IL}	Input Leakage Current		±10	µA	
I_{CC}	Power Supply Current		30	mA	f=8.0MHz
			40	mA	f=10.0MHz
					V_{CC}=5V
					V_{IH}=V_{CC}-0.2V
					V_{IL}=0.2V
C_I	Input Capacitance		5	pF	
C_O	Output Capacitance		10	pF	

October 1988

Z80C30 CMOS Z–BUS SCC/ Z85C30 CMOS SCC Serial Communications Controller

Features

- Low power CMOS.
- Pin compatible to NMOS versions.
- Two independent, 0 to 2.5M bit/second, full–duplex channels, each with a separate crystal oscillator, baud rate generator, and Digital Phase–Locked Loop for clock recovery.
- Multi–protocol operation under program control; programmable for NRZ, NRZI, or FM data encoding.
- Asynchronous mode with five to eight bits and one, one and one–half, or two stop bits per character, programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Synchronous mode with internal or external character synchronization on one or two synchronous characters and CRC generation and checking with CRC–16 or CRC–CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame–level control, automatic zero insertion and deletion, 1–field residue handling, abort generation and detection, CRC generation and checking, and SDLC Loop mode operation.
- Local Loopback and Auto Echo modes.
- Supports T1 digital trunk.
- Enhanced DMA support
  - 10 X 19–bit status FIFO
  - 14–bit byte counter

General Description

The Z80C30/Z85C30 CMOS SCC Serial Communications Controller is a CMOS version of the industry standard NMOS SCC. It is a dual channel, multi–protocol data communications peripheral that easily interfaces to CPU's with either multiplexed or non–multiplexed address/data buses. The advanced CMOS process offers lower power consumption, higher performance, and superior noise immunity. The programming flexibility of the internal registers allows the SCC to be configured to satisfy a wide variety of serial communications applications. The many on–chip features such as baud rate generators, digital phase locked loops, and crystal oscillators dramatically reduce the need for external logic. Additional features including a 10 X 19–bit status FIFO and 14–bit byte counter were added to support high speed SDLC transers using DMA controllers.

The SCC handles asynchronous formats, synchronous byte–oriented protocols such as IBM Bisync, and synchronous bit–oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives, etc.).

The device can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general–purpose I/O.

The daisy–chain interrupt hierarchy is also supported––as is standard for Zilog peripheral components.

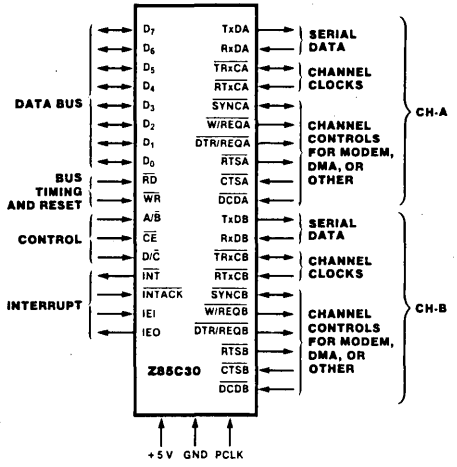


Figure 1a. Pin Functions, Z85C30

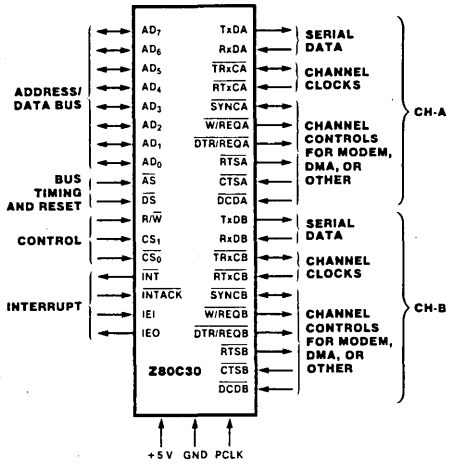


Figure 1b. Pin Functions, Z80C30

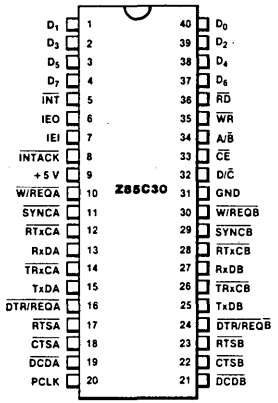


Figure 2a. DIP Pin Assignments, Z85C30

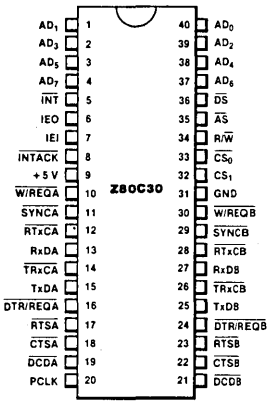


Figure 2b. DIP Pin Assignments, Z80C30

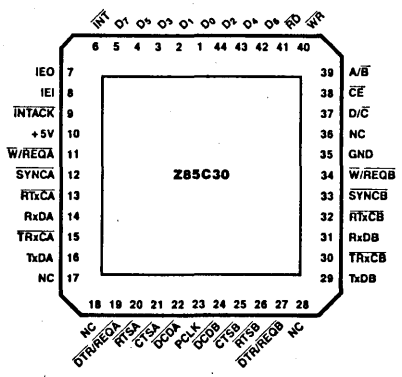


Figure 2c. Chip Carrier Pin Assignments, Z85C30

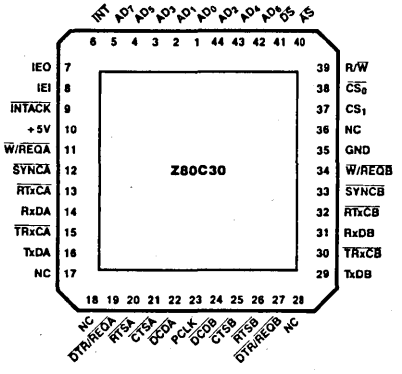


Figure 2d. Chip Carrier Pin Assignments, Z80C30

Pin Description

The following section describes the pin functions common to the Z85C30 and the Z80C30. Figures 1 and 2 detail the respective pin functions and pin assignments.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general–purpose inputs. Both inputs are Schmitt–trigger buffered to accommodate slow rise–time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general–purpose input pins. Both pins are Schmitt–trigger buffered to accommodate slow rise–time signals. The SCC detects pulses on these pins and can and can interrupt the CPU on both logic level transitions.

$\overline{DTR}/\overline{REQA}$, $\overline{DTR}/\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy–chain when there is more than one interrupt driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open–drain, active Low). This signal is activated when the SCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When $\overline{RD}$ or $\overline{DS}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of PCLK.

PCLK. *Clock* (input). This is the master SCC clock used to synchronize internal signals. PCLK is a TTL level signal. PCLK is not required to have any phase relationship with the master system clock.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RTxCA}$ $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase–Locked Loop. These pins can also be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 11) is set, the $\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general–purpose outputs.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 10) but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{SYNC}$ must be driven Low two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{SYNC}$.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{TRxCA}$, $\overline{TRxCB}$. *Transmit/Receive* Clocks (inputs or outputs, active Low). These pins can be programmed

in several different modes of operation. TRxC may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase–Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{W}/\overline{REQA}$, $\overline{W}/\overline{REQB}$. *Wait/Request* (outputs, open–drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual–purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait.

Z85C30

A/$\overline{B}$. *Channel A/Channel B* (input). This signal selects the channel in which the read or write operation occurs.

$\overline{CE}$. *Chip Enable* (input, active Low). This signal selects the SCC for a read or write operation.

D_0–D_7. *Data Bus* (bidirectional, 3–state). These lines carry data and commands to and from the SCC.

D/$\overline{C}$. *Data/Control Select* (input). This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.

$\overline{RD}$. *Read* (input, active Low). This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.

$\overline{WR}$. *Write* (input, active Low). When the SCC is selected, this signal indicates a write operation. The coincidence of $\overline{RD}$ and $\overline{WR}$ is interpreted as a reset.

Z80C30

AD_0–AD_7. *Address/Data Bus* (bidirectional, active High, 3–state). These multiplexed lines carry register addresses to the SCC as well as data or control information.

$\overline{AS}$. *Address Strobe* (input, active Low). Addresses on AD_0–AD_7 are latched by the rising edge of this signal.

$\overline{CS}_0$. *Chip Select 0* (input, active Low). This signal is latched concurrently with the addresses on AD_0–AD_7 and must be active for the intended bus transaction to occur.

$\overline{CS}_1$. *Chip Select 1* (input, active High). This second select signal must also be active before the intended bus transaction can occur. CS_1 must remain active throughout the transaction.

$\overline{DS}$. *Data Strobe* (input, active Low). This signal provides timing for the transfer of data into and out of the SCC. If $\overline{AS}$ and $\overline{DS}$ coincide, this is interpreted as a reset.

R/$\overline{W}$. *Read/Write* (input). This signal specifies whether the operation to be performed is a read or a write.

Functional Description

The functional capabilities of the SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, the SCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities. The SCC provides two independent full–duplex channels programmable for use in any common Asynchronous or Synchronous data communication protocol. Figure 3 and the following description briefly detail these protocols.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one–and–one–half, or two stop bits per character and can provide a break output at any time. The receiver break–detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike–rejection mechanism that checks the signal one–half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing or error conditions using dedicated routines. Furthermore, a built–in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one–half a bit time to the point at which the search for the next start bit begins.

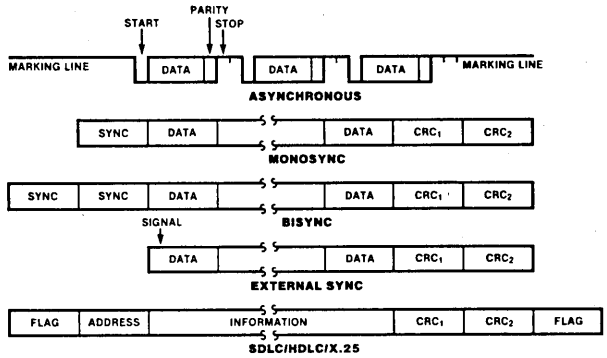


Figure 3. Some SCC Protocols

The SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In Asynchronous modes, the SYNC pin may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes. The SCC supports both byte–oriented and bit–oriented synchronous communication. Synchronous byte–oriented protocols can be handled in several modes, allowing character synchronization with a 6–bit or 8–bit synchronous character (Monosync), any 12–bit synchronization pattern (Bisync), or with an external synchronous signal. Leading sync characters can be removed without interrupting the CPU.

Five– or 7–bit synchronous characters are detected with 8– or 16–bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 4.

CRC checking for Synchronous byte–oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bysinc.

Both CRC–16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all Synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 6–, 8–, or 16–bit synchronous characters, regardless of the programmed character length.

The SCC supports Synchronous bit–oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, as external/status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user–selected address or to a global broadcast address. In this mode, frames not matching either the user–selected or broadcast address are ignored. The number of address

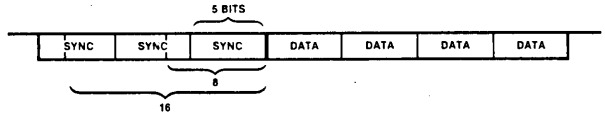


Figure 4. Detecting 5– or 7–Bit Synchronous Characters

bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end–of–frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s. The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The SCC can be conveniently used under DMA control to provide high speed reception or transmission. In reception, for example, the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end–of–frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end–of–frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode. The SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 5).

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by re–transmitting them with a one–bit–time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

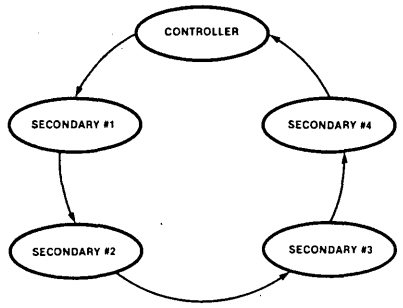


Figure 5. An SDLC Loop

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

Baud Rate Generator. Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8–bit time constant registers that form a 16–bit time constant, a 16–bit down counter, and a flip–flop on the output producing a square wave. On startup, the flip–flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase–Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the TRxC pin, the output of the baud rate generator may be echoed out via the TRxC pin.

The following formula relates the time constant to the baud rate where PCLK or RTxC is the baud rate generator input frequency in Hz. The clock mode is 1, 16, 32, or 64 as selected in Write Register 4, bits D6 and D7. Synchronous operation modes should select 1 and Asynchronous should select 16, 32, or 64.

$$\text{Time Constant} = \frac{\text{PCLK or RTxC Frequency}}{2\,(\text{Baud Rate})\,(\text{Clock Mode})} - 2$$

Digital Phase–Locked Loop. The SCC contains a Digital Phase–Locked Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding. The SCC may be programmed to encode and decode the serial data in four different ways (Figure 6). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi–phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi–phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi–phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

Auto Echo and Local Loopback. The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before re–transmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The SCC is also capable of local loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in Asynchronous, Synchronous and SDLC modes with NRZ, NRZI, or FM coding of the data stream.

I/O Interface Capabilities. The SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

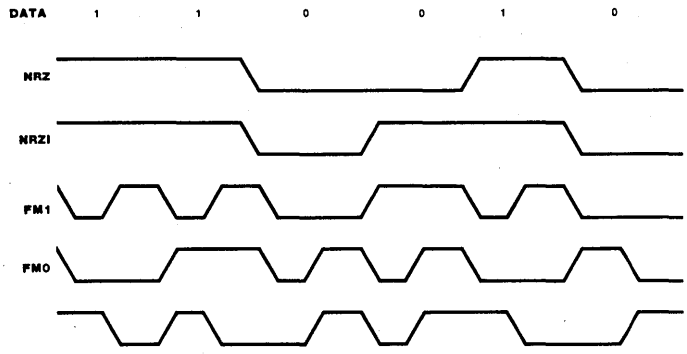


Figure 6. Data Encoding Methods

Polling. All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end–of–frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. When an SCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 10 and 11).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the interrupt priority chain (Figure 7). As a microprocessor peripheral, the SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the SCC and external to the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following: receiver overrun, framing error in Asynchronous mode, end–of–frame in SDLC mode and, optionally, a parity error. The Special

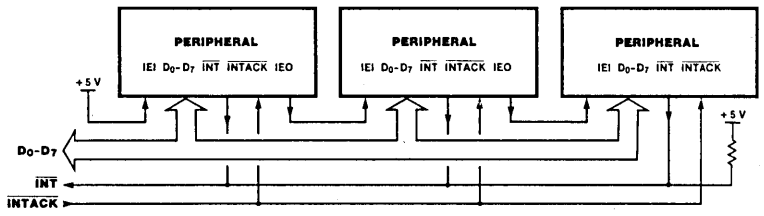


Figure 7. Interrupt Schedule

Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the CTS, DCD, and SYNC pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer. The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT/REQUEST}}$ output in conjuction with the Wait/Request bits in WR1. The $\overline{\text{WAIT/REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the SCC $\overline{\text{REQUEST}}$ output indicates that the SCC is ready to transfer data to or from memory. To the CPU, the WAIT line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR/REQUEST}}$ line allows full–duplex operation under DMA control.

Architecture

The SCC internal structure includes two full–duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 8).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general–purpose in nature and can optionally be used for functions other than modem control.

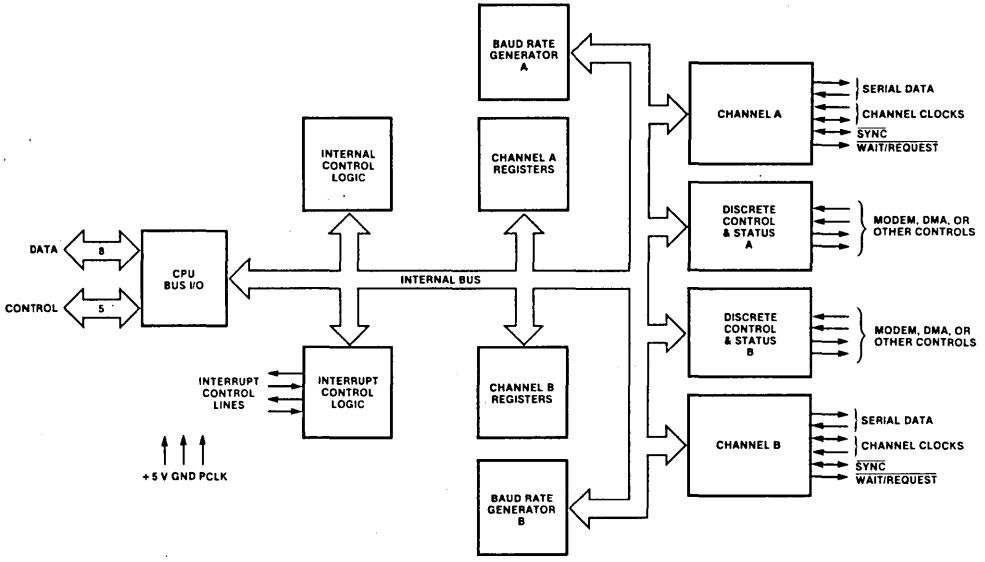


Figure 8. Block Diagram of SCC Architecture

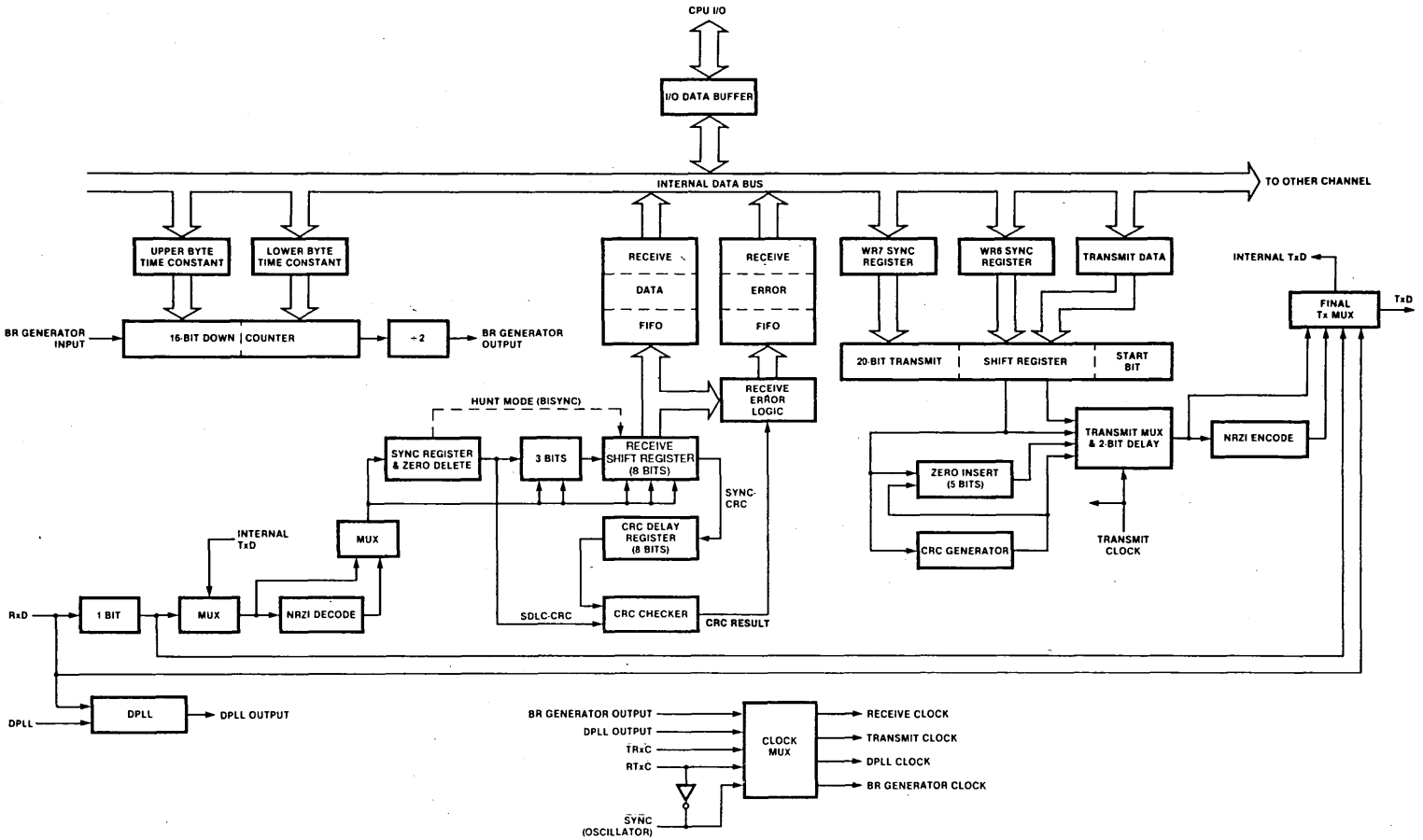


Figure 9. Data Path

The register set for each channel includes ten control (write) registers, two sync–character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write only Master Interrupt Control register and three read registers: one containing the vector with status information (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0–WR15 –– Write Registers 0 through 15.

RR0–RR3, RR10, RR12, RR13, RR15 –– Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 9 is identical for both channels. The receiver has three 8–bit buffer registers in a FIFO arrangement, in addition to the 8–bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8–bit Transmit Data buffer register loaded from the internal data bus and a 20–bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

	Read Register Functions
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information
	Write Register Functions
WR0	CRC initialize, initialization commands for the various modes, Register Pointers
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync character or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The SCC contains write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

Z85C30

In the SCC, register addressing is direct for the data registers only, which are selected by a High on the D/C pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed. All of the registers in the SCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

Z80C30

All SCC registers are directly addressable. How the SCC decodes the address placed on the address/data bus at the beginning of a Read or Write cycle is controlled by a command issued in WR0B. In the Shift Right mode the channel select A/B is taken from AD_0 and the state of AD_5 is ignored. In the Shift Left mode the channel select A/B is taken from AD_5 and the state of AD_0 is ignored. AD_7 and AD_6 are always ignored as address bits and the register address itself occupies AD_4–AD_1.

The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Read Registers. The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 10 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

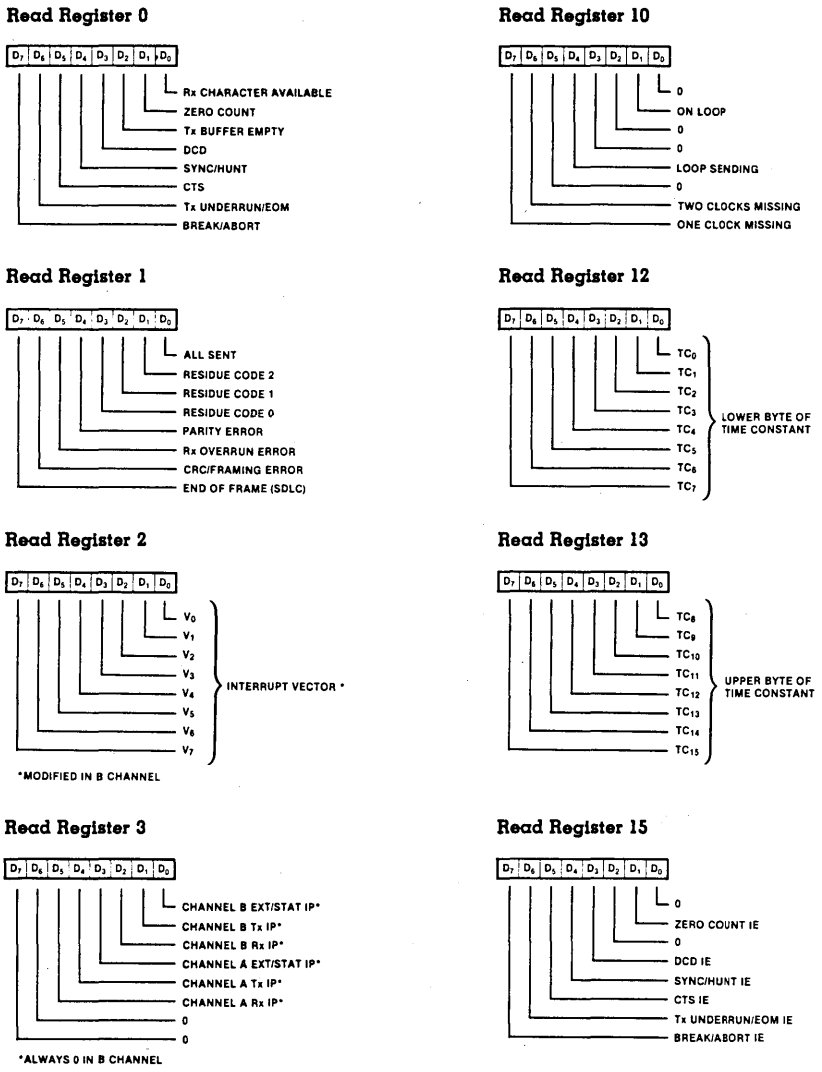


Figure 10. Read Register Bit Functions

Write Registers. The SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 11 shows the format of each write register.

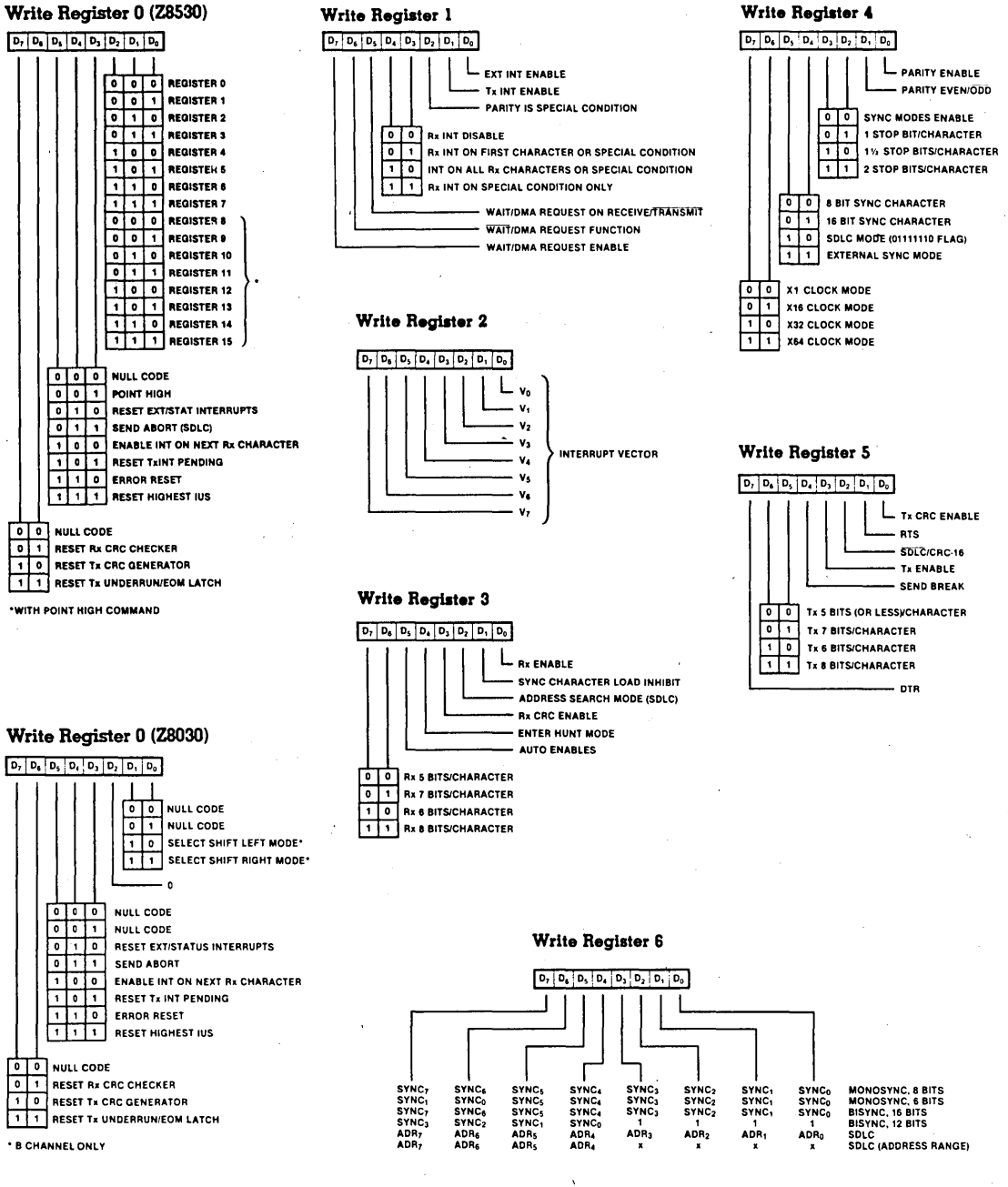
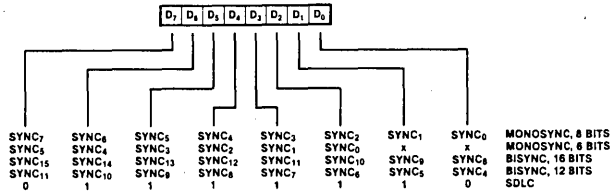
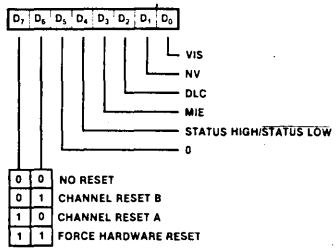


Figure 11. Write Register Bit Functions

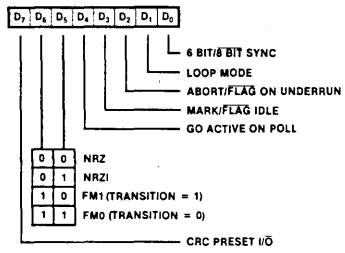
Write Register 7



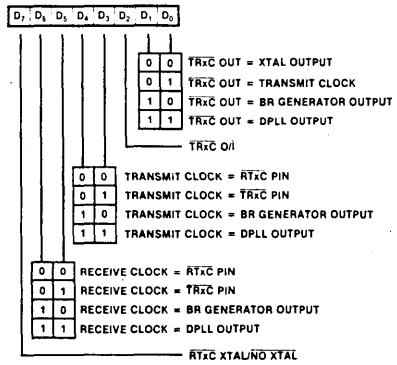
Write Register 9



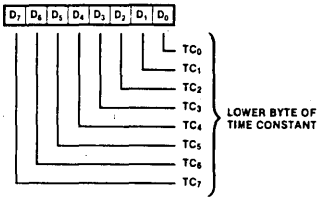
Write Register 10



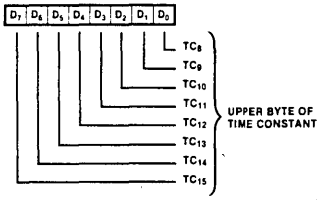
Write Register 11



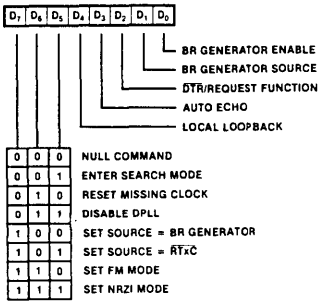
Write Register 12



Write Register 13



Write Register 14



Write Register 15

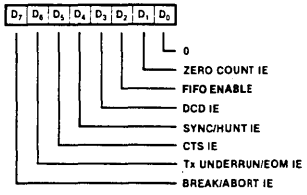


Figure 11. Write Register Bit Functions (Continued)

Z85C30 Timing

The SCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{WR}$ and $\overline{RD}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the falling edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the SCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the SCC. This time must be at least 4 PCLK regardless of which register or channel is being accessed.

Read Cycle Timing. Figure 12 illustrates Read cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{RD}$ falls or if it rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

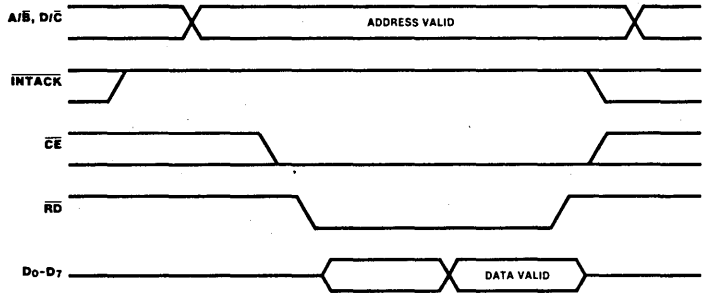


Figure 12. Read Cycle Timing

Write Cycle Timing. Figure 13 illustrates Write cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{WR}$ falls or if it rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened. Data must be valid before the falling edge of $\overline{WR}$.

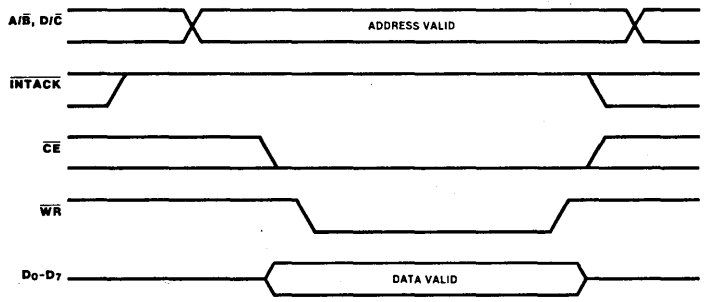


Figure 13. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 14 illustrates Interrupt Acknowledge cycle timing. Between the time $\overline{INTACK}$ goes Low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{RD}$ falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0–D_7 and it then sets the appropriate Interrupt–Under–Service latch internally.

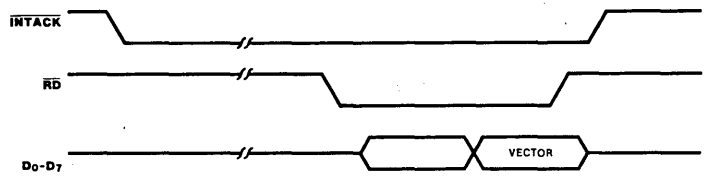


Figure 14. Interrupt Acknowledge Cycle Timing

Z80C30 Timing

The SCC generates internal control signals from $\overline{AS}$ and $\overline{DS}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{AS}$ and $\overline{DS}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the falling edge of $\overline{DS}$ in the first transaction involving the SCC to the falling edge of $\overline{DS}$ in the second transaction involving the SCC.

Read Cycle Timing. Figure 15 illustrates Read cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS}_0$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. $R/\overline{W}$ must be High to indicate a Read cycle. CS_1 must also be High for the Read cycle to occur. The data bus drivers in the SCC are then enabled while $\overline{DS}$ is Low.

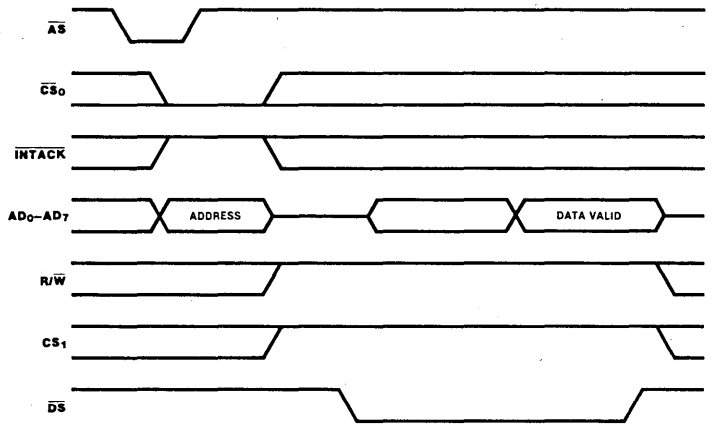


Figure 15. Read Cycle Timing

Write Cycle Timing. Figure 16 illustrates Write cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS}_0$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. $R/\overline{W}$ must be Low to indicate a Write cycle. CS_1 must be High for the Write cycle to occur. $\overline{DS}$ Low strobes the data into the SCC.

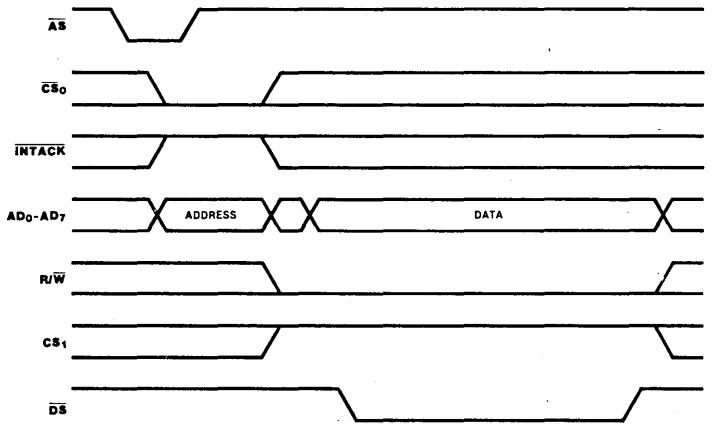


Figure 16. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 17 illustrates Interrupt Acknowledge cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. However, if $\overline{INTACK}$ is Low, the address and $\overline{CS_0}$ are ignored. The state of the R/$\overline{W}$ and CS_1 are also ignored for the duration of the Interrupt Acknowledge cycle. Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{DS}$ falls, the Acknowledge cycle was intended for the SCC. In this case, the SCC may be programmed to respond to RD Low by placing its interrupt vector on D_0–D_7 and it then internally sets the appropriate Interrupt–Under–Service latch.

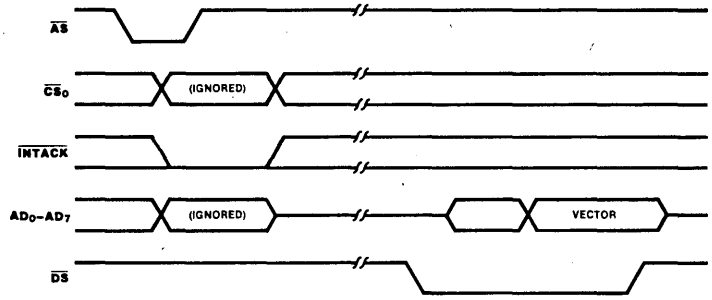


Figure 17. Interrupt Acknowledge Cycle Timing

FIFO

FIFO Enhancements. When used with a DMA controller, the Z85C30 FIFO enhancement maximizes the SCC's ability to receive high speed back–to–back SDLC messages while minimizing frame overruns due to CPU latencies in responding to interrupts. Additional logic was added to the industry standard NMOS SCC consisting of a 10 deep by 19 bit status FIFO, 14–bit receive byte counter, and control logic as shown in Figure 18. The 10 x 19 bit status FIFO is separate from the existing three byte receive data FIFO.

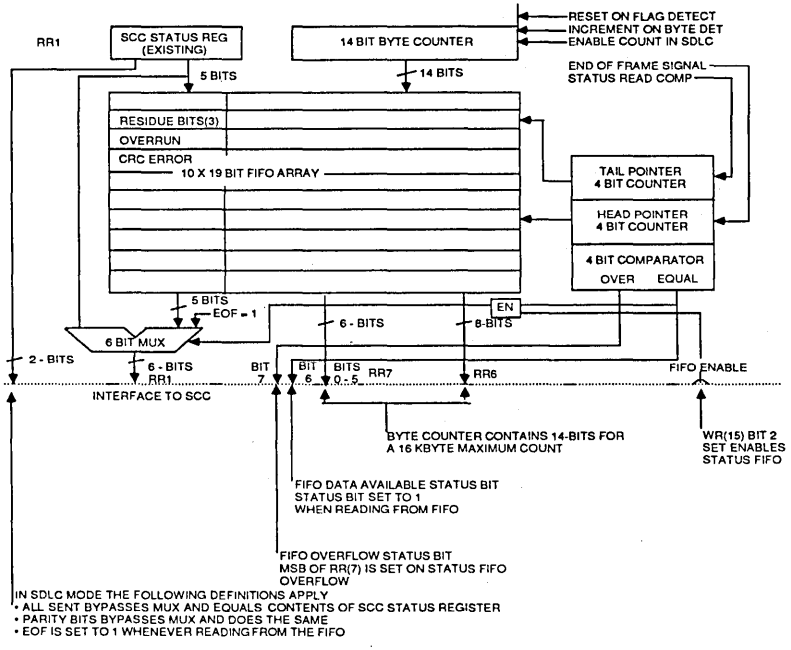


Figure 18. SCC Status Register Modifications.

When the enhancement is enabled, the status in read register 1 (RR1) and byte count for the SDLC frame will be stored in the 10 x 19 bit status FIFO. This allows the DMA controller to transfer the next frame into memory while the CPU verifies the message was properly received.

Summarizing the operation, data is received, assembled, loaded into the three byte receive FIFO before being transferred to memory by the DMA controller. When a flag is received at the end of an SDLC frame, the frame byte count from the 14–bit counter and five status bits are loaded into the status FIFO for verification by the CPU. The CRC checker is automatically reset in preparation for the next frame which can begin immediately. Since the byte count and status are saved for each frame, the message integrity can be verified at a later time. Status information for up to 10 frames can be stored before a status FIFO overrun could occur.

FIFO Detail. For a better understanding of details of the FIFO operation, refer to the block diagram contained in Figure 18.

Enable/Disable. This FIFO is implemented so that it is enabled when WR15 bit 2 is set and the SCC is in the SDLC/HDLC mode, otherwise the status register contents bypass the FIFO and go directly to the bus interface (the FIFO pointer logic is reset either when disabled or via a channel or power–on reset). When the FIFO mode is disabled, the SCC is completely downward–compatible with the NMOS Z8530. The FIFO mode is disabled on power–up (WR15 bit 2 is set to 0 on reset). The effects of backward compatibility on the register set are that RR4 is an image of RR0, RR5 is an image of RR1, RR6 is an image of RR2 and RR7 is an image of RR3. For the details of the added registers, refer to Figure 20. The status of the FIFO Enable signal can be obtained by reading RR15 bit 2. If the FIFO is enabled, the bit will be set to 1; otherwise, it will be reset.

Read Operation. When WR15 bit 2 is set and the FIFO is not empty, the next read to any of status register RR1 or the additional registers RR7 and RR6 will actually be from the FIFO. Reading status register RR1 causes one location of the FIFO to be emptied, so status should be read after reading the byte count, otherwise the count will be incorrect. Before the FIFO underflows, it is disabled. In this case, the multiplexer is switched to allow status to read directly from the status register, and reads from RR7 and RR6 will contain bits that are undefined. Bit 6 of RR7 (FIFO Data Available) can be used to determine if status data is coming from the FIFO or directly from the status register, since it is set to 1 whenever the FIFO is not empty.

Since not all status bits must be stored in the FIFO, the All Sent, Parity, and EOF bits will bypass the FIFO. The status bits sent through the FIFO will be Residue Bits (3), Overrun, and CRC Error.

The sequence for proper operation of the byte count and FIFO logic is to read the registers in the following order: RR7, RR6, and RR1 (reading RR6 is optional). Additional logic prevents the FIFO from being emptied by multiple reads from RR1. The read from RR7 latches the FIFO empty/full status bit (bit 6) and steers the status multiplexer to read from the SCC megacell instead of the status FIFO (since the status FIFO is empty). The read from RR1 allows an entry to be read from the FIFO (if the FIFO was empty, logic is added to prevent a FIFO underflow condition).

Write Operation. When the end of an SDLC frame (EOF) has been received and the FIFO is enabled, the contents of the status and byte–count registers are loaded into the FIFO. The EOF signal is used to increment the FIFO. If the FIFO overflows, the MSB of RR7 (FIFO Overflow) is set to indicate the overflow. This bit and the FIFO control logic is reset by disabling and re–enabling the FIFO control bit (WR15 bit 2). For details of FIFO control timing during an SDLC frame, refer to Figure 19.

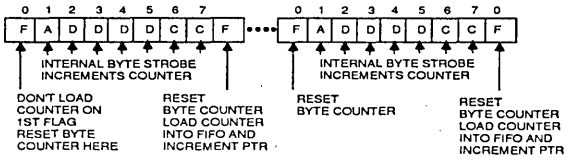


Figure 19. SDLC Byte Counting Detail.

Byte Counter Detail. The 14–bit byte counter allows for packets up to 16K bytes to be received. For a better understanding of its operation refer to Figures 18 and 19.

Enable. The byte counter is enabled in the SDLC/HDLC mode.

Reset. The byte counter is reset whenever an SDLC flag character is received. The reset is timed so that the contents of the byte counter are successfully written into the FIFO.

Increment. The byte counter is incremented by writes to the data FIFO. The counter represents the number of bytes received by the SCC, rather than the number of bytes transferred from the SCC. (These counts may differ by up to the number of bytes in the receive data FIFO contained in the SCC).

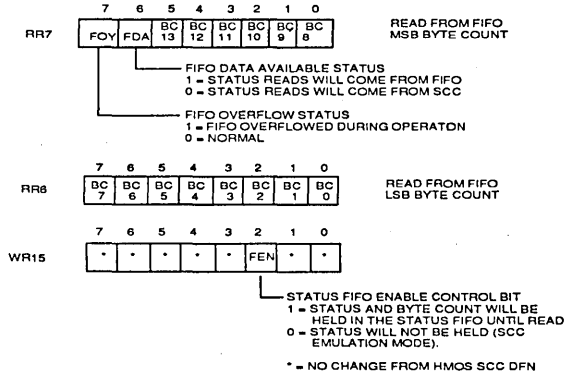


Figure 20. SCC Additional Registers.

Absolute Maximum Ratings

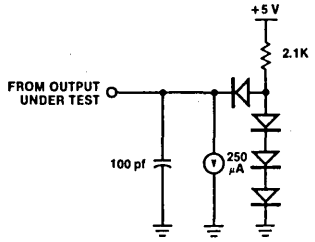
Voltages on all pins with respect to GND.....–0.3 V to +7.0 V
Operating Ambient Temperature..... See Ordering Information
Storage Temperature.....–65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

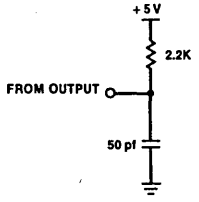
Standard Test Conditions

The DC characteristics and capacitance section below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.
Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information



Standard Test Load



Open–Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Typ	Max	Unit	Condition
V_{IH}	Input High Voltage	2.2		$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{OH1}	Output High Voltage	2.4			V	$I_{OH} = -1.6$ mA
V_{OH2}	Output High Voltage	$V_{CC}-0.8$			V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage			0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage			± 10.0	μA	$0.4 \le V_{IN} \le +2.4V$
I_{OL}	Output Leakage			± 10.0	μA	$0.4 \le V_{OUT} \le +2.4V$
I_{CC1}	V_{CC} Supply Current		7	30	mA	$V_{CC} = 5V$ $V_{IH} = 4.8V$ $V_{IL} = 0.2V$

$V_{CC} = 5\ V \pm 5\%$ unless otherwise specified, over specified temperature range. * Typical I_{CC} was measured with oscillator off.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.
Unmeasured pins returned to ground.

Miscellaneous

Gate Count	6000

Z85C30 AC CHARACTERISTICS

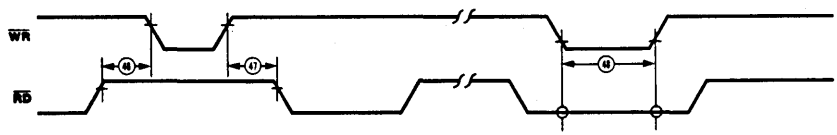
Number	Symbol	Parameter	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Notes †
1	TwPCl	PCLK Low Width	50	1000	40	1000	
2	TwPCh	PCLK High Width	50	1000	40	1000	
3	TfPC	PCLK Fall Time		10		10	
4	TrPC	PCLK Rise Time		10		10	
5	TcPC	PCLK Cycle Time	125	2000	100	2000	
6	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	70		50		
7	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		
8	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	70		50		
9	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		
10	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	20		20		
11	TsIAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	145		130		1
12	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		
13	TsIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	145		130		1
14	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		
15	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	40		30		
16	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		
17	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		
18	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	60		50		
19	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		1
20	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		1
21	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	60		50		1
22	TwRDl	$\overline{RD}$ Low Width	150		125		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
24	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
25	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		140		120	
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		40		35	

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.

†Units in nanoseconds (ns).

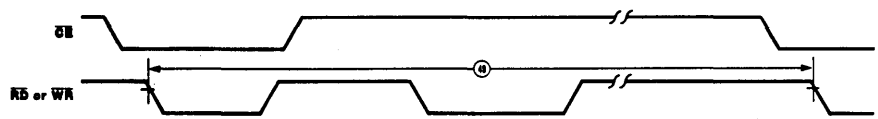
Reset Timing

Z85C30



Cycle Timing

Z85C30



Z85C30 AC CHARACTERISTICS (Continued)

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Notes †
27	TdA(DR)	Address Required Valid to Read Data Valid Delay		220		180	
28	TwWRI	$\overline{WR}$ Low Width	150		125		
29	TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time	10		10		
30	ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time	0		0		
31	TdWR(W)	$\overline{WR}$ ↓ to Wait Valid Delay		170		160	4
32	TdRD(W)	$\overline{RD}$ ↓ Wait Valid Delay		170		160	4
33	TdWRf(REQ)	$\overline{WR}$ ↓ to W/$\overline{REQ}$ Not Valid Delay		170		160	
34	TdRDf(REQ)	$\overline{RD}$ ↓ to W/$\overline{REQ}$ Not Valid Delay		170		160	
35	TdWRr(REQ)	$\overline{WR}$ ↓ $\overline{DTR}$/$\overline{REQ}$ Not Valid Delay		4TcPC		4TcPC	
36	TdRDr(REQ)	$\overline{RD}$ ↑ to $\overline{DTR}$/$\overline{REQ}$ Not Valid Delay		4TcPC		4TcPC	
37	TdPC(INT)	PCLK ↓ to $\overline{INT}$ Valid Delay		500		500	4
38	TdIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Delay	150			125	5
39	TwRDA	$\overline{RD}$ (Acknowledge) Width	150			125	
40	TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay		140		120	
41	TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time	95		95		
42	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time	0		0		
43	TdIEI(IEO)	IEI to IEO Delay Time		95		90	
44	TdPC(IEO)	PCLK ↑ to IEO Delay		200		175	
45	TdRDA(INT)	$\overline{RD}$ ↓ to $\overline{INT}$ Inactive Delay		500		500	4
46	TdRD(WRQ)	$\overline{RD}$ ↑ to $\overline{WR}$ ↓ Delay for No Reset	15		15		
47	TdWRQ(RD)	$\overline{WR}$ ↑ to $\overline{RD}$ ↓ Delay for No Reset	15		15		
48	TwRES	$\overline{WR}$ and $\overline{RD}$ Coincident Low for Reset	150		100		
49	Trc	Valid Access Recovery Time	4TcPC		4TcPC		3

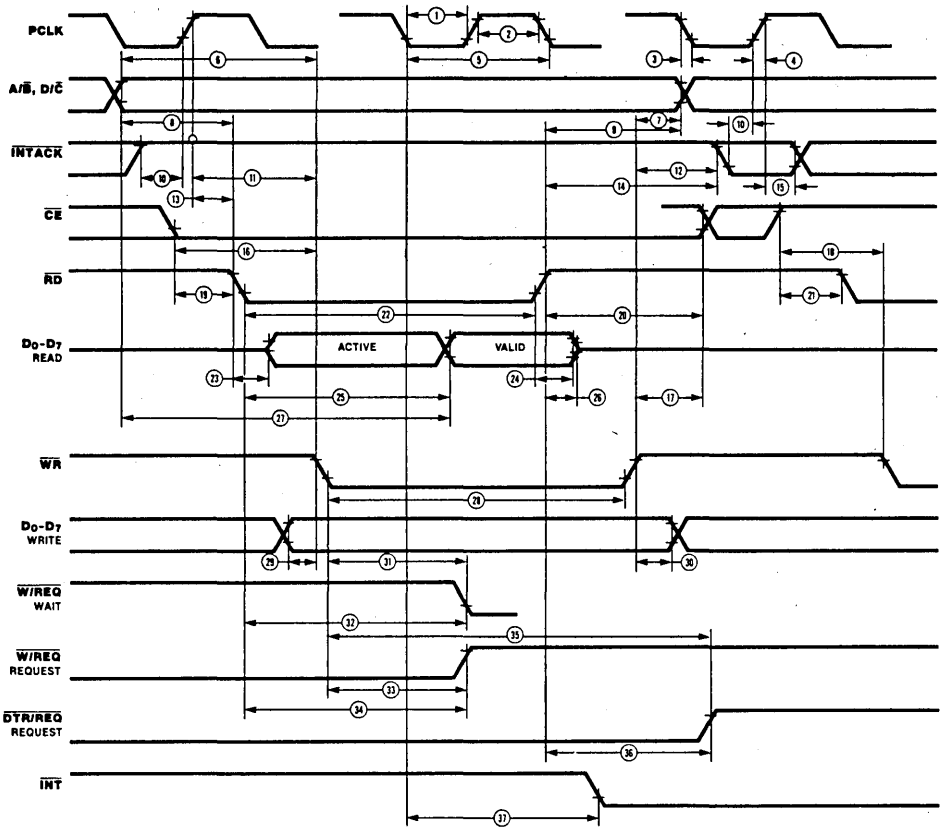
NOTES:

3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.

†Units in nanoseconds (ns).

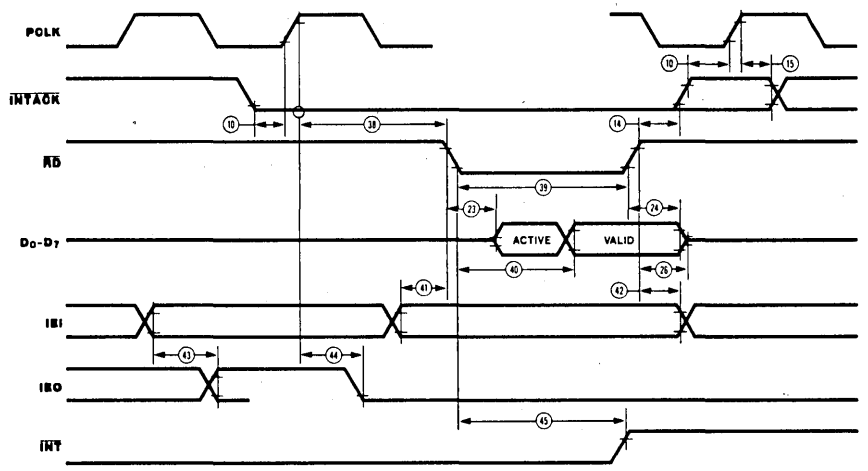
Read and Write Timing

Z85C30



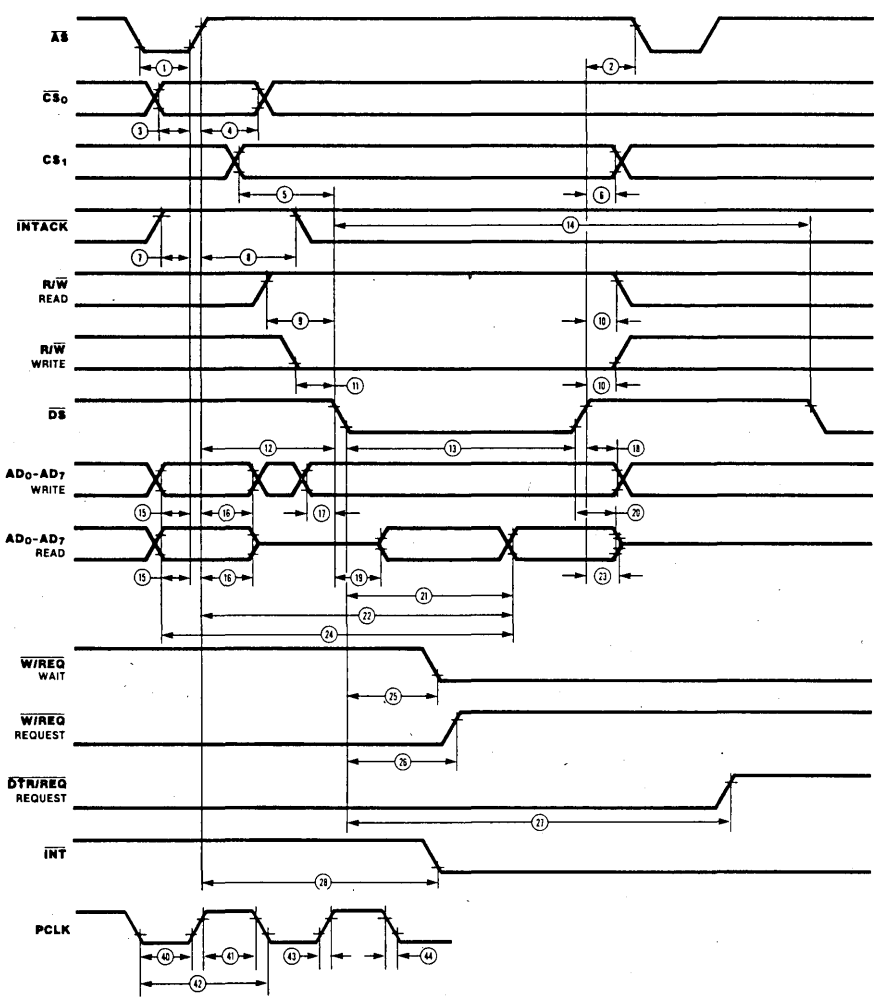
Interrupt Acknowledge Timing

Z85C30



Read and Write Timing

Z80C30



Z80C30 AC CHARACTERISTICS

Preliminary

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Notes †
1	TwAS	$\overline{AS}$ Low Width	35		30		
2	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	15		10		
3	TsCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
4	ThCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	30		20		1
5	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	65		50		1
6	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	30		20		1
7	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	10		10		
8	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	150		125		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	65		50		
10	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	35		25		
11	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
12	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	30		20		
13	TwDSl	$\overline{DS}$ Low Width	150		125		
14	TrC	Valid Access Recovery Time	4TcPC		4TcPC		2
15	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	10		10		1
16	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	25		20		1
17	TsDW(DS)	Write Data to $\overline{DS}$ ↓ Setup Time	15		10		
18	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	20		15		
19	TdDS(DA)	$\overline{DS}$ ↓ to Data Active Delay	0		0		
20	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
21	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		140	120		
22	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Valid Delay		250	190		

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Parameter applies only between transactions involving the SCC.

†Units in nanoseconds (ns).

Z80C30 AC CHARACTERISTICS (Continued)

			Preliminary				
			8 MHz		10 MHz		
Number	Symbol	Parameter	Min	Max	Min	Max	Notes †
23	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		40		35	3
24	TdA(DR)	Address Required Valid to Read Data Valid Delay		260		210	
25	TdDS(W)	$\overline{DS}$ ↓ to-Wait Valid Delay		170		160	4
26	TdDSf(REQ)	$\overline{DS}$ ↓ to $\overline{W/REQ}$ Not Valid Delay		170		160	
27	TdDSr(REQ)	$\overline{DS}$ ↓ to $\overline{DTR/REQ}$ Not Valid Delay		4TcPC		4TcPC	
28	TdAS(INT)	$\overline{AS}$ ↑ to $\overline{INT}$ Valid Delay		500		500	4
29	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay	250		225		5
30	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	150		125		
31	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		140		120	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	80		80		
33	ThIEI(DSA)	IEI to $\overline{DS}$ ↑ (Acknowledge) Hold Time	0		0		
34	TdIEI(IEO)	IEI to IEO Delay		90		90	
35	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		200		175	6
36	TdDSA(INT)	$\overline{DS}$ ↓ (Acknowledge) to $\overline{INT}$ Inactive Delay		450		450	4
37	TdDS(ASQ)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay for No Reset	15		15		
38	TdASQ(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay for No Reset	20		15		
39	TwRES	$\overline{AS}$ and $\overline{DS}$ Coincident Low for Reset	150		100		7
40	TwPCl	PCLK Low Width	50	1000	40	1000	
41	TwPCh	PCLK High Width	50	1000	40	1000	
42	TcPC	PCLK Cycle Time	125	2000	100	2000	
43	TrPC	PCLK Rise Time		10		10	
44	TfPC	PCLK Fall Time		10		10	

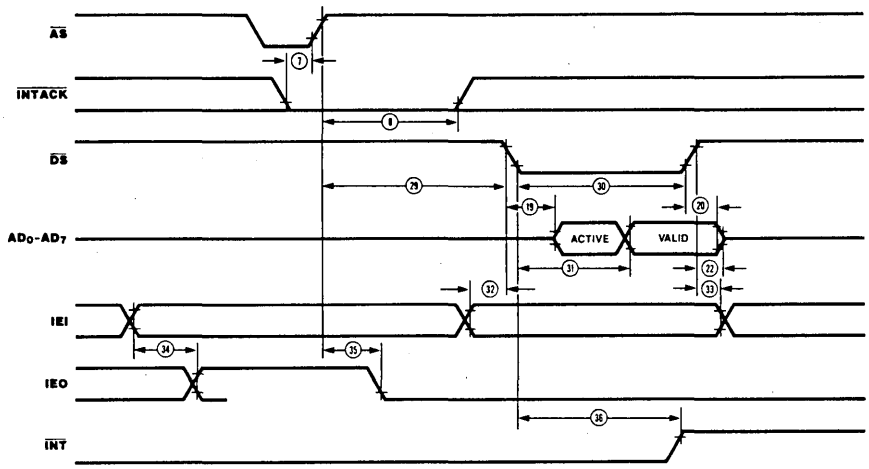
NOTES:

3. Float delay is defined as the time required for a ±0.5V change in the output with a maximum dc load and a minimum ac load.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any Z-SCC in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the highest priority device in the daisy chain, TsIEI(DSA) for the Z-SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.
6. Parameter applies only to a Z-SCC pulling INT Low at the beginning of the Interrupt Acknowledge transaction.
7. Internal circuitry allows for the reset provided by the Z8 to be recognized as a reset by the Z-SCC. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".

†Units in nanoseconds (ns).

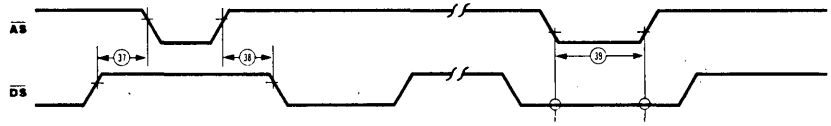
Interrupt Acknowledge Timing

Z80C30



Reset Timing

Z80C30



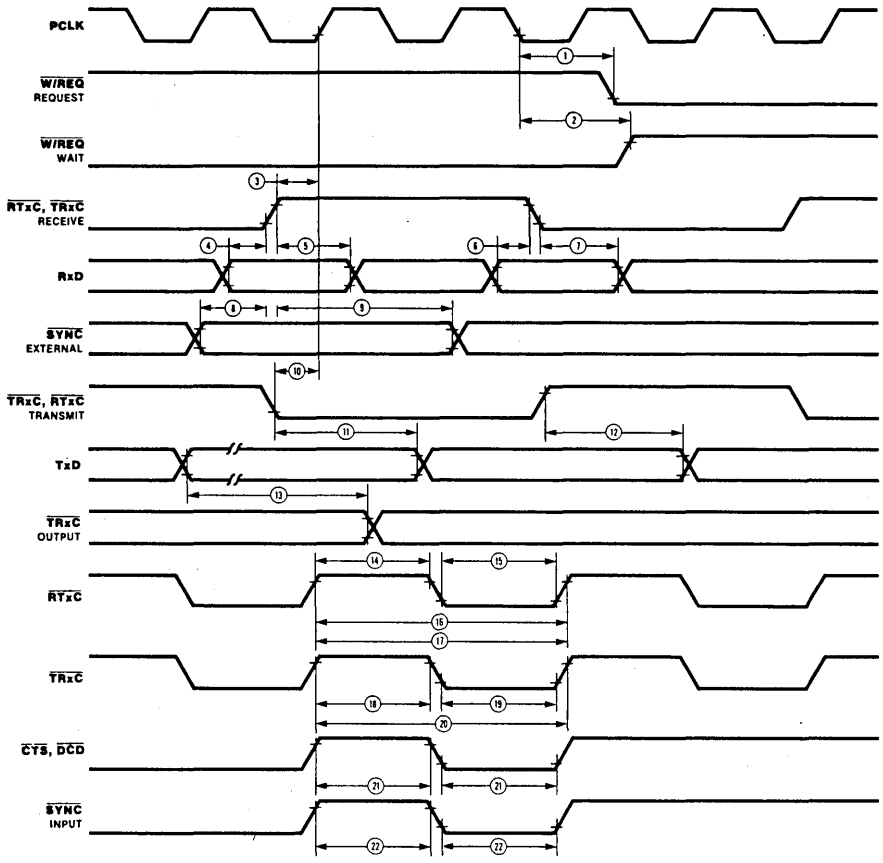
Z80C30/Z85C30 GENERAL TIMING AC CHARACTERISTICS

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Notes †
1	TdPC(REQ)	PCLK ↓ to W/$\overline{REQ}$ Valid Delay		250		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350		350	
3	TsRXC(PC)	$\overline{RxC}$ ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	60	TwPCL	40	TwPCL	1,4
4	TsRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Setup Time (X1 Mode)	0		0		1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode)	150		150		1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Setup Time (X1 Mode)	0		0		1,5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode)	150		150		1,5
8	TsSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Setup Time	−200		−200		1
9	ThSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Hold Time	5TcPC		5TcPC		1
10	TsTXC(PC)	$\overline{TxC}$ ↓ to PCLK ↑ Setup Time	0		0		2,4
11	TdTXCf(TXD)	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode)		200		150	2
12	TdTxCr(TXD)	$\overline{TxC}$ ↑ to TxD Delay (X1 Mode)		200		150	2,5
13	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)		200		200	
14	TwRTXh	$\overline{RTxC}$ High Width	150		150		6
15	TwRTXl	$\overline{RTxC}$ Low Width	150		150		6
16	TcRTX	$\overline{RTxC}$ Cycle Time (RxD, TxD)	500		400		6,7
17	TcRTXX	Crystal Oscillator Period	125	1000	100	1000	3
18	TwTRXh	$\overline{TRxC}$ High Width	150		150		6
19	TwTRXl	$\overline{TRxC}$ Low Width	150		150		6
20	TcTRX	$\overline{TRxC}$ Cycle Time	500		400		6,7
21	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ Pulse Width	200		200		
22	TwSY	$\overline{SYNC}$ Pulse Width	200		200		

NOTES:
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both $\overline{RTxC}$ and $\overline{SYNC}$ have 30 pf capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to case PCLK requirements.
7. The maximum receive or transmit data is ¼ PCLK.

†Units in nanoseconds (ns).

General Timing



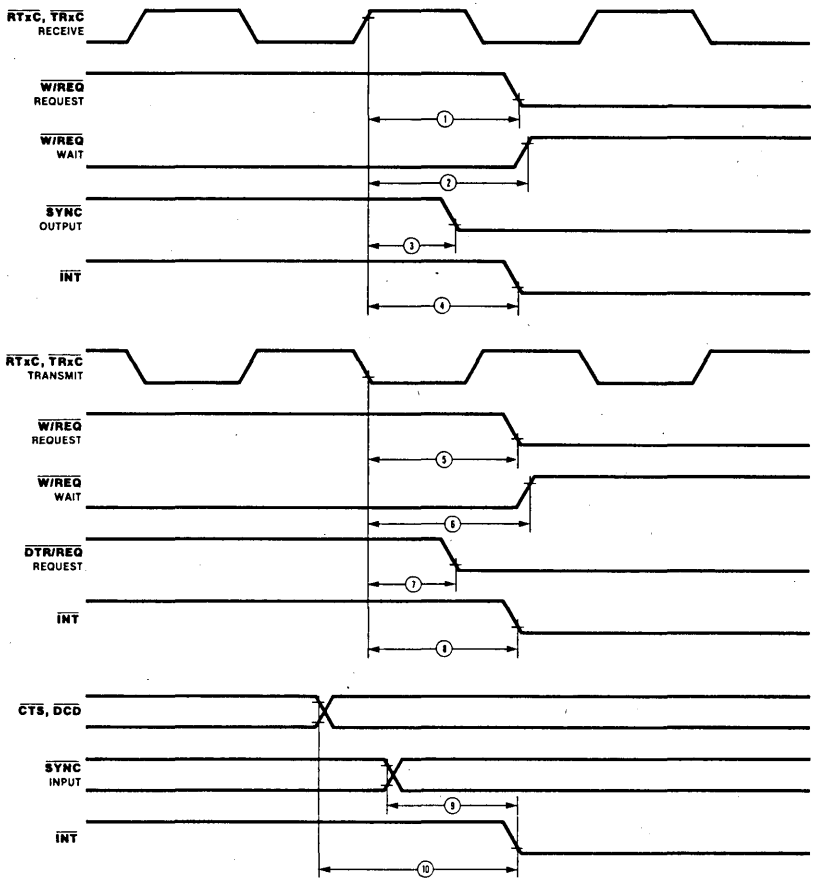
Z80C30/Z85C30 SYSTEM TIMING AC CHARACTERISTICS

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Notes †
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	8	12	8	12	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	14	8	14	1,2
3	TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	4	7	4	7	2
4a.	TdRXC(INT), Z8530	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	10	16	10	16	1,2
4b.	TdRXC(INT), Z8030		8	12	8	12	1,2
			+2	+3	+2	+3	4
5	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	5	8	5	8	3
6	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	11	5	11	1,3
7	TdTXC(DRQ)	$\overline{TxC}$ ↓ $\overline{DTR/REQ}$ Valid Delay	4	7	4	7	3
8a.	TdTXC(INT), Z8530	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	6	10	6	10	1,3
8b.	TdTXC(INT), Z8030		4	6	4	6	1,3
			+2	+3	+2	+3	4
9a.	TdSY(INT), Z8530	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	1
9b.	TdSY(INT), Z8030		2	3	2	3	1,4
10a.	TdEXT(INT), Z8530	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	1
10b.	TdEXT(INT), Z8030		2	3	2	3	1,4

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
4. Units equal to $\overline{AS}$.

†Units equal to TcPC.

System Timing



Z8030 Z-BUS SCC/ Z8530 SCC Serial Communications Controller

October 1987

Features

- Two independent, 0 to 2M bit/second, full-duplex channels, each with a separate crystal oscillator, baud rate generator, and Digital Phase-Locked Loop for clock recovery.
- Multi-protocol operation under program control; programmable for NRZ, NRZI, or FM data encoding.
- Asynchronous mode with five to eight bits and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Synchronous mode with internal or external character synchronization on one or two synchronous characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and SDLC Loop mode operation.
- Local Loopback and Auto Echo modes.
- Supports T1 digital trunk.

General Description

The SCC Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with conventional non-multiplexed buses and the Zilog Z-BUS.® The SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, Digital PhaseLocked Loops, and crystal oscillators that dramatically reduce the need for external logic.

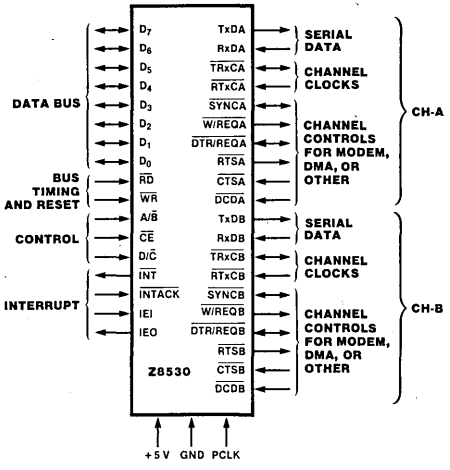


Figure 1a. Pin Functions, Z8530

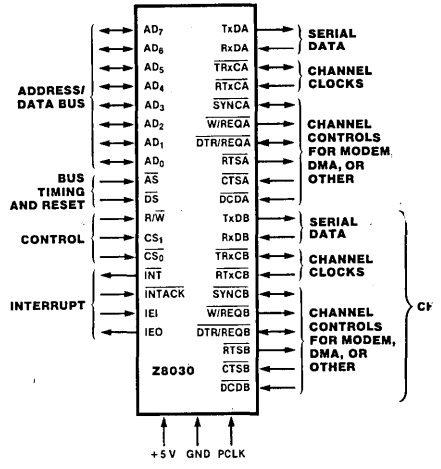


Figure 1b. Pin Functions, Z8030

General Description (Continued)

The SCC handles asynchronous formats, Synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives, etc.).

The device can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

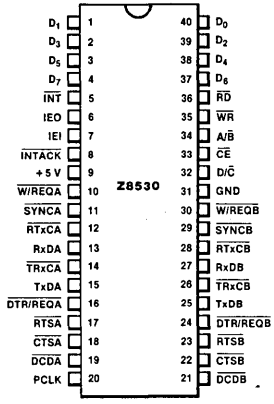


Figure 2a. DIP Pin Assignments, Z8530

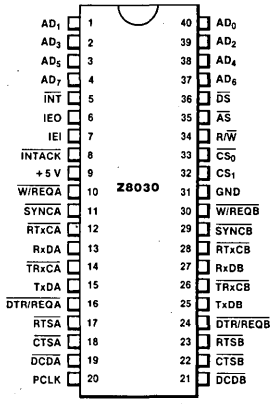


Figure 2b. DIP Pin Assignments, Z8030

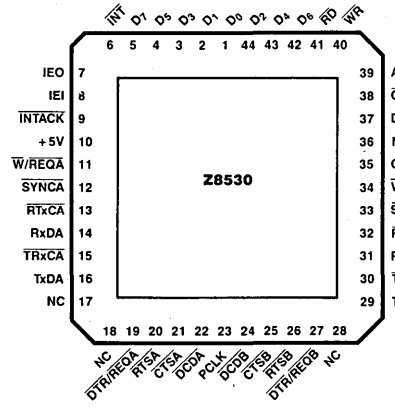


Figure 2c. Chip Carrier Pin Assignments, Z8530

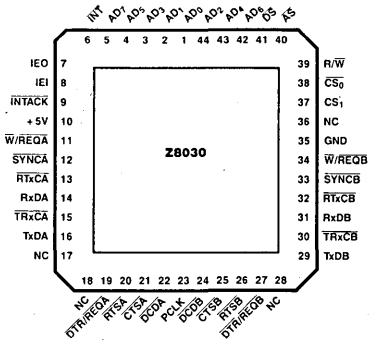


Figure 2d. Chip Carrier Pin Assignments, Z8030

Pin Description

The following section describes the pin functions common to the Z8530 and the Z8030. Figures 1 and 2 detail the respective pin functions and pin assignments.

$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$. *Clear To Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$. *Data Carrier Detect* (inputs/outputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accomodate slow rise-time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

$\overline{\text{DTR/REQA}}$, $\overline{\text{DTR/REQB}}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interruptdriven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{\text{INT}}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the SCC requests an interrupt.

$\overline{\text{INTACK}}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When $\overline{\text{RD}}$ or $\overline{\text{DS}}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). $\overline{\text{INTACK}}$ is latched by the rising edge of PCLK.

PCLK. *Clock* (input). This is the master SCC clock used to synchronize internal signals. PCLK is a TTL level signal. PCLK is not required to have any phase relationship with the master system clock.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{\text{RTxCA}}$ $\overline{\text{RTxCB}}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{\text{RTxC}}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{\text{SYNC}}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 11) is set, the $\overline{\text{RTS}}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$. *Synchronization* (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 10) but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{\text{SYNC}}$ must be driven Low two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{\text{SYNC}}$.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{TRxCA}$, $\overline{TRxCB}$. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. TRxC may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{W/REQA}$, $\overline{W/REQB}$. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait.

Z8530

A/$\overline{B}$. *Channel A/Channel B Select* (input). This signal selects the channel in which the read or write operation occurs.

$\overline{CE}$. *Chip Enable* (input, active Low). This signal selects the SCC for a read or write operation.

D_0-D_7 *Data Bus* (bidirectional, 3-state). These lines carry data and commands to and from the SCC.

D/$\overline{C}$. *Data/Control Select* (input). This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.

$\overline{RD}$. *Read* (input, active Low). This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.

$\overline{WR}$. *Write* (input, active Low). When the SCC is selected, this signal indicates a write operation. The coincidence of $\overline{RD}$ and $\overline{WR}$ is interpreted as a reset.

Z8030

AD_0-AD_7. *Address/Data Bus* (bidirectional, active High, 3-state). These multiplexed lines carry register addresses to the SCC as well as data or control information.

$\overline{AS}$. *Address Strobe* (input, active Low). Addresses on AD_0-AD_7 are latched by the rising edge of this signal.

$\overline{CS_0}$. *Chip Select 0* (input, active Low). This signal is latched concurrently with the addresses on AD_0-AD_7 and must be active for the intended bus transaction to occur.

CS_1. *Chip Select 1* (input, active High). This second select signal must also be active before the intended bus transaction can occur. CS_1 must remain active throughout the transaction.

$\overline{DS}$. *Data Strobe* (input, active Low). This signal provides timing for the transfer of data into and out of the SCC. If $\overline{AS}$ and $\overline{DS}$ coincide, this is interpreted as a reset.

R/$\overline{W}$. *Read/Write* (input). This signal specifies whether the operation to be performed is a read or a write.

Functional Description

The functional capabilities of the SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, the SCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities. The SCC provides two independent full-duplex channels programmable for use in any common Asynchronous or Synchronous data-communication protocol. Figure 3 and the following description briefly detail these protocols.

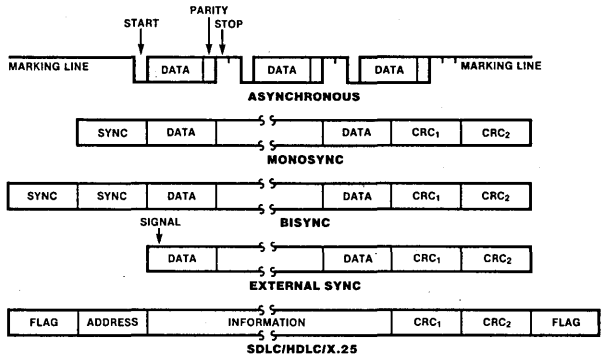


Figure 3. Some SCC Protocols

Functional Description (Continued)

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes. The SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes, allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit synchronization pattern (Bisync), or with an external synchronous signal. Leading sync characters can be removed without interrupting the CPU.

Five- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 4.

CRC checking for Synchronous byte-oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all Synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The SCC supports Synchronous bit-oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of

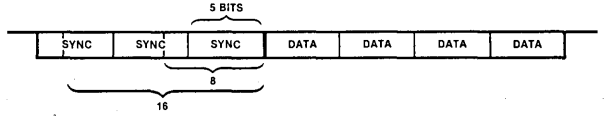


Figure 4. Detecting 5- or 7-Bit Synchronous Characters

Functional Description (Continued)

transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s. The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The SCC can be conveniently used under DMA control to provide high speed reception or transmission. In reception, for example, the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode. The SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 5).

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

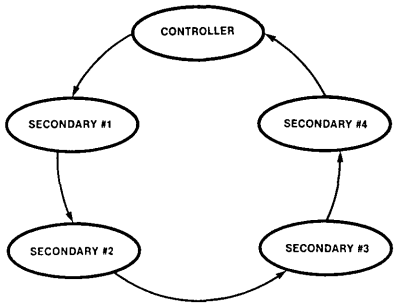


Figure 5. An SDLC Loop

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

Baud Rate Generator. Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the TRxC pin, the output of the baud rate generator may be echoed out via the TRxC pin.

The following formula relates the time constant to the baud rate where PCLK or RTxC is the baud rate generator input frequency in Hz. The clock mode is 1, 16, 32, or 64 as selected in Write Register 4, bits D6 and D7. Synchronous operation modes should select 1 and Asynchronous should select 16, 32, or 64.

$$\text{Time Constant} = \frac{\text{PCLK or RTxC Frequency}}{2\ (\text{Baud Rate})\ (\text{Clock Mode})} - 2$$

Digital Phase-Locked Loop. The SCC contains a Digital Phase-Locked-Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the

Functional Description (Continued)

incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding. The SCC may be programmed to encode and decode the serial data in four different ways (Figure 6). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

Auto Echo and Local Loopback. The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The SCC is also capable of local loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in Asynchronous, Synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be

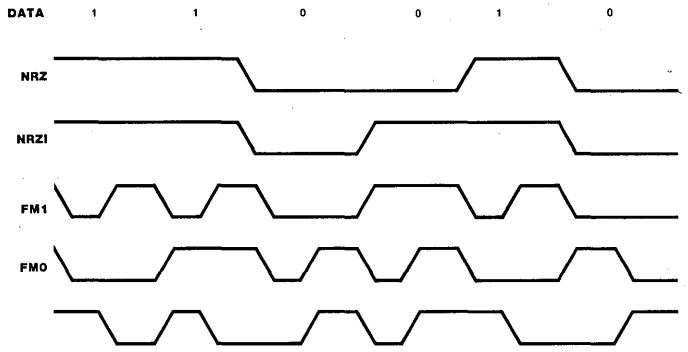


Figure 6. Data Encoding Methods

Functional Description (Continued)

read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. When an SCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 10 and 11).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the interrupt priority chain (Figure 7). As a microprocessor peripheral, the SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the SCC and external to the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following: receiver overrun, framing error in Asynchronous mode, end-of-frame in SDLC mode and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$, and $\overline{SYNC}$ pins; however, an

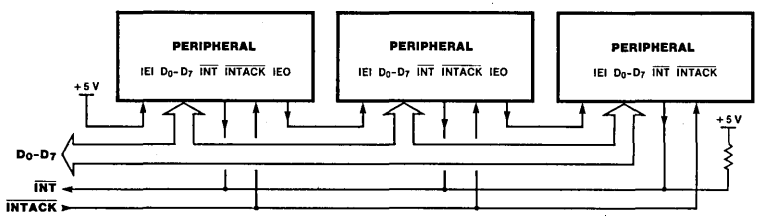


Figure 7. Interrupt Schedule

Functional Description (Continued)

External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer. The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the SCC $\overline{\text{REQUEST}}$ output indicates that the SCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}/\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The SCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 8).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two sync-character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a

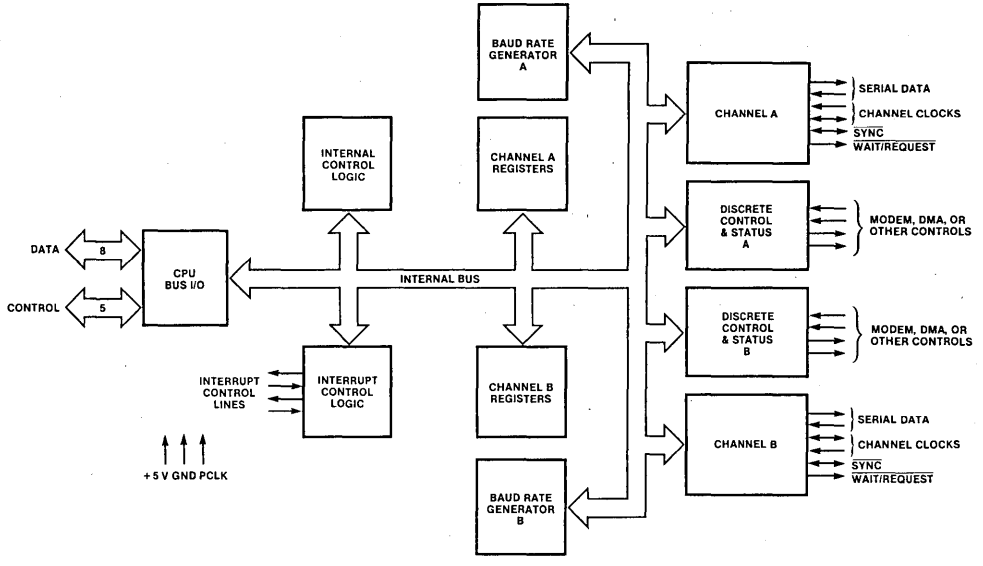


Figure 8. Block Diagram of SCC Architecture

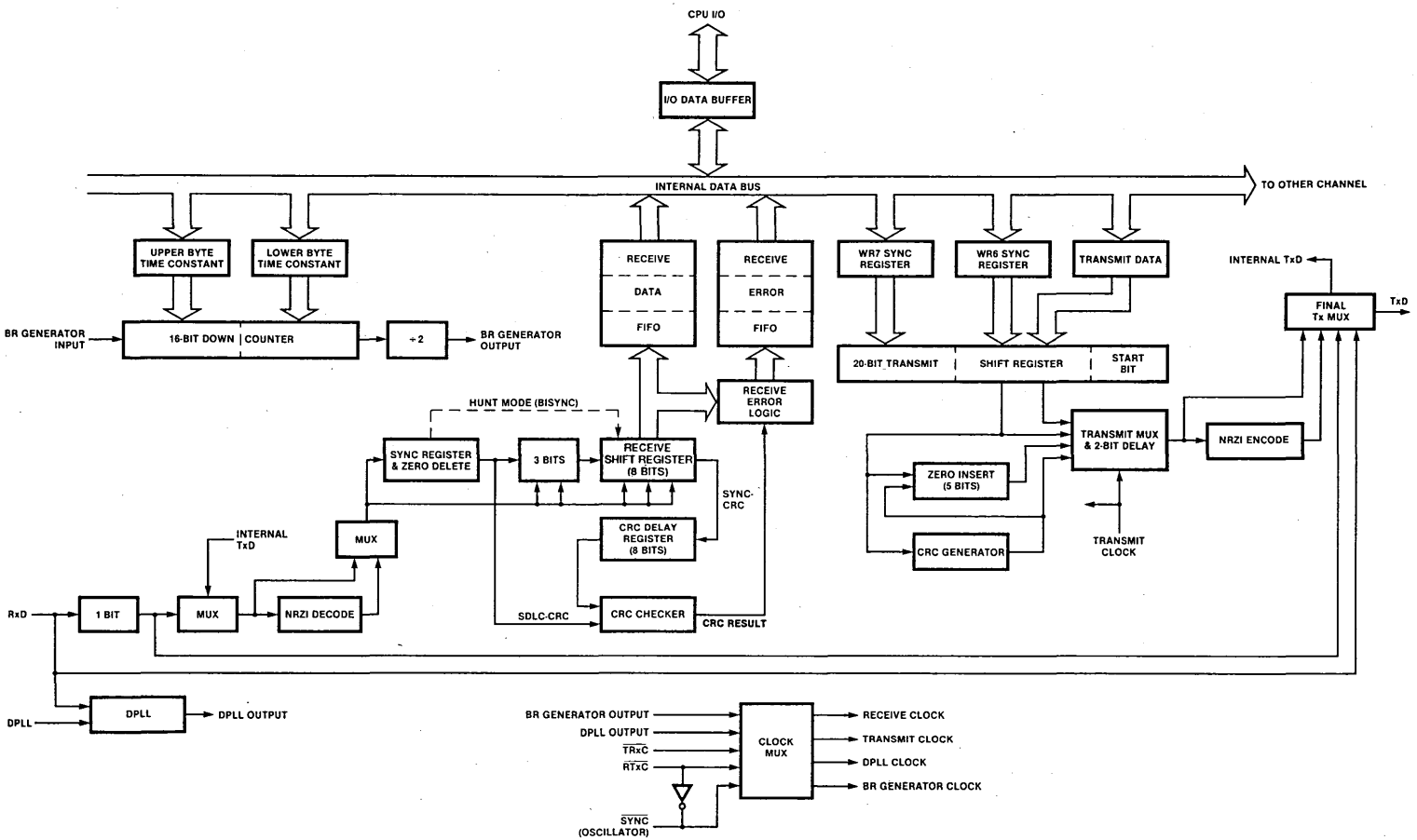


Figure 9. Data Path

Architecture (Continued)

write only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0-WR15 — Write Registers 0 through 15.

RR0-RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 9 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and a 20-bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD)

Read Register Functions	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information
Write Register Functions	
WR0	CRC initialize, initialization commands for the various modes, Register Pointers
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync character or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The SCC contains write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

Z8530

In the SCC, register addressing is direct for the data registers only, which are selected by a High on the D/C pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed. All of the registers in the SCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

Z8030

All SCC registers are directly addressable. How the SCC decodes the address placed on the address/data bus at the beginning of a Read or Write cycle is controlled by a command issued in WROB. In the Shift Right mode the channel select A/B is taken from AD_0 and the state of AD_5 is ignored. In the Shift Left mode the channel select A/B is taken from AD_5 and the state of AD_0 is ignored. AD_7 and AD_6 are always ignored as address bits and the register address itself occupies AD_4-AD_1.

Z8530/Z8030

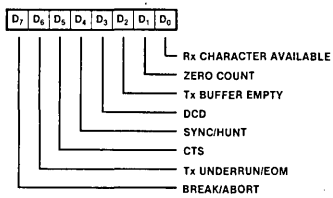
The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

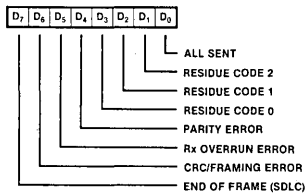
Read Registers. The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 10 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

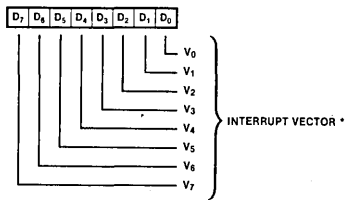
Read Register 0



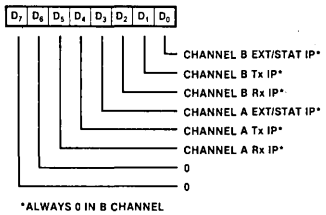
Read Register 1



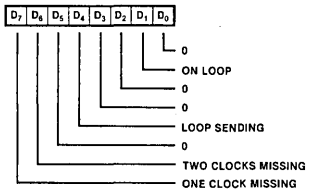
Read Register 2



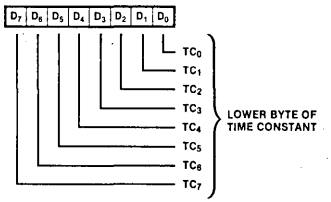
Read Register 3



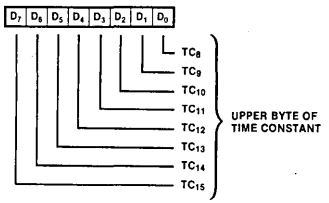
Read Register 10



Read Register 12



Read Register 13



Read Register 15

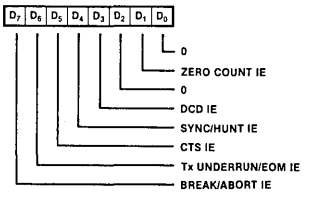
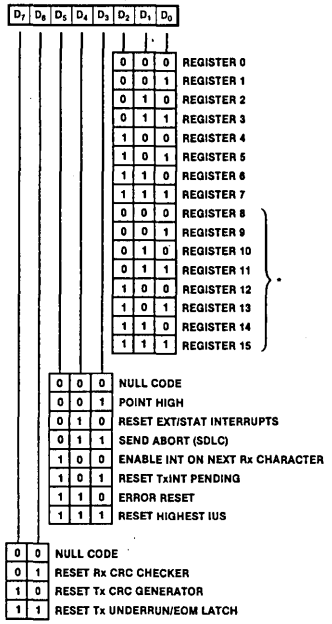


Figure 10. Read Register Bit Functions

Write Registers. The SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 11 shows the format of each write register.

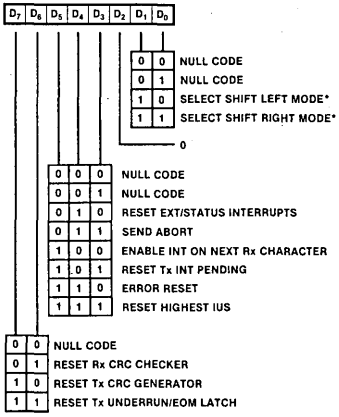
Programming
(Continued)

Write Register 0 (Z8530)



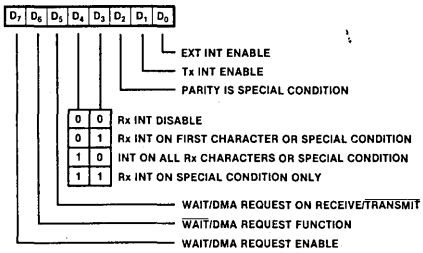
*WITH POINT HIGH COMMAND

Write Register 0 (Z8030)

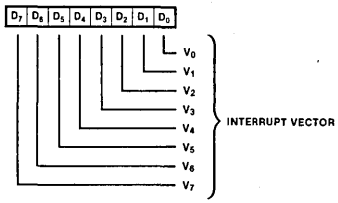


* B CHANNEL ONLY

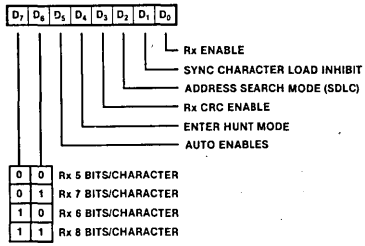
Write Register 1



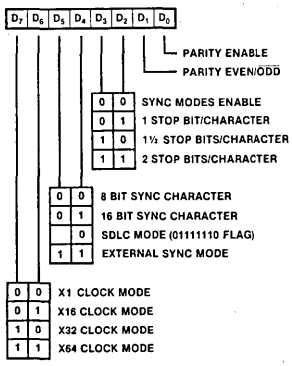
Write Register 2



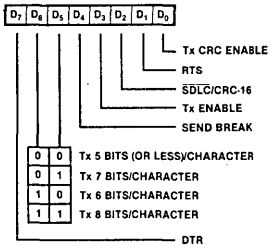
Write Register 3



Write Register 4



Write Register 5



Write Register 6

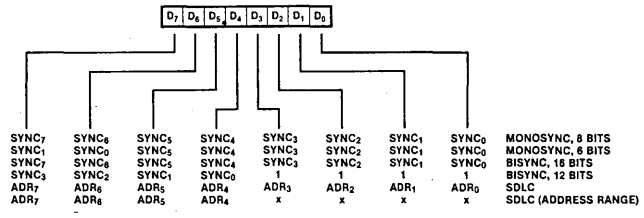
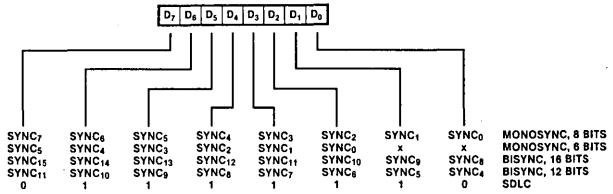
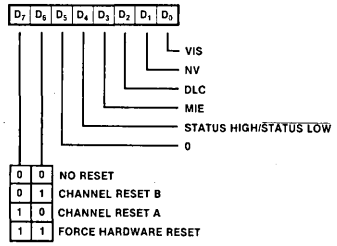


Figure 11. Write Register Bit Functions

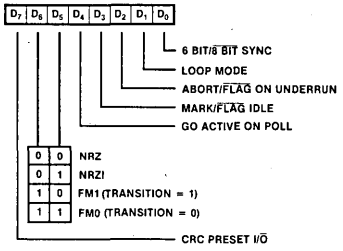
Write Register 7



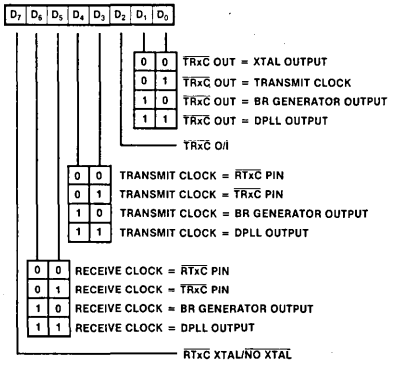
Write Register 9



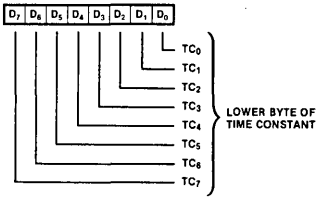
Write Register 10



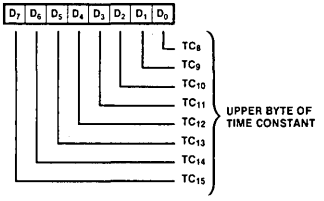
Write Register 11



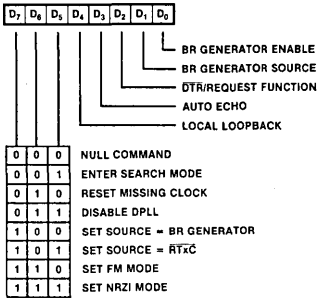
Write Register 12



Write Register 13



Write Register 14



Write Register 15

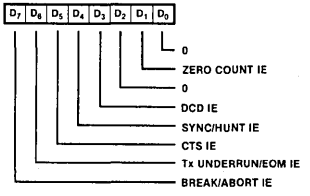


Figure 11. Write Register Bit Functions (Continued)

Z8530 Timing

The SCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to PCLK. Since PCLK has no phase relationship with WR and RD, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the falling edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the SCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the SCC. This time must be at least 4 PCLK regardless of which register or channel is being accessed.

Read Cycle Timing. Figure 12 illustrates Read cycle timing. Addresses on A/B and D/C and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{RD}$ falls or if it rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

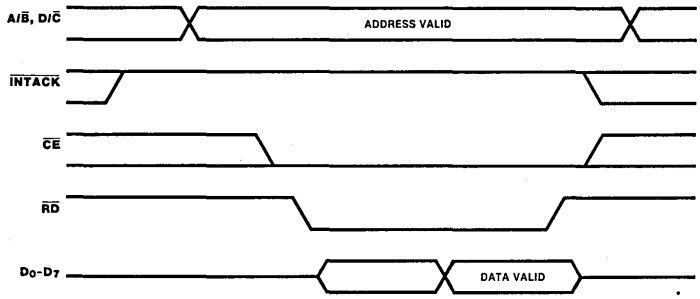


Figure 12. Read Cycle Timing

Write Cycle Timing. Figure 13 illustrates Write cycle timing. Addresses on A/B and D/C and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{WR}$ falls or if it rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened. Data must be valid before the falling edge of $\overline{WR}$.

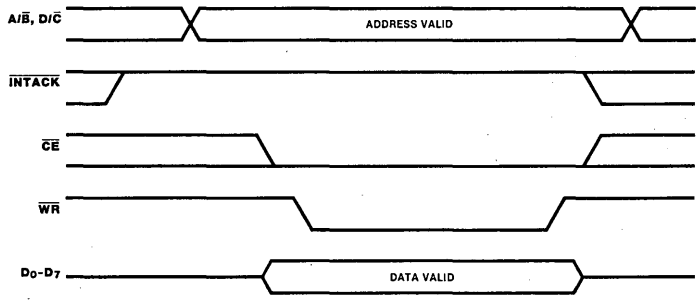


Figure 13. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 14 illustrates Interrupt Acknowledge cycle timing. Between the time $\overline{INTACK}$ goes Low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{RD}$ falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0–D_7 and it then sets the appropriate Interrupt-UnderService latch internally.

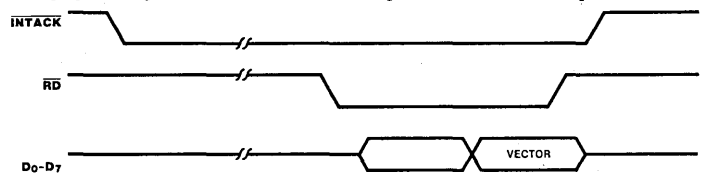


Figure 14. Interrupt Acknowledge Cycle Timing

The SCC generates internal control signals from $\overline{AS}$ and $\overline{DS}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{AS}$ and $\overline{DS}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the falling edge of $\overline{DS}$ in the first transaction involving the SCC to the falling edge of $\overline{DS}$ in the second transaction involving the SCC.

Read Cycle Timing. Figure 15 illustrates Read cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of AS. R/$\overline{W}$ must be High to indicate a Read cycle. CS_1 must also be High for the Read cycle to occur. The data bus drivers in the SCC are then enabled while $\overline{DS}$ is Low.

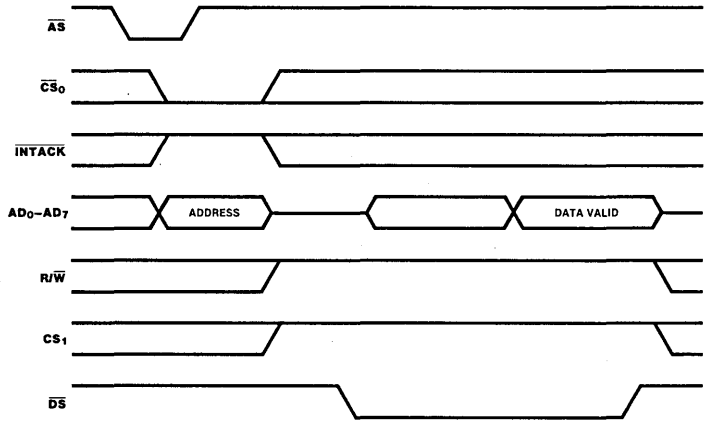


Figure 15. Read Cycle Timing

Write Cycle Timing. Figure 16 illustrates Write cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be Low to indicate a Write cycle. CS_1 must be High for the Write cycle to occur. $\overline{DS}$ Low strobes the data into the SCC.

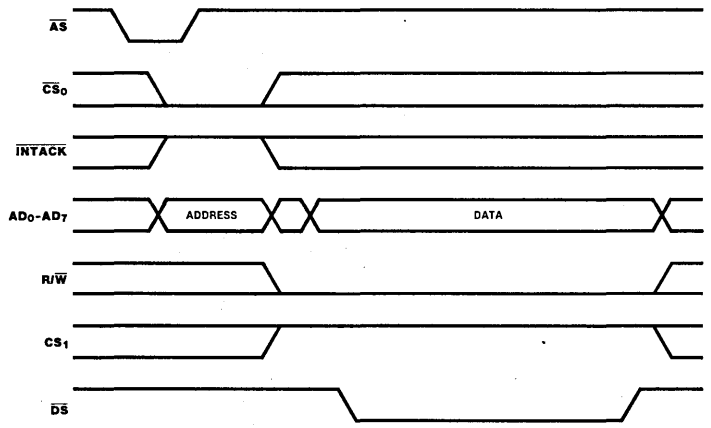


Figure 16. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 17 illustrates Interrupt Acknowledge cycle timing. The address on AD_0-AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. However, if $\overline{INTACK}$ is Low, the address and $\overline{CS_0}$ are ignored. The state of the R/$\overline{W}$ and CS_1 are also ignored for the duration of the Interrupt Acknowledge cycle. Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{DS}$ falls, the Acknowledge cycle was intended for the SCC. In this case, the SCC may be programmed to respond to RD Low by placing its interrupt vector on D_0-D_7 and it then internally sets the appropriate Interrupt-Under-Service latch.

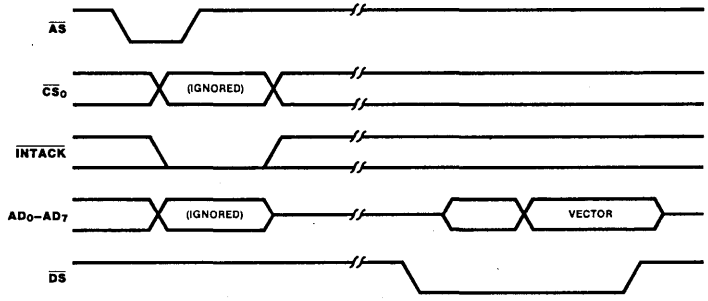


Figure 17. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all pins with respect to GND ..... −0.3V to +7.0V
Operating Ambient Temperature ..... See Ordering Information
Storage Temperature ..... −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

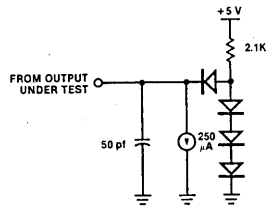
Standard Test Conditions

The DC characteristics and capacitance section below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

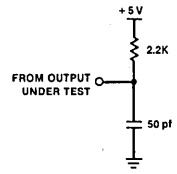
Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.



Standard Test Load



Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		± 10.0	μA	$0.4 \leq V_{IN} \leq +2.4V$
I_{OL}	Output Leakage		± 10.0	μA	$0.4 \leq V_{OUT} \leq +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

$V_{CC} = 5\ V \pm 5\%$ unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.
Unmeasured pins returned to ground.

Miscellaneous

Gate Count	6000

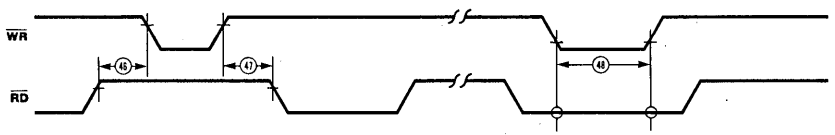
Z8530 AC CHARACTERISTICS

Number	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Notes †
1	TwPCl	PCLK Low Width	105	2000	70	1000	50	1000	
2	TwPCh	PCLK High Width	105	2000	70	1000	50	1000	
3	TfPC	PCLK Fall Time		20		10		10	
4	TrPC	PCLK Rise Time		20		10		10	
5	TcPC	PCLK Cycle Time	250	4000	165	2000	125	2000	
6	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		80		70		
7	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		0		
8	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		80		70		
9	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		0		
10	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	10		10		10		
11	TsIAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		160		145		1
12	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		0		
13	TsIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		160		145		1
14	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		0		
15	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	100		100		85		
16	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		0		
17	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		0		
18	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		70		60		
19	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		0		1
20	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		0		1
21	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		70		60		1
22	TwRDl	$\overline{RD}$ Low Width	240		200		150		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		0		
24	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		0		
25	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		250		180		140	
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70		45		40	2

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a ±0.5V change at the output with a maximum dc load and minimum ac load.
†Units in nanoseconds (ns).

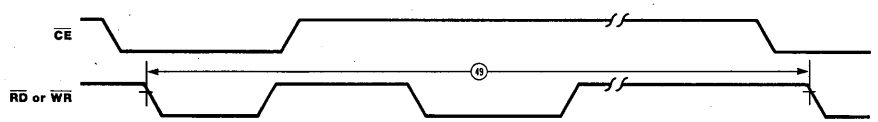
Reset Timing

Z8530



Cycle Timing

Z8530



Z8530 AC CHARACTERISTICS (Continued)

Number	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Notes †
27	TdA(DR)	Address Required Valid to Read Data Valid Delay		300		280		220	
28	TwWRI	$\overline{WR}$ Low Width	240		200		150		
29	TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time	10		10		10		
30	ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time	0		0		0		
31	TdWR(W)	$\overline{WR}$ ↓ to Wait Valid Delay		240		200		170	4
32	TdRD(W)	$\overline{RD}$ ↓ Wait Valid Delay		240		200		170	4
33	TdWRf(REQ)	$\overline{WR}$ ↓ to $\overline{W/REQ}$ Not Valid Delay		240		200		170	
34	TdRDf(REQ)	$\overline{RD}$ ↓ to $\overline{W/REQ}$ Not Valid Delay		240		200		170	
35	TdWRr(REQ)	$\overline{WR}$ ↓ $\overline{DTR/REQ}$ Not Valid Delay		4TcPC		4TcPC		4TcPC	
36	TdRDr(REQ)	$\overline{RD}$ ↑ to $\overline{DTR/REQ}$ Not Valid Delay		4TcPC		4TcPC		4TcPC	
37	TdPC(INT)	PCLK ↓ to $\overline{INT}$ Valid Delay		500		500		500	4
38	TdIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Delay	250		200		150		5
39	TwRDA	$\overline{RD}$ (Acknowledge) Width	250		200		150		
40	TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay		250		180		140	
41	TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time	120		100		95		
42	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time	0		0		0		
43	TdIEI(IEO)	IEI to IEO Delay Time		120		100		95	
44	TdPC(IEO)	PCLK ↑ to IEO Delay		250		250		200	
45	TdRDA(INT)	$\overline{RD}$ ↓ to $\overline{INT}$ Inactive Delay		500		500		450	4
46	TdRD(WRQ)	$\overline{RD}$ ↑ to $\overline{WR}$ ↓ Delay for No Reset	30		15		15		
47	TdWRQ(RD)	$\overline{WR}$ ↑ to $\overline{RD}$ ↓ Delay for No Reset	30		30		20		
48	TwRES	$\overline{WR}$ and $\overline{RD}$ Coincident Low for Reset	250		200		150		
49	Trc	Valid Access Recovery Time	4TcPC		4TcPC		4TcPC		3

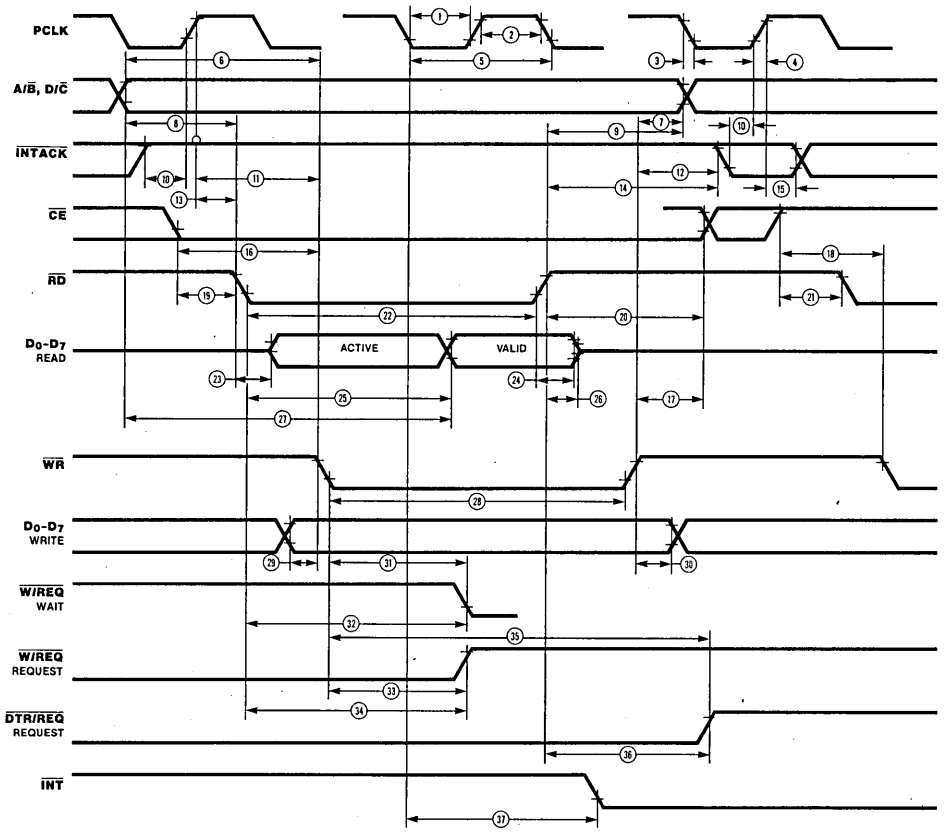
NOTES:

3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.

†Units in nanoseconds (ns).

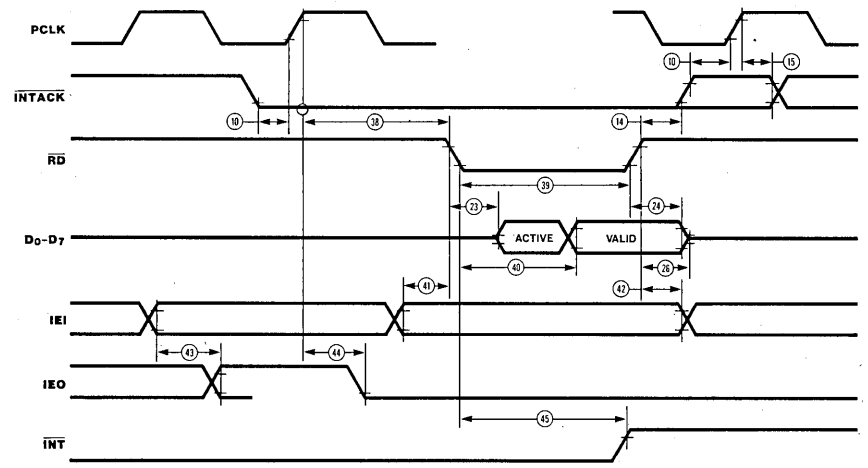
Read and Write Timing

Z8530



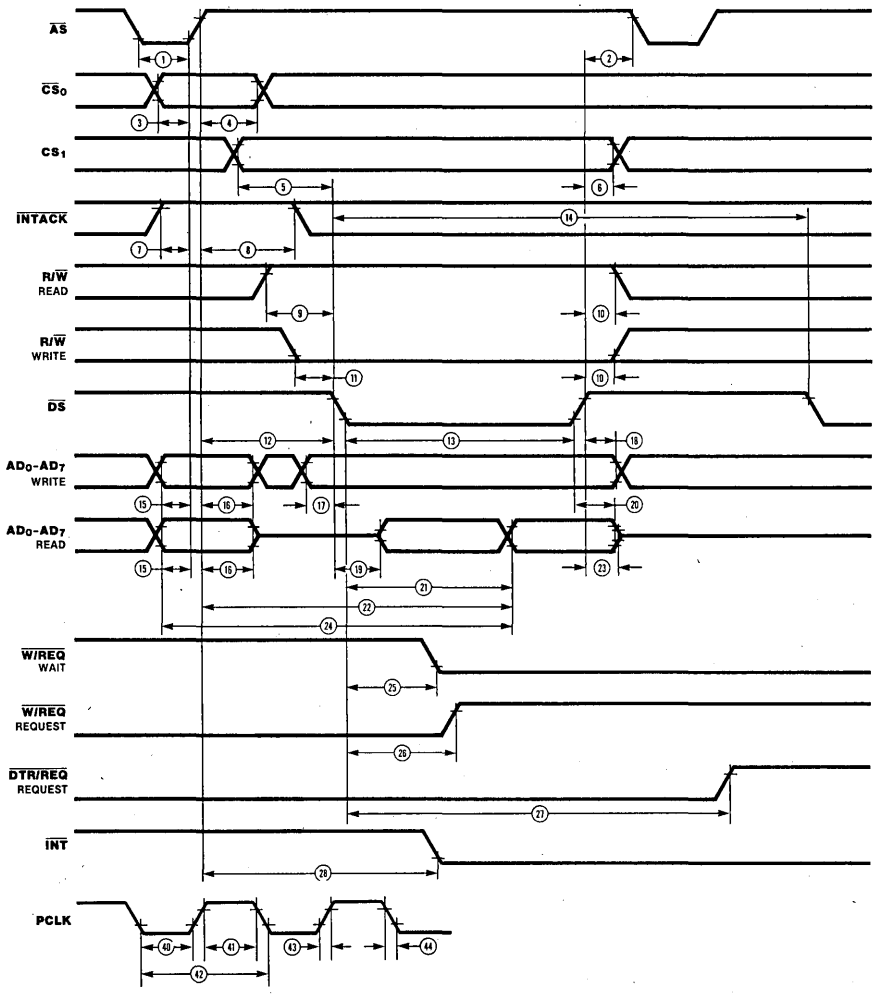
Interrupt Acknowledge Timing

Z8530



Read and Write Timing

Z8030



Z8030 AC CHARACTERISTICS

Number	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Notes †
1	TwAS	$\overline{AS}$ Low Width	70		50		35		
2	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		15		
3	TsCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	0		0		0		1
4	ThCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	60		40		30		1
5	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		80		65		1
6	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	55		40		30		1
7	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	10		10		10		
8	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		200		150		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		65		
10	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		35		
11	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		0		
12	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		40		30		
13	TwDSl	$\overline{DS}$ Low Width	240		200		150		
14	TrC	Valid Access Recovery Time	4TcPC		4TcPC		4TcPC		2
15	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		10		1
16	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		25		1
17	TsDW(DS)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		15		
18	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		20		
19	TdDS(DA)	$\overline{DS}$ ↓ to Data Active Delay	0		0		0		
20	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		0		
21	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180		140	
22	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Valid Delay		520		300		250	

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Parameter applies only between transactions involving the SCC.

†Units in nanoseconds (ns).

Z8030 AC CHARACTERISTICS (Continued)

Number	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Notes †
23	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45		40	3
24	TdA(DR)	Address Required Valid to Read Data Valid Delay		570		310		260	
25	TdDS(W)	$\overline{DS}$ ↓ to Wait Valid Delay		240		200		170	4
26	TdDSf(REQ)	$\overline{DS}$ ↓ to $\overline{W/REQ}$ Not Valid Delay		240		200		170	
27	TdDSr(REQ)	$\overline{DS}$ ↓ to $\overline{DTR/REQ}$ Not Valid Delay		4TcPC		4TcPC		4TcPC	
28	TdAS(INT)	$\overline{AS}$ ↑ to $\overline{INT}$ Valid Delay		500		500		500	4
29	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay	250		250		250		5
30	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	390		200		150		
31	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		250		180		140	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	120		100		80		
33	ThIEI(DSA)	IEI to $\overline{DS}$ ↑ (Acknowledge) Hold Time	0		0		0		
34	TdIEI(IEO)	IEI to IEO Delay		120		100		90	
35	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		250		250		200	6
36	TdDSA(INT)	$\overline{DS}$ ↓ (Acknowledge) to $\overline{INT}$ Inactive Delay		500		500		450	4
37	TdDS(ASQ)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay for No Reset	30		15		15		
38	TdASQ(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay for No Reset	30		30		20		
39	TwRES	$\overline{AS}$ and $\overline{DS}$ Coincident Low for Reset	250		200		150		7
40	TwPCl	PCLK Low Width	105	2000	70	1000	50		
41	TwPCh	PCLK High Width	105	2000	70	1000	50		
42	TcPC	PCLK Cycle Time	250	4000	165	2000	125		
43	TrPC	PCLK Rise Time		20		10		10	
44	TfPC	PCLK Fall Time		20		10		10	

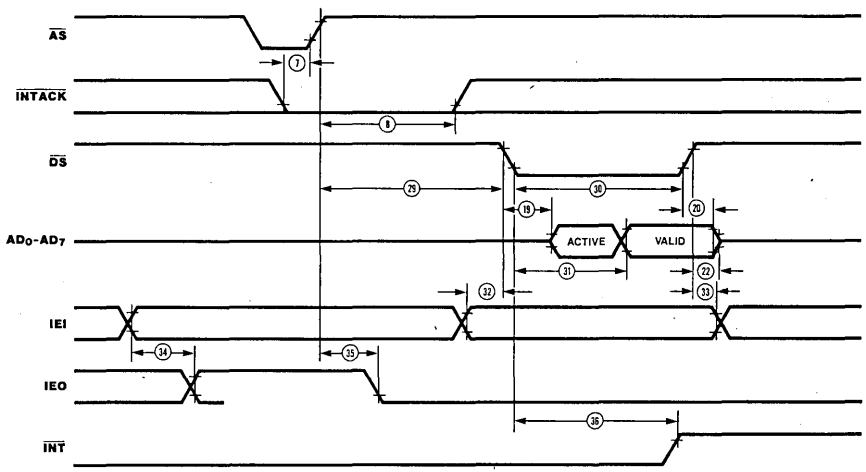
NOTES:

3. Float delay is defined as the time required for a ±0.5V change in the output with a maximum dc load and a minimum ac load.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any Z-SCC in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the highest priority device in the daisy chain, TsIEI(DSA) for the Z-SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.
6. Parameter applies only to a Z-SCC pulling INT Low at the beginning of the Interrupt Acknowledge transaction.
7. Internal circuitry allows for the reset provided by the Z8 to be recognized as a reset by the Z-SCC.
   All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".

†Units in nanoseconds (ns).

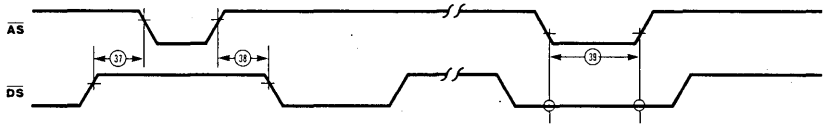
Interrupt Acknowledge Timing

Z8030



Reset Timing

Z8030



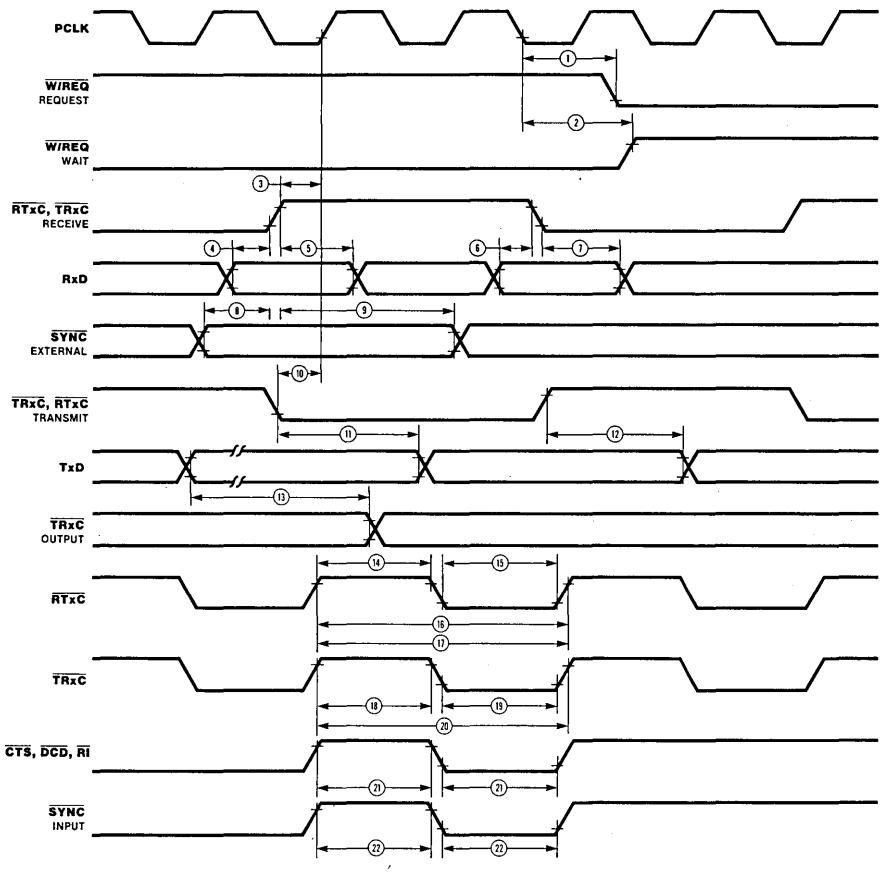
Z8030/Z8530 GENERAL TIMING AC CHARACTERISTICS

Number	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Notes †
1	TdPC(REQ)	PCLK ↓ to W/$\overline{REQ}$ Valid Delay		250		250		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350		350		350	
3	TsRXC(PC)	$\overline{RxC}$ ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	80	TwPCL	70	TwPCL	60	TwPCL	1,4
4	TsRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Setup Time (X1 Mode)	0		0		0		1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode)	150		150		150		1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Setup Time (X1 Mode)	0		0		0		1,5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode)	150		150		150		1,5
8	TsSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Setup Time	−200		−200		−200		1
9	ThSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Hold Time	3TcPC +400		3TcPC +320		3TcPC +250		1
10	TsTXC(PC)	$\overline{TxC}$ ↓ to PCLK ↑ Setup Time	0		0		0		2,4
11	TdTXCf(TXD)	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode)		300		230		200	2
12	TdTxCr(TXD)	$\overline{TxC}$ ↑ to TxD Delay (X1 Mode)		300		230		200	2,5
13	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)		200		200		200	
14	TwRTXh	$\overline{RTxC}$ High Width	180		180		150		6
15	TwRTXl	$\overline{RTxC}$ Low Width	180		180		150		6
16	TcRTX	$\overline{RTxC}$ Cycle Time (RxD, TxD)	1000		640		500		6,7
17	TcRTXX	Crystal Oscillator Period	250	1000	165	1000	125	1000	3
18	TwTRXh	$\overline{TRxC}$ High Width	180		180		150		6
19	TwTRXl	$\overline{TRxC}$ Low Width	180		180		150		6
20	TcTRX	$\overline{TRxC}$ Cycle Time	1000		640		500		6,7
21	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ Pulse Width	200		200		200		
22	TwSY	$\overline{SYNC}$ Pulse Width	200		200		200		

NOTES:
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both $\overline{RTxC}$ and $\overline{SYNC}$ have 30 pf capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to case PCLK requirements.
7. The maximum receive or transmit data is ¼ PCLK.

†Units in nanoseconds (ns).

General Timing

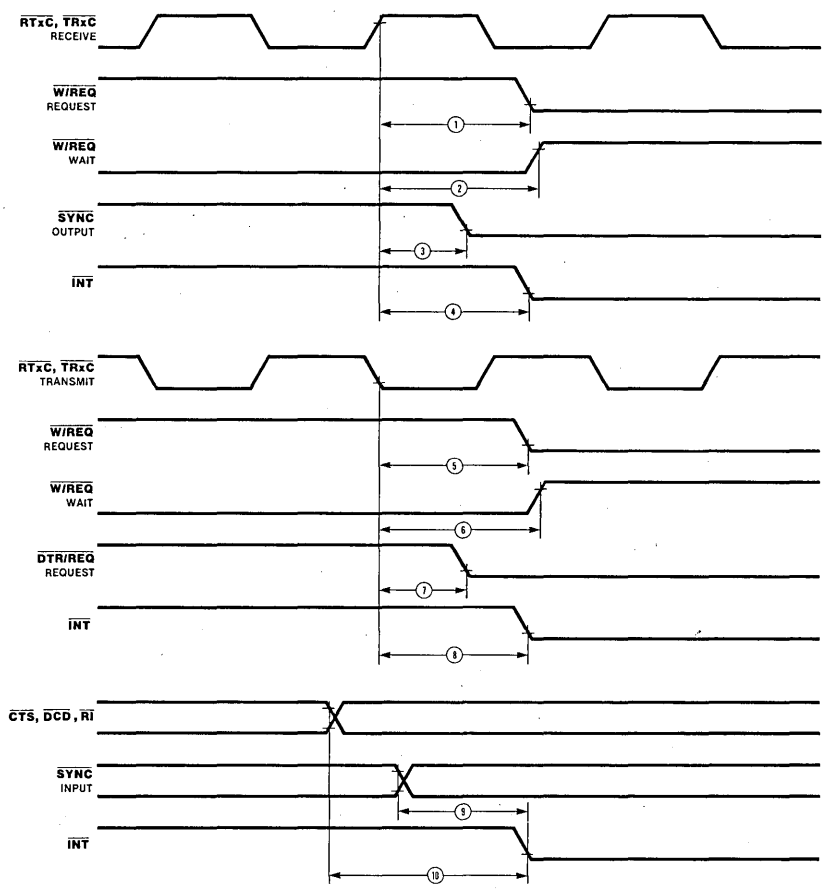


Z8030/Z8530 SYSTEM TIMING AC CHARACTERISTICS

Number	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Notes †
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to W/$\overline{REQ}$ Valid Delay	8	12	8	12	8	12	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	14	8	14	8	14	1,2
3	TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	4	7	4	7	4	7	2
4a.	TdRXC(INT), Z8530	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	10	16	10	16	10	16	1,2
4b.	TdRXC(INT), Z8030		8	12	8	12	8	12	1,2
			+2	+3	+2	+3	+2	+3	4
5	TdTXC(REQ)	$\overline{TxC}$ ↓ to W/$\overline{REQ}$ Valid Delay	5	8	5	8	5	8	3
6	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	11	5	11	5	11	1,3
7	TdTXC(DRQ)	$\overline{TxC}$ ↓ DTR/$\overline{REQ}$ Valid Delay	4	7	4	7	4	7	3
8a.	TdTXC(INT), Z8530	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	6	10	6	10	6	10	1,3
8b.	TdTXC(INT), Z8030		4	6	4	6	4	6	1,3
			+2	+3	+2	+3	+2	+3	4
9a.	TdSY(INT), Z8530	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	2	6	1
9b.	TdSY(INT), Z8030		2	3	2	3	2	3	1,4
10a.	TdEXT(INT), Z8530	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	2	6	1
10b.	TdEXT(INT), Z8030		2	3	2	3	2	3	1,4

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
4. Units equal to $\overline{AS}$.

†Units equal to TcPC.


RTxC, TRxC
RECEIVE
W/REQ
REQUEST
W/REQ
WAIT
SYNC
OUTPUT
INT
1
2
3
4
RTxC, TRxC
TRANSMIT
W/REQ
REQUEST
W/REQ
WAIT
DTR/REQ
REQUEST
INT
5
6
7
8
CTS, DCD, RI
SYNC
INPUT
INT
9
10

Z8036 Z8000® Z-CIO Counter/Timer and Parallel I/O Unit

October 1988

Features

- Two independent 8-bit, double-buffered, bidirectional I/O ports plus a 4-bit special-purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/$\overline{\text{WAIT}}$ signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or nonretriggerable.
- Easy to use since all registers are read/write and directly addressable.

General Description

The Z8036 Z-CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications.

The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique address so that it can be accessed directly—no special sequential operations are required. The Z-CIO is directly Z-Bus compatible.

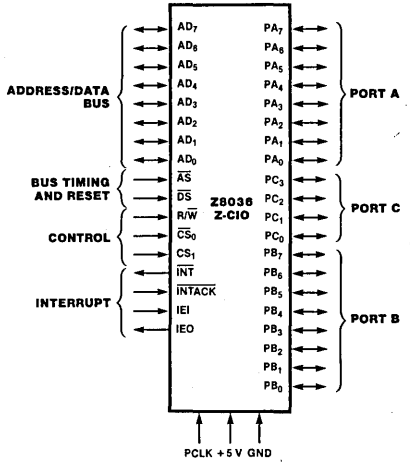


Figure 1. Pin Functions

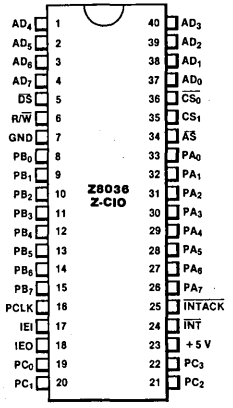


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

Pin Description

AD_0-AD_7. *Z-Bus Address/Data lines* (bidirectional/3-state). These multiplexed Address/Data lines are used for transfers between the CPU and Z-CIO.

$\overline{AS}$*. *Address Strobe* (input, active Low). Addresses, $\overline{INTACK}$, and $\overline{CS_0}$ are sampled while $\overline{AS}$ is Low.

$\overline{CS_0}$ and CS_1. *Chip Select 0* (input, active Low) *and Chip Select 1* (input, active High). $\overline{CS_0}$ and CS_1 must be Low and High, respectively, in order to select a device. $\overline{CS_0}$ is latched by $\overline{AS}$.

$\overline{DS}$*. *Data Strobe* (input, active Low). $\overline{DS}$ provides timing for the transfer of data into or out of the Z-CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting Z-CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

*When $\overline{AS}$ and $\overline{DS}$ are detected Low at the same time (normally an illegal condition), the Z-CIO is reset.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the Z-CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates to the Z-CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ is sampled while $\overline{AS}$ is Low.

PA_0-PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the Z-CIO's Port A and external devices.

PB_0-PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the Z-CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0-PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the Z-CIO's Port C.

PCLK. *(input, TTL-compatible).* This is a peripheral clock that may be, but is not necessarily, the CPU clock. It is used with timers and REQUEST/$\overline{WAIT}$ logic.

R/$\overline{W}$. *Read/Write* (input). R/$\overline{W}$ indicates that the CPU is reading from (High) or writing to (Low) the Z-CIO.

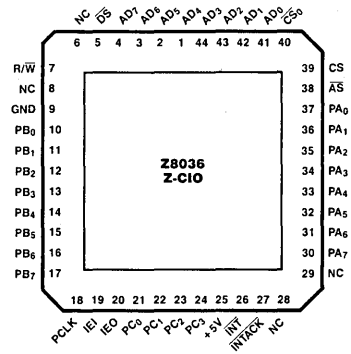


Figure 2b. 44-pin Chip Carrier, Pin Assignments

Architecture

The Z8036 Z-CIO Counter/Timer and Parallel I/O element (Figure 3) consists of a Z-Bus interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose

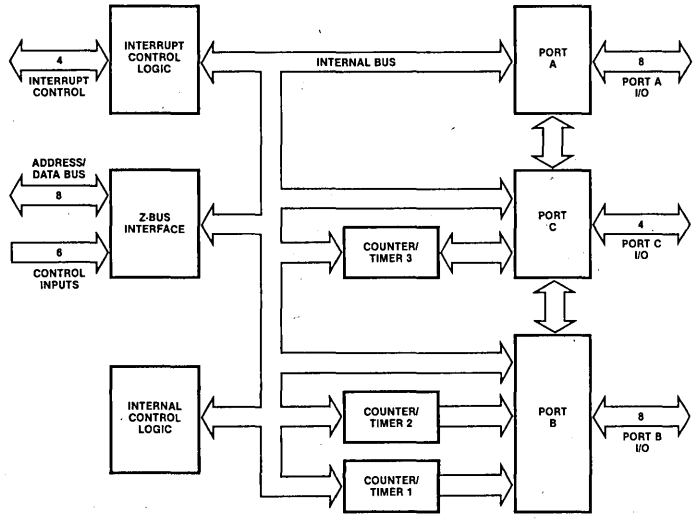


Figure 3. Z-CIO Block Diagram

Architecture (Continued)

4-bit port), three 16-bit counter/timers, an interrupt control logic block, and the internal control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

The two general-purpose 8-bit I/O ports (Figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers: the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

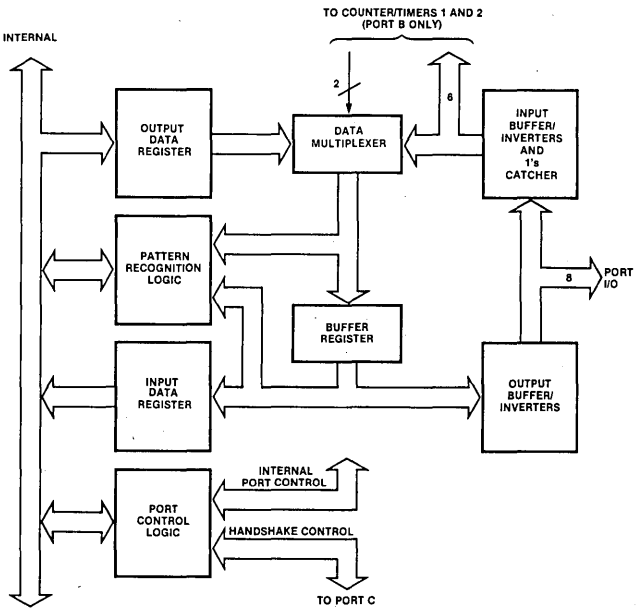


Figure 4. Ports A and B Block Diagram

Architecture (Continued)

The function of the special-purpose 4-bit port, Port C (Figure 5), depends upon the roles of Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path: the Data Path Polarity, Data Direction, and Special I/O Control registers.

The three counter/timers (Figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Counter register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer: counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or nonretriggerable. With these and other options, most counter/timer applications are covered.

The interrupt control logic provides standard Z-Bus interrupt capabilities. There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

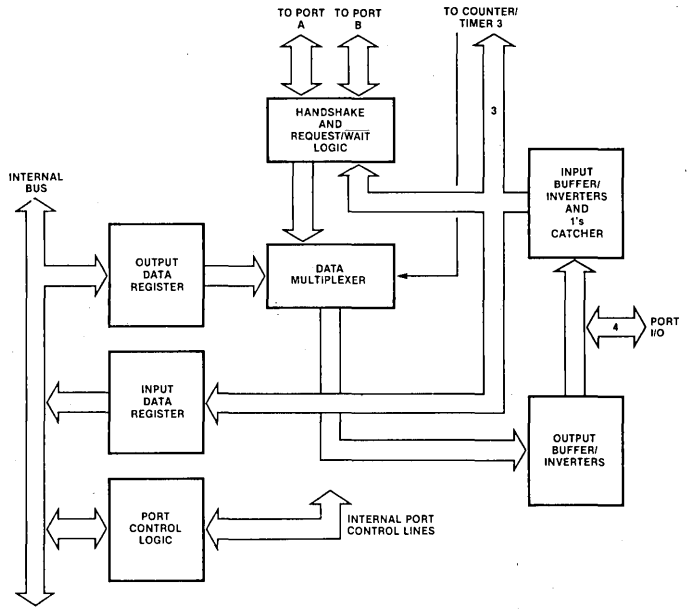


Figure 5. Port C Block Diagram

Architecture (Continued)

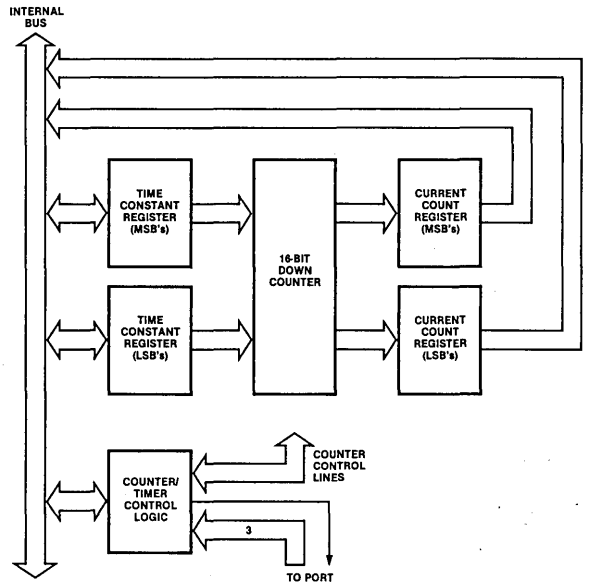


Figure 6. Counter/Timer Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the Z-CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/$\overline{\text{WAIT}}$ line can also be provided so that Z-CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/$\overline{\text{WAIT}}$ can be used as input or output bits (individually data direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to a 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Functional Description (Continued)

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The Z-CIO provides four different handshakes for its ports: Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in Table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3.

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked, Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (Table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the Z-CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the Z-CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another Z-CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{DAV}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{DAV}$ is

Port A/B Configuration	PC_3	PC_2	PC_1	PC_0
Ports A and B: Bit Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O
Port B: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O	RFD or $\overline{DAV}$	$\overline{ACKIN}$
Port A or B: Input Port (3-Wire Handshake)	RFD (Output)	$\overline{DAV}$ (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	DAC (Output)
Port A or B: Output Port (3-Wire Handshake)	$\overline{DAV}$ (Output)	DAC (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	RFD (Input)
Port A or B: Bidirectional Port (Interlocked or Strobed Handshake)	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	IN/$\overline{OUT}$

*Both Ports A and B can be specified input or output with Interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{WAIT}$.

Table 1. Port C Bit Utilization

Functional Description (Continued)

allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{DAV}$ goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input ($\overline{ACKIN}$) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked Handshake, the signal indicating the port is ready for another data transfer operates independently of the $\overline{ACKIN}$ input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (Figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (Figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the $\overline{ACKIN}$ path. The external $\overline{ACKIN}$ input triggers the timer and its output is used as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{DAV}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{DAV}$ output goes Low and the timer output is used as the actual $\overline{DAV}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

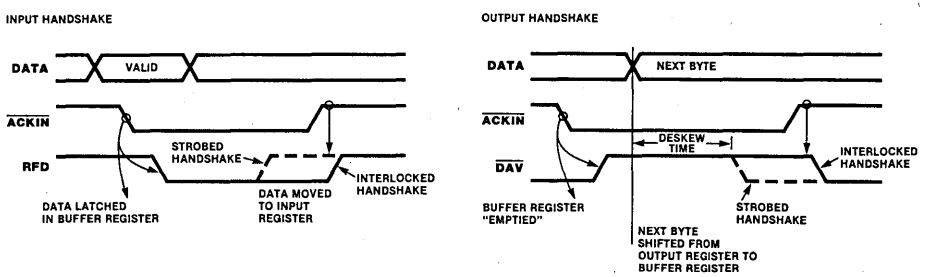


Figure 7. Interlocked and Strobed Handshakes

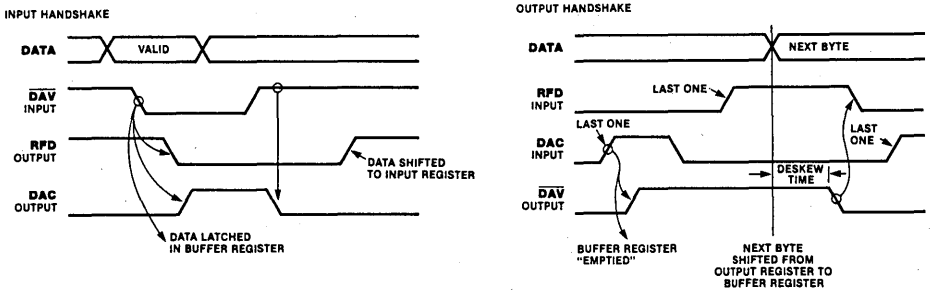


Figure 8. 3-Wire Handshake

Functional Description (Continued)

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the Z-Bus. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the Z-CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all nonmasked bit specifications in the AND mode or the satisfaction of any nonmasked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

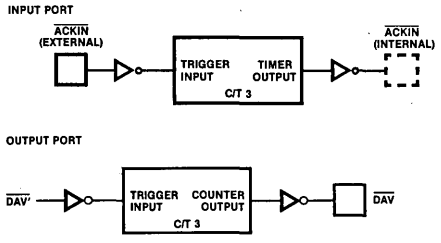


Figure 9. Pulsed Handshake

The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt. Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes: Transparent and Latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1, after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. Once the Acknowledge cycle is initiated, the vector is

Functional Description (Continued)

frozen until the corresponding IP is cleared. Where inputs that cause interrupts might change before the interrupt is serviced, the 1's catcher can be used to hold the value. Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register to the Input register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern-match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type applications when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

Table 2. Counter/Timer External Access

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four least significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (Table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output: pulse, one-shot, and square-wave. Figure 10 shows the counter/

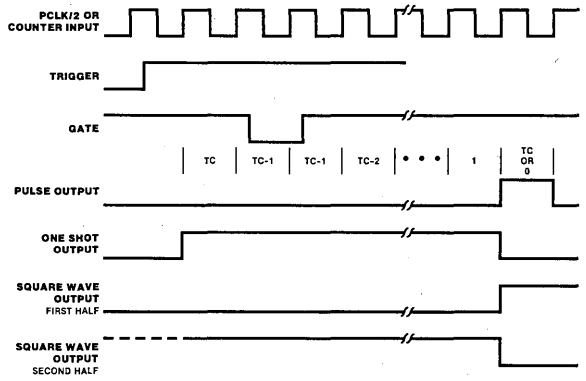


Figure 10. Counter/Timer Waveforms

Functional Description (Continued)

timer waveforms. When the Pulse mode is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal countdown sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/$\overline{\text{Single}}$ $\overline{\text{Cycle}}$ (C/$\overline{\text{SC}}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{\text{SC}}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{\text{SC}}$ bit is 1 each time the countdown counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65,536. The down-counter is loaded in one of three ways: by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal output if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Once the down-counter is loaded, the countdown sequence continues toward terminal count as long as all the counter/timers' hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a countdown sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the countdown sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to a 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways: by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The interrupts generated by the Z-CIO follow the Z-Bus operation as described more fully in the Zilog *Z-Bus Summary*. The Z-CIO has five potential sources of interrupts: the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order: Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The Z-CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device,

Functional Description (Continued)

the Z-CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the Z-CIO contains three bits for the control and status of the interrupt logic: an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS) status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{INT}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. It can also be set directly by the CPU. Its primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, an interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service; however, the $\overline{INT}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the Z-CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the Z-CIO's IEO is forced Low, independent of the state of the Z-CIO or its IEI input, and all lower priority devices' interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the Z-CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The Z-CIO contains three vector registers: one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1. The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the Z-CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

Programming

Programming the Z-CIO entails loading control registers with bits to implement the desired operation. Individual enable bits are provided for the various major blocks so that erroneous operations do not occur while the part is being initialized. Before the ports are enabled, IPs cannot be set, REQUEST and $\overline{WAIT}$ cannot be asserted, and all outputs remain high-impedance. The handshake lines are ignored until Port C is enabled. The counter/timers cannot be triggered until their enable bits are set.

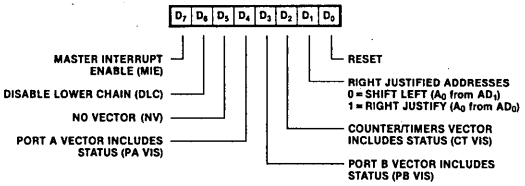
The Z-CIO is reset by forcing $\overline{AS}$ and $\overline{DS}$ Low simultaneously or by writing a 1 to the Reset bit. Once reset, the only thing that can be done is to read and write the Reset bit. Writes to all other bits are ignored and all reads return 0s. In this state, all control bits are forced to 0. Only after clearing the Reset bit (by writing to it) can the other command bits be programmed.

Register Addressing. The Z-CIO allows two schemes for register addressing. Both schemes use only six of the eight bits of the address/data bus. The scheme used is determined by the Right Justify Address (RJA) bit in the Master Interrupt Control register. When RJA equals 0, address bus bits 0 and 7 are ignored, and bits 1 through 6 are decoded for the register address (A_0 from AD_1). When RJA equals 1, bits 0 through 5 are decoded for the register address (A_0 from AD_0). In the following register descriptions, only six bits are shown for addresses and represent address/data bus bits 0 through 5 or 1 through 6, depending on the state of the RJA bit.

Registers

Master Interrupt Control Register

Address: 000000
(Read/Write)



Master Configuration Control Register

Address: 000001
(Read/Write)

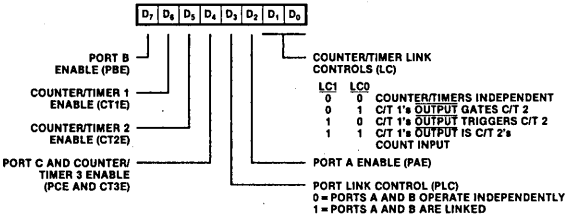
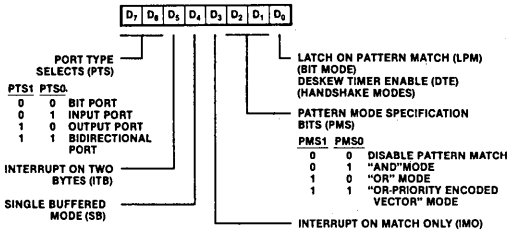


Figure 11. Master Control Registers

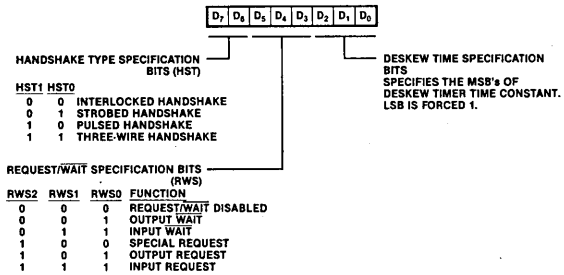
Port Mode Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)



Port Handshake Specification Registers

Addresses: 100001 Port A
101001 Port B
(Read/Write)



Port Command and Status Registers

Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)

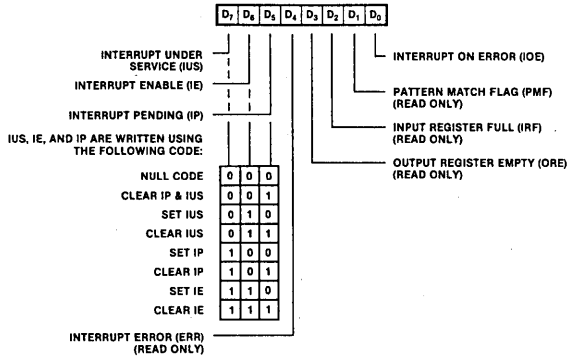
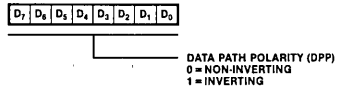


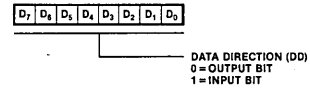
Figure 12. Port Specification Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

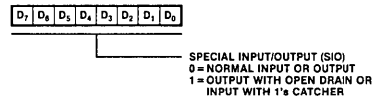
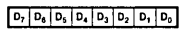


Figure 13. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A
001110 Port B
(Read/Write)



Port C Data Register
Address: 001111
(Read/Write)

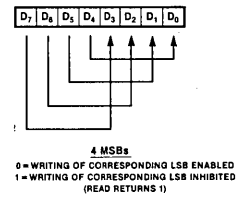
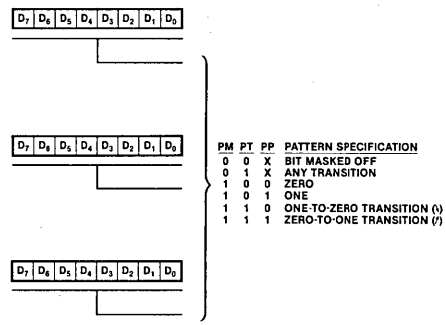


Figure 14. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)

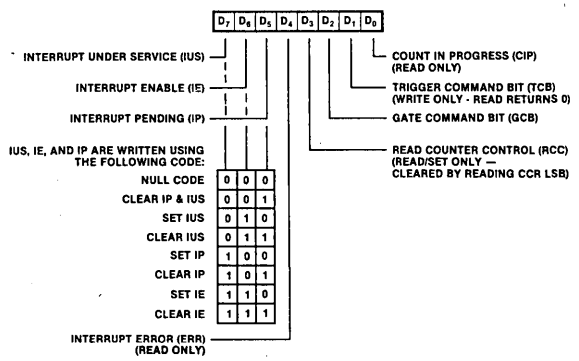


PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE-TO-ZERO TRANSITION (\)
1	1	1	ZERO-TO-ONE TRANSITION (/)

Figure 15. Pattern Definition Registers

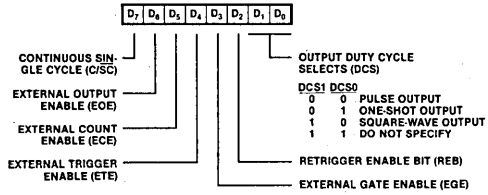
Counter/Timer Command and Status Registers

Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)



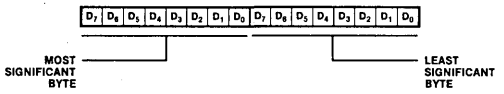
Counter/Timer Mode Specification Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)



Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

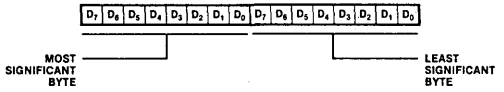
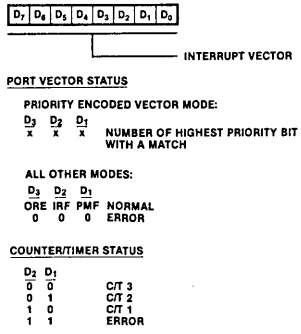


Figure 16. Conter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



Current Vector Register
Address: 011111
(Read Only)

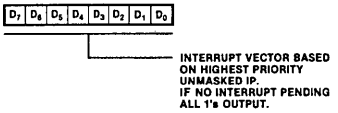


Figure 17. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address*	Register Name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Most Often Accessed Registers

Address*	Register Name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data
001110	Port B's Data
001111	Port C's Data

Counter/Timer Related Registers

Address*	Register Name
010000	Counter/Timer 1's Current Count-MSBs
010001	Counter/Timer 1's Current Count-LSBs
010010	Counter/Timer 2's Current Count-MSBs
010011	Counter/Timer 2's Current Count-LSBs
010100	Counter/Timer 3's Current Count-MSBs
010101	Counter/Timer 3's Current Count-LSBs
010110	Counter/Timer 1's Time Constant-MSBs
010111	Counter/Timer 1's Time Constant-LSBs
011000	Counter/Timer 2's Time Constant-MSBs
011001	Counter/Timer 2's Time Constant-LSBs
011010	Counter/Timer 3's Time Constant-MSBs
011011	Counter/Timer 3's Time Constant-LSBs
011100	Counter/Timer 1's Mode Specification
011101	Counter/Timer 2's Mode Specification
011110	Counter/Timer 3's Mode Specification
011111	Current Vector

Port A Specification Registers

Address*	Register Name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Port B Specification Registers

Address*	Register Name
101000	Port B's Mode Specification
101001	Port B's Handshake Specification
101010	Port B's Data Path Polarity
101011	Port B's Data Direction
101100	Port B's Special I/O Control
101101	Port B's Pattern Polarity
101110	Port B's Pattern Transition
101111	Port B's Pattern Mask

*When RJA = 0, A_0 from AD_1; when RJA = 1, A_0 from AD_0

Read Cycle. The CPU places an address on the address/data bus. The more significant bits and status information are combined and decoded by external logic to provide two Chip Selects ($\overline{CS}_0$ and CS_1). Six bits of the least significant byte of the address are latched within the Z-CIO and used to specify a Z-CIO register. The data from the register specified is strobed onto the address/data bus when the CPU issues a Data Strobe ($\overline{DS}$). If the register indicated by the address does not exist, the Z-CIO remains high-impedance.

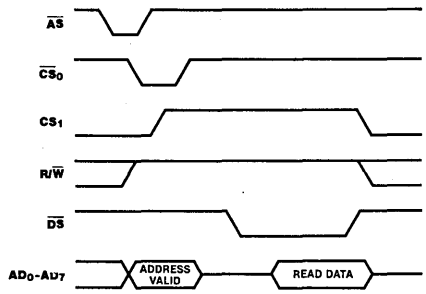


Figure 18. Read Cycle Timing

Write Cycle. The CPU places an address on the address/data bus. The more significant bits and status information are combined and decoded by external logic to provide two Chip Selects ($\overline{CS}_0$ and CS_1). Six bits of the least significant byte of the address are latched within the Z-CIO and used to specify a Z-CIO register. The CPU places the data on the address/data bus and strobes it into the Z-CIO register by issuing a Data Strobe ($\overline{DS}$).

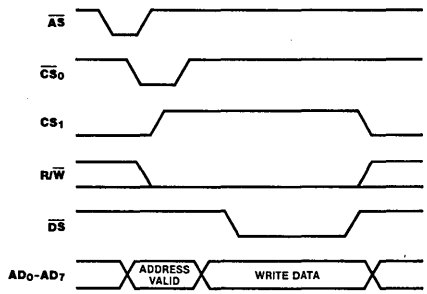
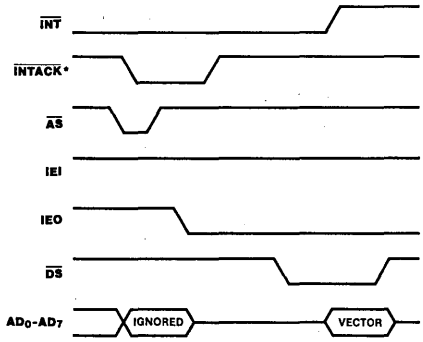


Figure 19. Write Cycle Timing

Interrupt Acknowledge Cycle. When one of the IP bits in the Z-CIO goes High and interrupts are enabled, the Z-CIO pulls its $\overline{INT}$ output line Low, requesting an interrupt. The CPU responds with an Interrupt Acknowledge cycle. When $\overline{INTACK}$ goes Low with IP set, the Z-CIO pulls its Interrupt Enable Out (IEO) Low, disabling all lower priority devices on the daisy chain. The CPU reads the Z-CIO interrupt vector by issuing a Low $\overline{DS}$, thereby strobing the interrupt vector onto the address/data bus. The IUS that corresponds to the IP is also set, which causes IEO to remain Low.



*$\overline{INTACK}$ is decoded from Z8000 status.

Figure 20. Interrupt Acknowledge Timing

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The DC characteristics and capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pf max.

The Ordering Information section lists temperature ranges and product numbers. Package drawings are in the Package Information section in this book. Refer to the Literature List for additional documentation.

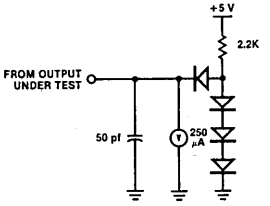


Figure 21. Standard Test Load

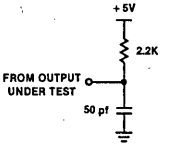


Figure 22. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
			0.5	V	I_{OL} = +3.2 mA
I_{IL}	Input Leakage		±10.0	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{OL}	Output Leakage		±10.0	µA	0.4 ≤ V_{OUT} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		200	mA	

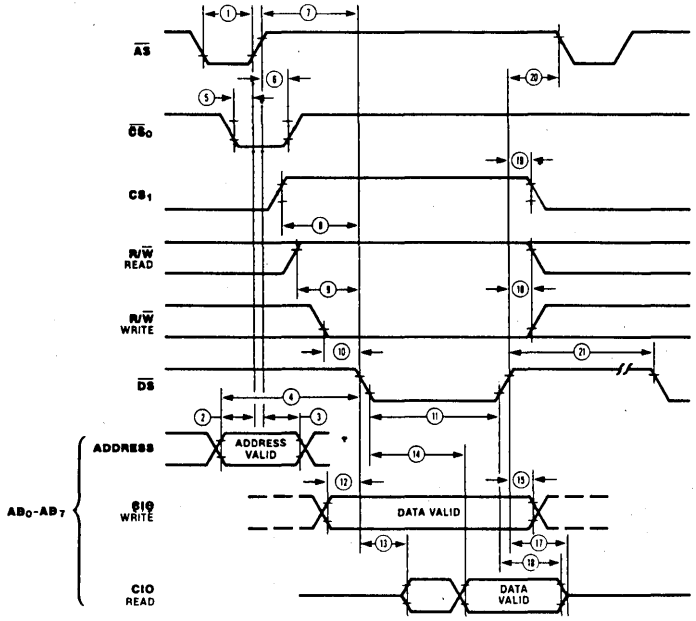
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

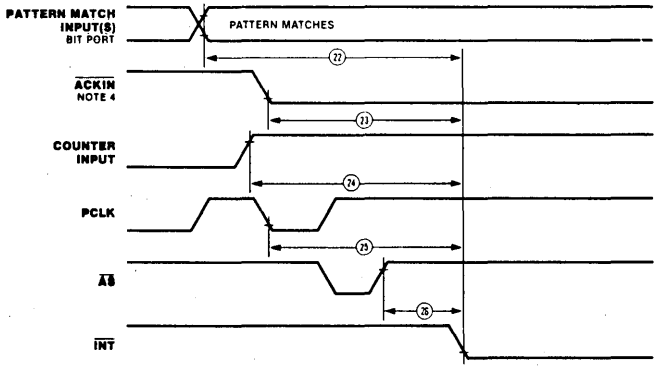
Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pf	
C_{OUT}	Output Capacitance		15	pf	
$C_{I/O}$	Bidirectional Capacitance		20	pf	

f = 1 MHz, over specified temperature range.
Unmeasured pins returned to ground.

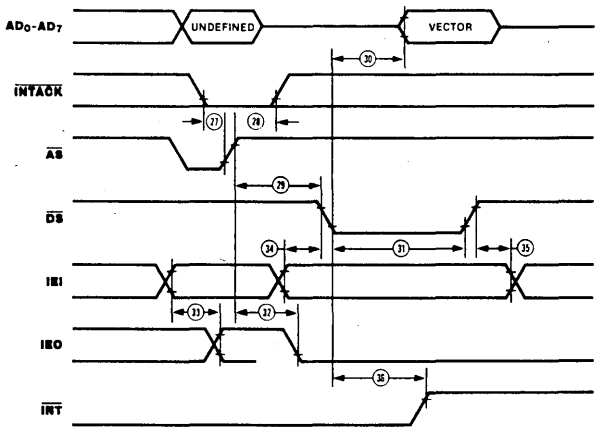
CPU Interface Timing



Interrupt Timing



Interrupt Acknowledge Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwAS	$\overline{AS}$ Low Width	70	2000	50	2000	
2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
4	TsA(DS)	Address to $\overline{DS}$ ↓ Setup Time	180		110		1
5	TsCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	5		5		1
6	ThCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	60		50		1
7	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	150		100		1
8	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		100		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
10	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	100		80		
11	TwDS	$\overline{DS}$ Low Width	500		320		
12	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	50		45		
13	TdDS(DRV)	$\overline{DS}$ (Read) ↓ to Address Data Bus Driven	0		0		
14	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		460		280	
15	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	200		115		
16	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
17	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	2
18	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	150		80		
19	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	150		60		
20	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
21	Trc	Valid Access Recovery Time	1000		650		3
22	TdPM(INT)	Pattern Match to $\overline{INT}$ Delay (Bit Port)		1 + 800		1 + 800	6
23	TdACK(INT)	$\overline{ACKIN}$ to $\overline{INT}$ Delay (Port with Handshake)		4 + 600		4 + 600	4,6
24	TdCI(INT)	Counter Input to $\overline{INT}$ Delay (Counter Mode)		1 + 700		1 + 700	6
25	TdPC(INT)	PCLK to $\overline{INT}$ Delay (Timer Mode)		1 + 700		1 + 700	6
26	TdAS(INT)	$\overline{AS}$ to $\overline{INT}$ Delay		300			
27	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
28	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		140		
29	TsAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Setup Time	900		580		5
30	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		710		440	
31	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	750		480		
32	TdAS(IEO)	$\overline{AS}$ ↑ to IEO ↓ Delay ($\overline{INTACK}$ Cycle)		430		300	5
33	TdIEI(IEO)	IEI to IEO Delay		180		160	5
34	TsIEI(DSA)	IEO to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		70		5
35	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↑ Hold Time	100		70		
36	TdDSA(INT)	$\overline{DS}$ (Acknowledge) ↓ to $\overline{INT}$ ↑ Delay		1280		840	

NOTES:

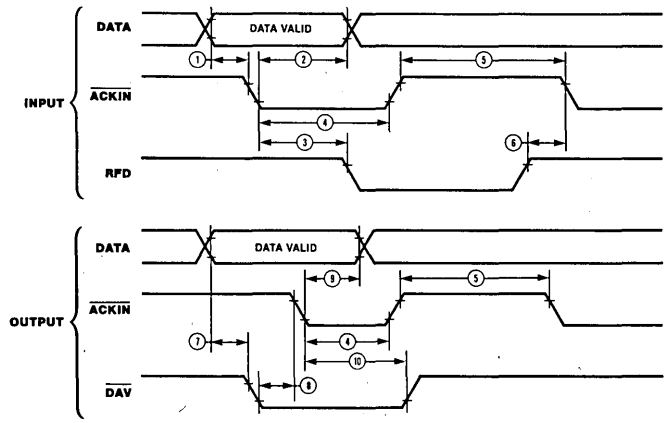
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{DS}$ ↑ of one CIO access to $\overline{DS}$ ↓ of another CIO access.
4. The delay is from DAV ↑ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake. One additional AS cycle is required for ports in the Single Buffered mode.
5. The parameters for the devices in any particular daisy chain must meet the following constraint: the delay from $\overline{AS}$ ↑ to $\overline{DS}$ ↓ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.
6. Units equal to $\overline{AS}$ cycle + ns.

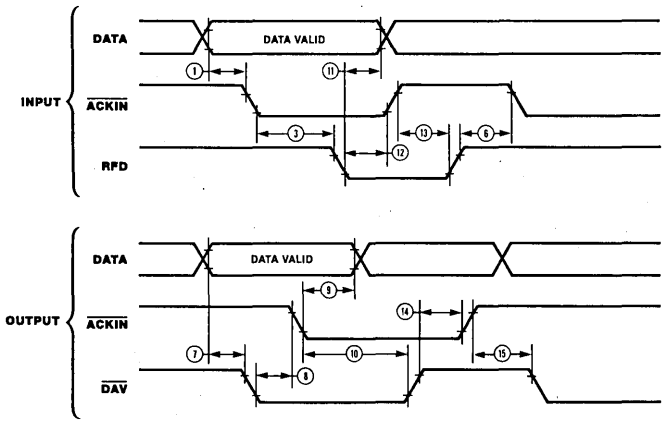
* Timings are preliminary and subject to change.

† Units in nanoseconds(ns), except as noted.

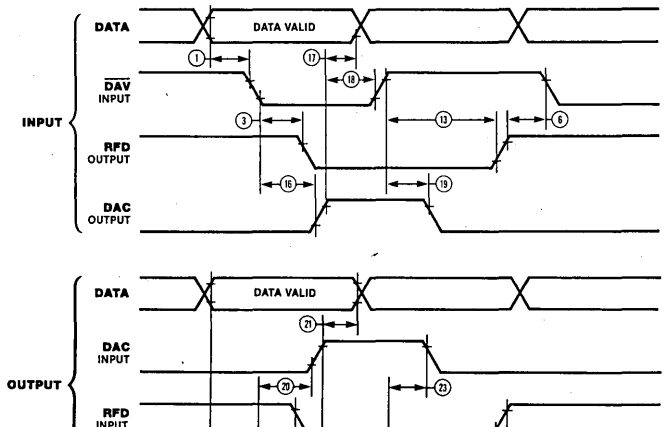
7. The $\overline{AS}$ functions as the clock to the 8036. If $\overline{AS}$ strobe stops, then data does not get clocked through the device. $\overline{AS}$ cycle functions similar to a clock cycle, following $\overline{AS}$ timing specifications. Refer to 7-1 of the Technical Manual.

Interlocked Handshake





3-Wire Handshake

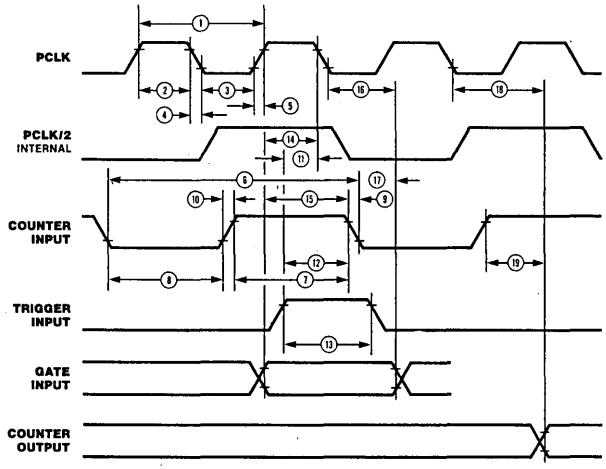


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Setup Time	0		0		
2	ThDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Hold Time - Strobed Handshake	500		330		
3	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		0		
4	TwACKl	$\overline{ACKIN}$ Low Width - Strobed Handshake	250		165		
5	TwACKh	$\overline{ACKIN}$ High Width - Strobed Handshake	250		165		
6	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		0		
7	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	25		20		1
8	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		0		
9	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↓ Hold Time	1		1		2
10	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	1		1		2
11	ThDI(RFD)	Data Input to RFD ↓ Hold Time - Interlocked Handshake	0		0		
12	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay - Interlocked Handshake	0		0		
13	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay - Interlocked and 3-Wire Handshake	0		0		
14	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑) - Interlocked and 3-Wire Handshake	0		0		
15	TdACK(DAV)	$\overline{ACKIN}$ ↑ (RFD ↑)to $\overline{DAV}$ ↓ Delay - Interlocked and 3-Wire Handshake	0		0		
16	TdDAVIf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay - Input 3-Wire Handshake	0		0		
17	ThDI(DAC)	Data Input to DAC ↑ Hold Time - 3-Wire Handshake	0		0		
18	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay - Input 3-Wire Handshake	0		0		
19	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay - Input 3-Wire Handshake	0		0		
20	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay - Output 3-Wire Handshake	0		0		
21	ThDO(DAC)	Data Output to DAC ↑ Hold Time - 3-Wire Handshake	1		1		2
22	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay - Output 3-Wire Handshake	1		1		2
23	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay - Output 3-Wire Handshake	0		0		

NOTES:
1. This time can be extended through the use of the deskew timers.
2. Units equal to $\overline{AS}$ cycle.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Counter/ Timer Timing

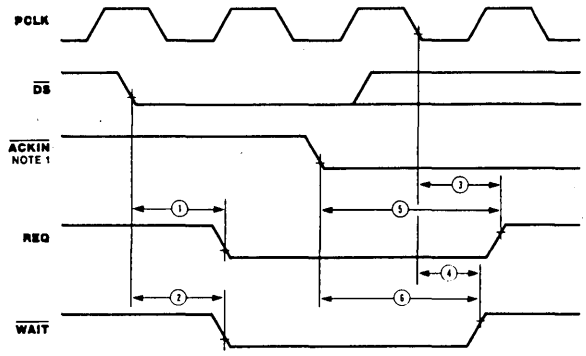


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TcPC	PCLK Cycle Time	250	4000	165	4000	1
2	TwPCh	PCLK High Width	140	2000	75	2000	
3	TwPCl	PCLK Low Width	105	2000	70	2000	
4	TfPC	PCLK Fall Time		10		10	
5	TrPC	PCLK Rise Time		10		15	
6	TcCI	Counter Input Cycle Time	500		330		
7	TCIh	Counter Input High Width	230		150		
8	TwCIl	Counter Input Low Width	230		150		
9	TfCI	Counter Input Fall Time		20		15	
10	TrCI	Counter Input Rise Time		20		15	
11	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (Timer Mode)	150		120		2
12	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (Counter Mode)	150		100		2
13	TwTI	Trigger Input Pulse Width (High or Low)	200		130		
14	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (Timer Mode)	100		100		2
15	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (Counter Mode)	100		80		2
16	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (Timer Mode)	100		70		2
17	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (Counter Mode)	100		70		2
18	TdPC(CO)	PCLK to Counter Output Delay (Timer Mode)		475		320	
19	TdCI(CO)	Counter Input to Counter Output Delay (Counter Mode)		475		420	

NOTES:
1. PCLK is only used with the counter/timers (in Timer mode), the deskew timers, and the REQUEST/WAIT logic. If these functions are not used, the PCLK input can be held low.
2. These parameters must be met to guarantee that trigger or gate are valid for the next counter/timer cycle.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

REQUEST/ WAIT Timing



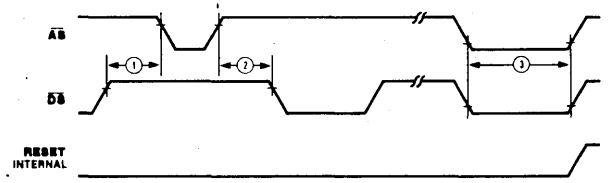
No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDS(REQ)	$\overline{DS}$ ↓ to REQ ↓ Delay		500		450	
2	TdDS(WAIT)	$\overline{DS}$ ↓ to $\overline{WAIT}$ ↓ Delay		500		450	
3	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay		300		320	
4	TdPC(WAIT)	PCLK ↓ to $\overline{WAIT}$ ↑ Delay		300		300	
5	TdACK(REQ)	ACKIN ↓ to REQ ↑ Delay		3+2 +1000		3+2 +9000	1,2
6	TdACK(WAIT)	ACKIN ↓ to $\overline{WAIT}$ ↑ Delay		10+600		10+500	3

NOTES:
1. The Delay is from DAV ↓ for the 3-Wire Input Handshake. The delay is from DAC ↑ for the 3-Wire Output Handshake.
2. Units equal to AS cycles + PCLK cycles + ns.
3. Units equal to PCLK cycles + ns.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Units in nanoseconds (ns), except as noted.

Reset Timing



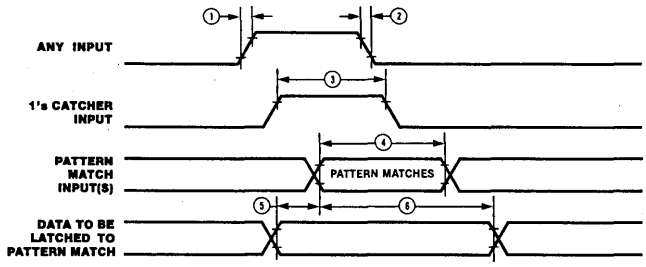
No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for No Reset	40		15		
2	TdASQ(DS)	Delay from $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for No Reset	50		30		
3	TwRES	Minimum Width of $\overline{AS}$ and $\overline{DS}$ both Low for Reset	250		170		1

NOTES:
1. Internal circuitry allows for the reset provided by the Z8 ($\overline{DS}$ held Low while $\overline{AS}$ pulses) to be sufficient.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Units in nanoseconds (ns).

Miscellaneous Port Timing

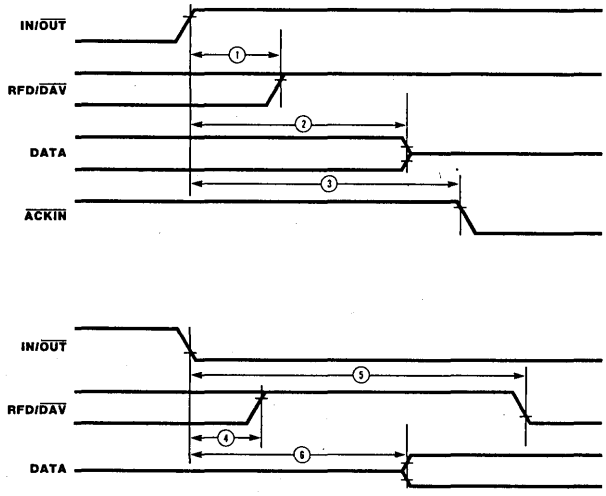


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TrI	Any Input Rise Time		100		100	
2	TfI	Any Input Fall Time		100		100	
3	Twl's	1's Catcher High Width	250		170		1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		500		
5	TsPMD	Data Latched on Pattern Match Setup Time (Bit Port)	0		0		
6	ThPMD	Data Latched on Pattern Match Hold Time (Bit Port)	1000		650		

NOTES:
1. If the input is programmed inverting, a Low-going pulse of the same width will be detected.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

Bidirectional Port Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdIOr(DAV)	I/O ↑ to RFD/DAV High Delay		500		500	
2	TdIOr(DRZ)	I/O ↑ to Data Float Delay		500		500	
3	TdIOr(ACK)	I/O ↑ to ACKIN ↓ Delay					2
4	TdIOf(RFD)	I/O ↓ to RFD/DAV High Delay		500		500	
5	TdIOf(DAV)	I/O ↓ to RFD/DAV ↓ Delay	3		3		1
6	TdDO(IO)	I/O ↓ to Data Bus Driven	2		2		1

NOTES:
1. Units equal to AS cycles.
2. Minimum delay is four AS cycles or one AS cycle after the corresponding IP is cleared, whichever is longer.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

Z8536 CIO Counter/Timer and Parallel I/O Unit

October 1988

Features

- Two independent 8-bit, double-buffered, bidirectional I/O ports plus a 4-bit special-purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/$\overline{\text{WAIT}}$ signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or nonretriggerable.
- Easy to use since all registers are read/write.

General Description

The Z8536 CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique internal address, so that any register can be accessed in two operations. All data registers can be directly accessed in a single operation. The CIO is easily interfaced to all popular microprocessors.

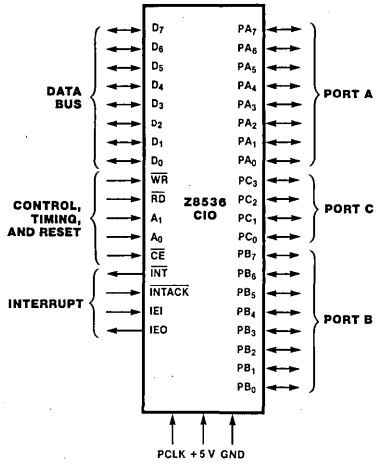


Figure 1. Pin Functions

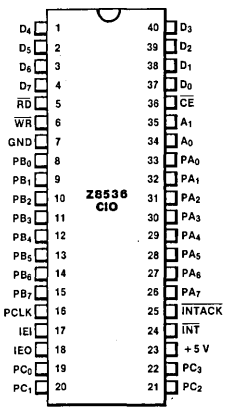


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

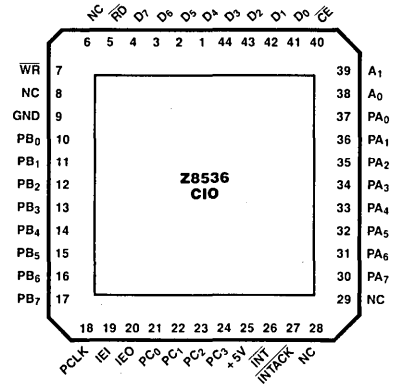


Figure 2b. 44-pin Chip Carrier, Pin Assignments

Pin Description

A_0-A_1. *Address Lines* (input). These two lines are used to select the register involved in the CPU transaction: Port A's Data register, Port B's Data register, Port C's Data register, or a control register.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level on this input enables the CIO to be read from or written to.

D_0-D_7. *Data Bus* (bidirectional 3-state). These eight data lines are used for transfers between the CPU and the CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This input indicates to the CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ must be synchronized to PCLK, and it must be stable throughout the Interrupt Acknowledge cycle.

PA_0-PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port A and external devices.

PB_0-PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0-PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the CIO's Port C.

PCLK. *Peripheral Clock* (input, TTL-compatible). This is the clock used by the internal control logic and the counter/timers in timer mode. It does not have to be the CPU clock.

$\overline{RD}$*. *Read* (input, active Low). This signal indicates that a CPU is reading from the CIO. During an Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the data bus if the CIO is the highest priority device requesting an interrupt.

$\overline{WR}$*. *Write* (input, active Low). This signal indicates a CPU write to the CIO.

*When $\overline{RD}$ and $\overline{WR}$ are detected Low at the same time (normally an illegal condition), the CIO is reset.

Architecture

The CIO Counter/Timer and Parallel I/O element (Figure 3) consists of a CPU interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose 4-bit port), three 16-bit counter/timers, an interrupt-control logic block, and the internal-control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

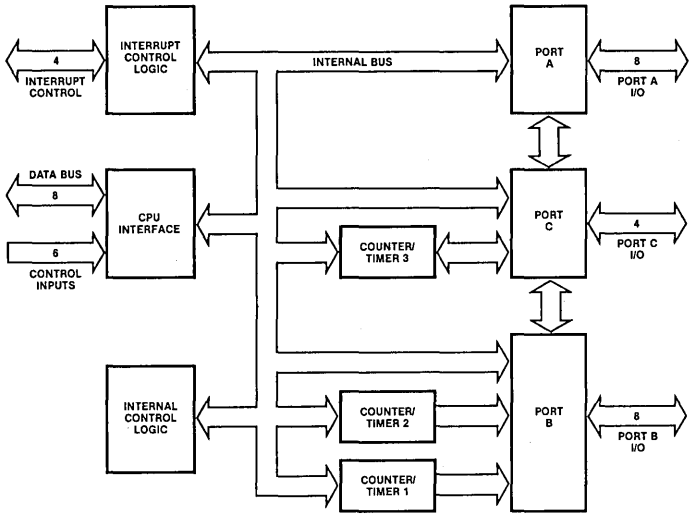


Figure 3. CIO Block Diagram

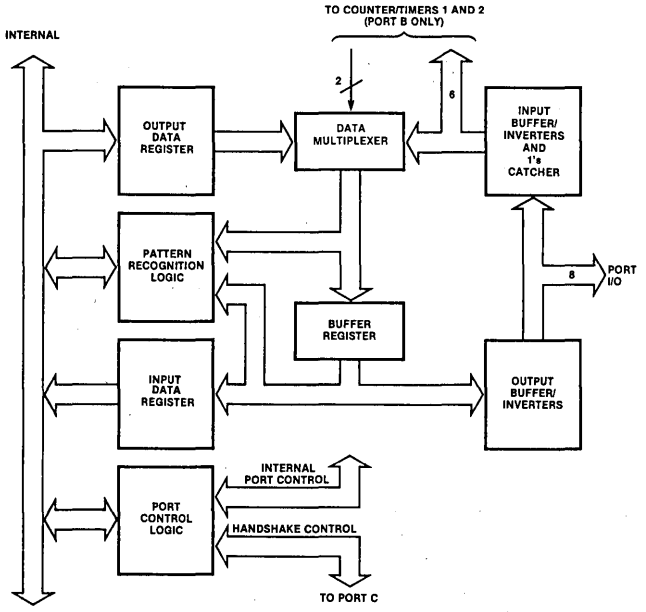


Figure 4. Ports A and B Block Diagram

The two general-purpose 8-bit I/O ports (Figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers: the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

The function of the special-purpose 4-bit port, Port C (Figure 5), depends upon the roles of Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path: the Data Path Polarity, Data Direction, and Special I/O Control registers.

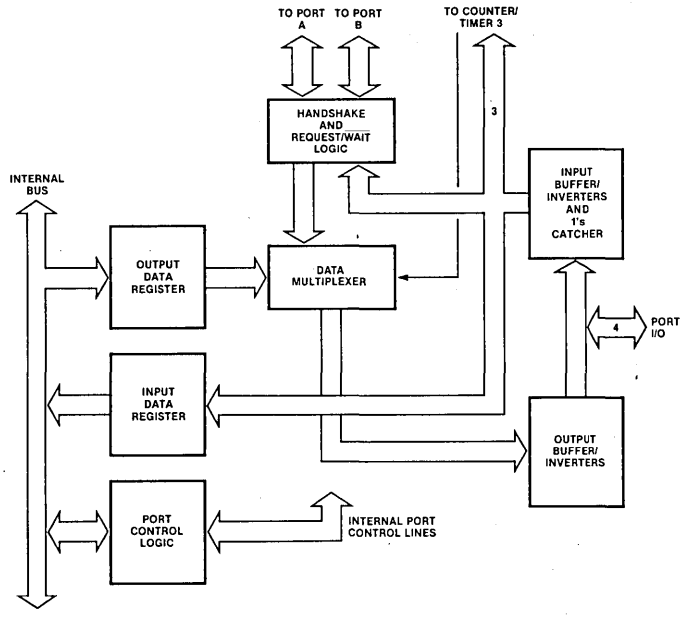


Figure 5. Port C Block Diagram

The three counter/timers (Figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer: counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or nonretriggerable. With these and other options, most counter/timer applications are covered.

There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

Architecture
(Continued)

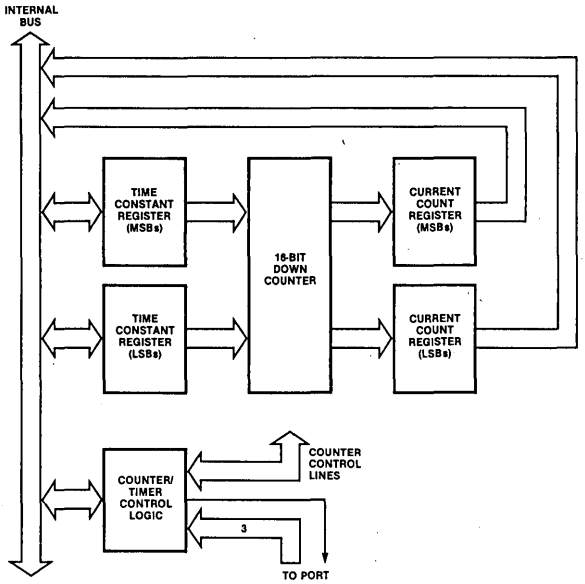


Figure 6. Counter/Timer Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/$\overline{\text{WAIT}}$ line can also be provided so that CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/$\overline{\text{WAIT}}$ can be used as input or output bits (individually data-direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Functional Description (Continued)

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The CIO provides four different handshakes for its ports: Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in Table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3.

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked, Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (Table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{DAV}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{DAV}$ is allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{DAV}$

Port A/B Configuration	PC_3	PC_2	PC_1	PC_0
Ports A and B: Bit Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O
Port B: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O	RFD or $\overline{DAV}$	$\overline{ACKIN}$
Port A or B: Input Port (3-Wire Handshake)	RFD (Output)	$\overline{DAV}$ (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	DAC (Output)
Port A or B: Output Port (3-Wire Handshake)	$\overline{DAV}$ (Output)	DAC (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	RFD (Input)
Port A or B: Bidirectional Port (Interlocked or Strobed Handshake)	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	IN/$\overline{OUT}$

*Both Ports A and B can be specified input or output with Interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{WAIT}$.

Table 1. Port C Bit Utilization

Functional Description (Continued)

goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input ($\overline{ACKIN}$) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked handshake, the signal indicating the port is ready for another data transfer operates independently of the $\overline{ACKIN}$ input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (Figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (Figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the $\overline{ACKIN}$ path. The external $\overline{ACKIN}$ input triggers the timer and its output is used as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{DAV}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{DAV}$ output goes Low and the timer output is used as the actual $\overline{DAV}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

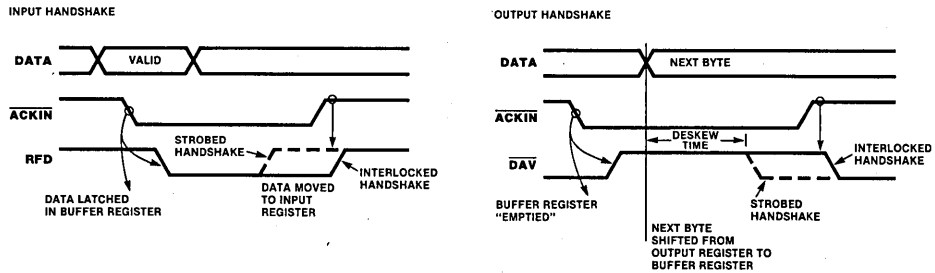


Figure 7. Interlocked and Strobed Handshakes

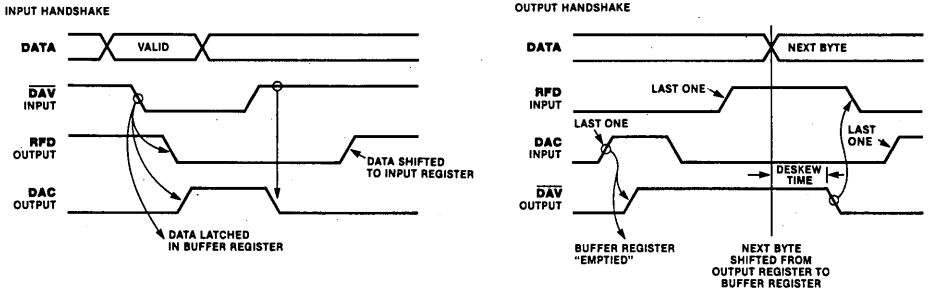


Figure 8. 3-Wire Handshake

Functional Description (Continued)

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the CPU interface. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all nonmasked bit specifications in the AND mode or the satisfaction of any nonmasked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

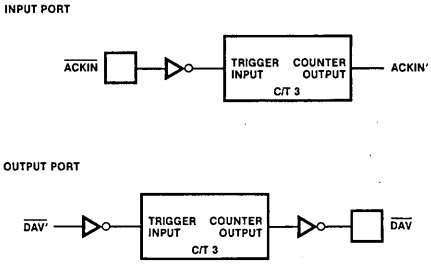


Figure 9. Pulsed Handshake

The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt. Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes: transparent and latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1 after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. Once the

Functional Description (Continued)

Acknowledge cycle is initiated, the vector is frozen until the corresponding IP is cleared. Where inputs that cause interrupts might change before the interrupt is serviced, the 1's catcher can be used to hold the value. Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode, and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern-Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register to the Input register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type application when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

Table 2. Counter/Timer External Access

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four least significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (Table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output: pulse, one-shot, and square-wave. Figure 10 shows the counter/timer waveforms. When the Pulse mode

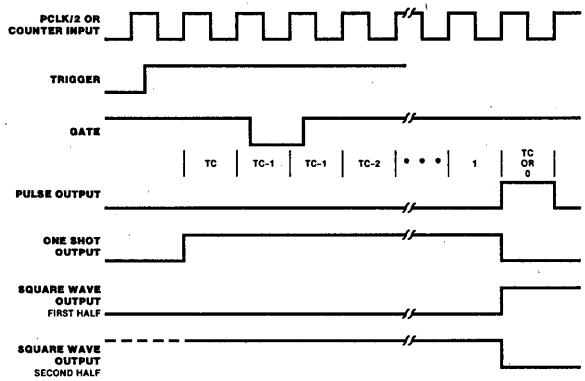


Figure 10. Counter/Timer Waveforms

Functional Description (Continued)

is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal countdown sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/Single Cycle (C/$\overline{SC}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{SC}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{SC}$ bit is 1 each time the countdown counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65,536. The down-counter is loaded in one of three ways: by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal output if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Once the down-counter is loaded, the countdown sequence continues toward terminal count as long as all the counter/timers' hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a countdown sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the countdown sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways: by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The CIO has five potential sources of interrupts: the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order: Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device, the CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the CIO contains three bits for the control and status of the interrupt logic: an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS)

Functional Description (Continued)

status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{INT}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. It can also be set directly by the CPU. Its primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service; however, the $\overline{INT}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the CIO's IEO is forced Low, independent of the state of the CIO or its IEI input, and all lower priority devices' interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The CIO contains three vector registers: one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1. The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

Programming

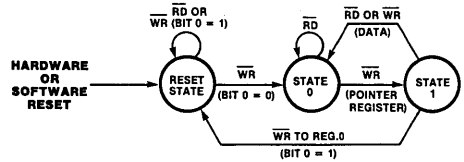
The data registers within the CIO are directly accessed by address lines A_0 and A_1 (Table 3). All other internal registers are accessed by the following two-step sequence, with the address lines specifying a control operation. First, write the address of the target register to an internal 6-bit Pointer Register; then read from or write to the target register. The Data registers can also be accessed by this method.

An internal state machine determines if accesses with A_0 and A_1 equalling 1 are to the Pointer Register or to an internal control register (Figure 11). Following any control read operation, the state machine is in State 0 (the next control access is to the Pointer Register). This can be used to force the state machine into a known state. Control reads in State 0 return the contents of the last register pointed to. Therefore, a register can be read continuously without writing to the Pointer. While the CIO is in State 1 (next control access is to the register pointed to), many internal operations are suspended—no IPs are set and internal status is frozen. Therefore, to minimize interrupt latency and to allow continuous status updates, the CIO should not be left in State 1.

The CIO is reset by forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously (normally an illegal condition) or by writing a 1 to the Reset bit. Reset disables all functions except a read from or write to the Reset bit; writes to all other bits are ignored, and all reads return 01_H. In this state, all control bits are forced to 0 and may be programmed only after clearing the Reset bit (by writing a 0 to it).

A_1	A_0	Register
0	0	Port C's Data Register
0	1	Port B's Data Register
1	0	Port A's Data Register
1	1	Control Registers

Table 3. Register Selection



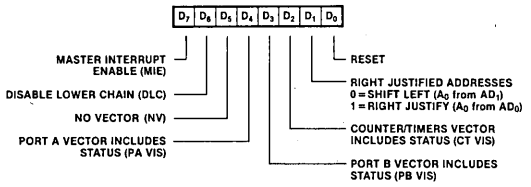
NOTE: State changes occur only when $A_0 = A_1 = 1$. No other accesses have effect.

Figure 11. State Machine Operation

Registers

Master Interrupt Control Register

Address: 000000
(Read/Write)



Master Configuration Control Register

Address: 000001
(Read/Write)

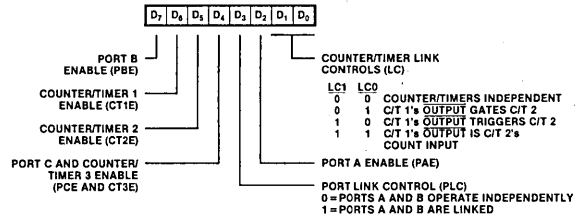
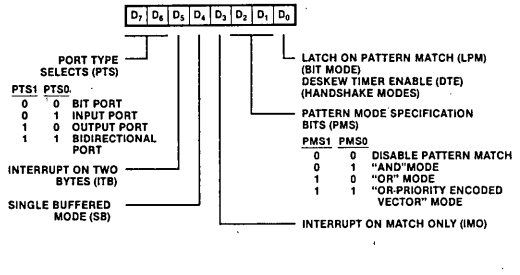


Figure 12. Master Control Registers

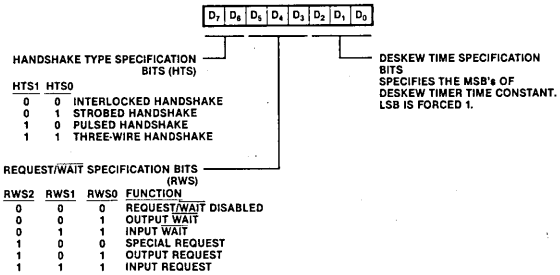
Port Mode Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)



Port Handshake Specification Registers

Addresses: 100001 Port A
101001 Port B
(Read/Write)



Port Command and Status Registers

Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)

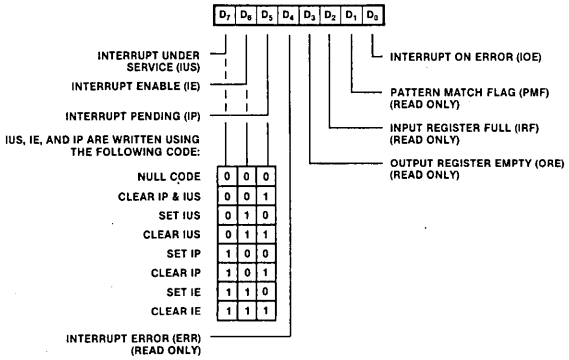
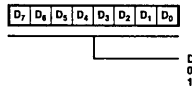


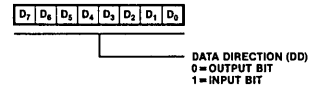
Figure 13. Port Specifications Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

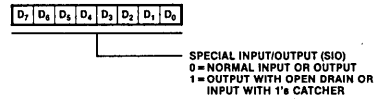
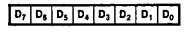


Figure 14. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A*
001110 Port B*
(Read/Write)



*These registers can be addressed directly.

Port C Data Register
Address: 001111*
(Read/Write)

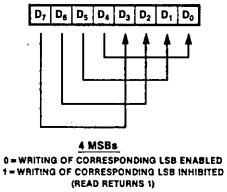
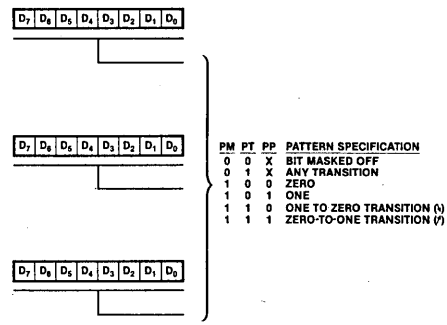


Figure 15. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)



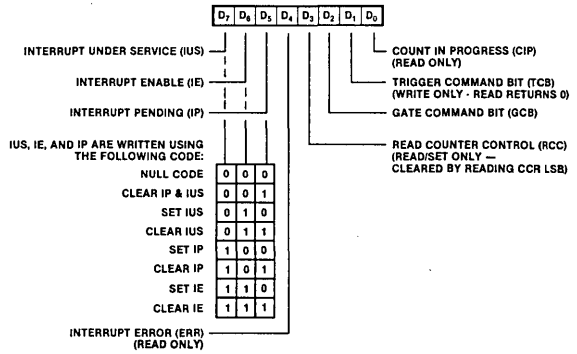
PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE TO ZERO TRANSITION (\)
1	1	1	ZERO-TO-ONE TRANSITION (/)

Figure 16. Pattern Definition Registers

Registers
(Continued)

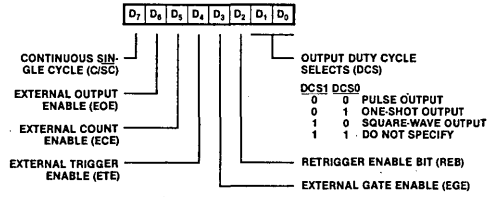
Counter/Timer Command and Status Registers

Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)



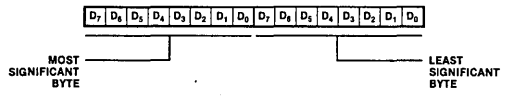
Counter/Timer Mode Specification Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)



Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

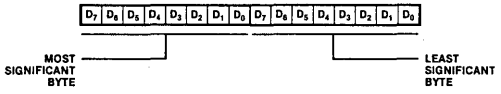
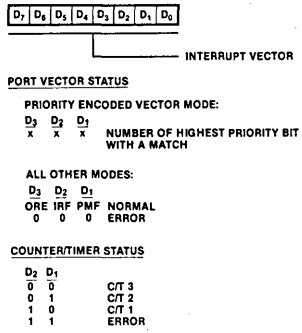


Figure 17. Counter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



Current Vector Register
Address: 011111
(Read only)

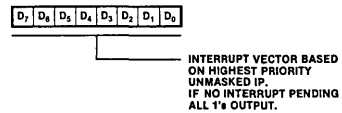


Figure 18. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address	Register Name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Most Often Accessed Registers

Address	Register Name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data (can be accessed directly)
001110	Port B's Data (can be accessed directly)
001111	Port C's Data (can be accessed directly)

Counter/Timer Related Registers

Address	Register Name
010000	Counter/Timer 1's Current Count-MSBs
010001	Counter/Timer 1's Current Count-LSBs
010010	Counter/Timer 2's Current Count-MSBs
010011	Counter/Timer 2's Current Count-LSBs
010100	Counter/Timer 3's Current Count-MSBs
010101	Counter/Timer 3's Current Count-LSBs
010110	Counter/Timer 1's Time Constant-MSBs
010111	Counter/Timer 1's Time Constant-LSBs
011000	Counter/Timer 2's Time Constant-MSBs
011001	Counter/Timer 2's Time Constant-LSBs
011010	Counter/Timer 3's Time Constant-MSBs
011011	Counter/Timer 3's Time Constant-LSBs
011100	Counter/Timer 1's Mode Specification
011101	Counter/Timer 2's Mode Specification
011110	Counter/Timer 3's Mode Specification
011111	Current Vector

Port A Specification Registers

Address	Register Name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Port B Specification Registers

Address	Register Name
101000	Port B's Mode Specification
101001	Port B's Handshake Specification
101010	Port B's Data Path Polarity
101011	Port B's Data Direction
101100	Port B's Special I/O Control
101101	Port B's Pattern Polarity
101110	Port B's Pattern Transition
101111	Port B's Pattern Mask

Timing

Read Cycle. At the beginning of a read cycle, the CPU places an address on the address bus. Bits A_0 and A_1 specify a CIO register; the remaining address bits and status information are combined and decoded to generate a Chip Enable ($\overline{CE}$) signal that selects the CIO. When Read ($\overline{RD}$) goes Low, data from the specified register is gated onto the data bus.

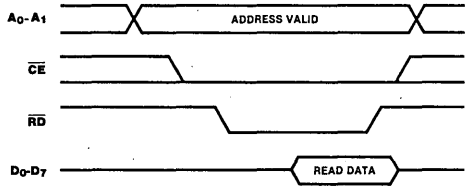


Figure 19. Read Cycle Timing

Write Cycle. At the beginning of a write cycle, the CPU places an address on the data bus. Bits A_0 and A_1 specify a CIO register; the remaining address bits and status information are combined and decoded to generate a Chip Enable ($\overline{CE}$) signal that selects the CIO. When $\overline{WR}$ goes Low, data placed on the bus by the CPU is strobed into the specified CIO register.

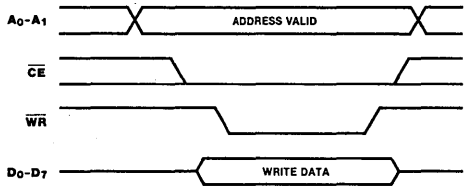


Figure 20. Write Cycle Timing

Interrupt Acknowledge. The CIO pulls its Interrupt Request ($\overline{INT}$) line Low, requesting interrupt service from the CPU, if an Interrupt Pending (IP) bit is set and interrupts are enabled. The CPU responds with an Interrupt Acknowledge cycle. When Interrupt Acknowledge ($\overline{INTACK}$) goes true and the IP is set, the CIO forces Interrupt Enable Out (IEO) Low, disabling all lower priority devices in the interrupt daisy chain. If the CIO is the highest priority device requesting service (IEI is High), it places its interrupt vector on the data bus and sets the Interrupt Under Service (IUS) bit when Read ($\overline{RD}$) goes Low.

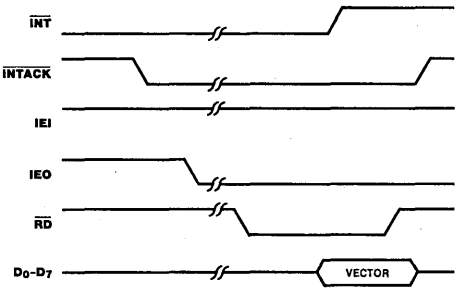
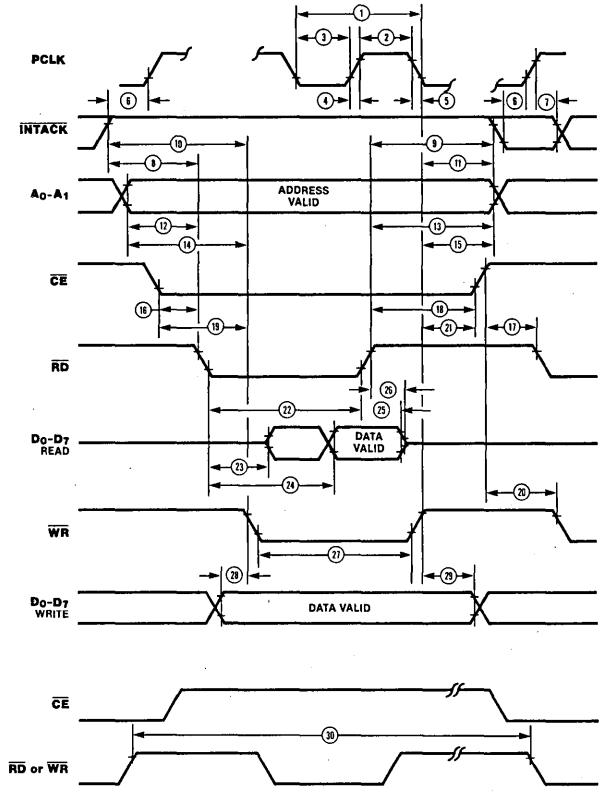
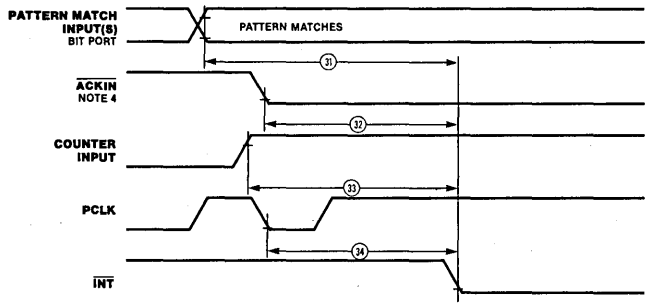


Figure 21. Interrupt Acknowledge Timing

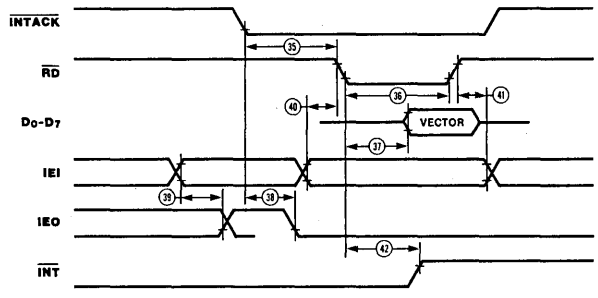
CPU Interface Timing



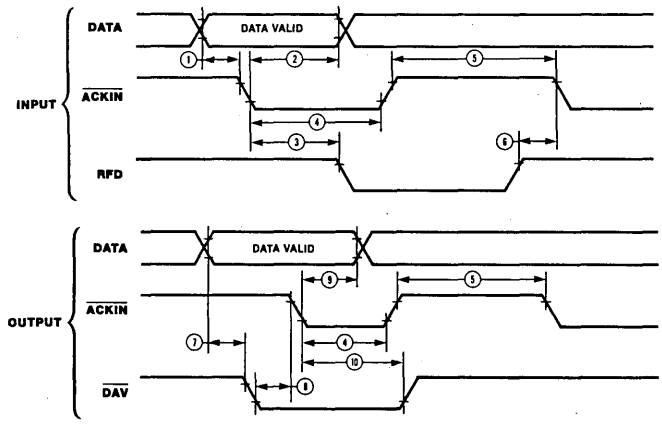
Interrupt Timing



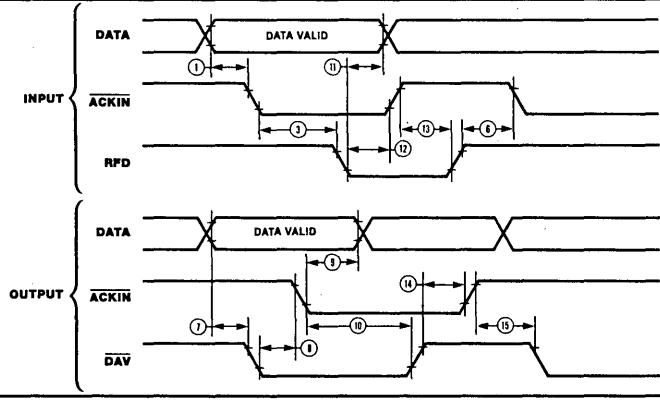
Interrupt Acknowledge Timing



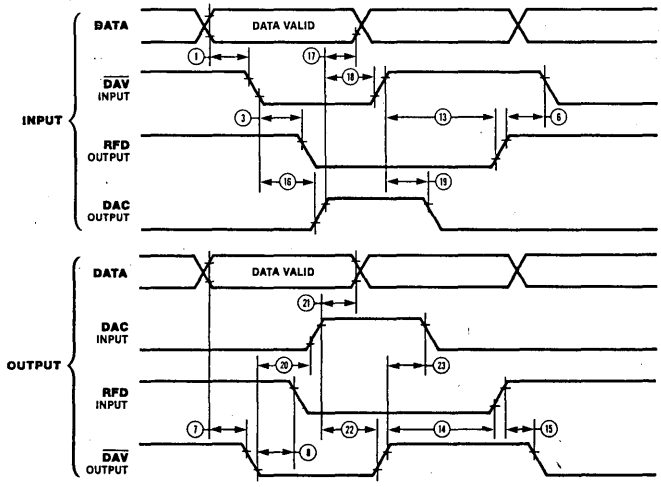
Strobed Handshake



Interlocked Handshake



3-Wire Handshake



Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The DC characteristics and capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

The Ordering Information section lists temperature ranges and product numbers. Package drawings are in the Package Information section in this book. Refer to the Literature List for additional documentation.

All ac parameters assume a load capacitance of 50 pf max.

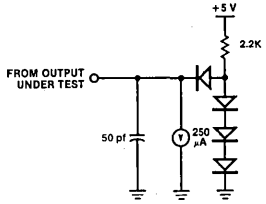


Figure 22. Standard Test Load

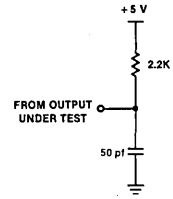


Figure 23. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
			0.5	V	$I_{OL} = +3.2$ mA
I_{IL}	Input Leakage		±10.0	μA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10.0	μA	$0.4 \leq V_{OUT} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		200	mA	

V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pf	
C_{OUT}	Output Capacitance		15	pf	
$C_{I/O}$	Bidirectional Capacitance		20	pf	

f = 1 MHz, over specified temperature range.
Unmeasured pins returned to ground.

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TcPC	PCLK Cycle time	250	4000	165	4000	
2	TwPCh	PCLK Width (High)	105	2000	70	2000	
3	TwPCl	PCLK Width (Low)	105	2000	70	2000	
4	TrPC	PCLK Rise Time		20		10	
5	TfPC	PCLK Fall Time		20		15	
6	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	100		100		
7	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	0		0		
8	TsIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		200		
9	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		
10	TsIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		200		
11	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		
12	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		80		
13	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		
14	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		80		
15	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		
16	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		1
17	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		70		1
18	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		1
19	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		
20	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		70		
21	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		
22	TwRDl	$\overline{RD}$ Low Width	390		250		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
24	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		255		180	
25	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70		45	2
27	TwWRl	$\overline{WR}$ Low Width	390		250		
28	TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time	0		0		
29	ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time	0		0		
30	Trc	Valid Access Recovery Time	1000*		650		3
31	TdPM(INT)	Pattern Match to $\overline{INT}$ Delay (Bit Port)		2+800		**2+800**	6
32	TdACK(INT)	ACKIN to $\overline{INT}$ Delay (Port with Handshake)		10+600		**10+600**	4,6
33	TdCI(INT)	Counter Input to $\overline{INT}$ Delay (Counter Mode)		2+700		**2+700**	6
34	TdPC(INT)	PCLK to $\overline{INT}$ Delay (Timer Mode)		3+700		**3+700**	6
35	TsIA(RDA)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Setup Time	350		250		5
36	TwRDA	$\overline{RD}$ (Acknowledge) Width	350		250		
37	TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay		250		180	
38	TdIA(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		350		250	5
39	TdIEI(IEO)	IEI to IEO Delay		150		100	5
40	TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time	100		70		5
41	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time	100		70		
42	TdRDA(INT)	$\overline{RD}$ ↓ (Acknowledge) to $\overline{INT}$ ↑ Delay		600		600	

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V with minimum ac load and maximum dc load.
3. Trc is the specified number or 3 TcPC, whichever is longer.
4. The delay is from DAV ↓ for 3 Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.
5. The parameters for the devices in any particular daisy chain must meet the following constraint: The delay from INTACK ↓ to RD ↓ must be greater than the sum of TdIA(IEO) for the highest priority peripheral, TsIEI(RDA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.
6. Units are equal to TcPC plus ns.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

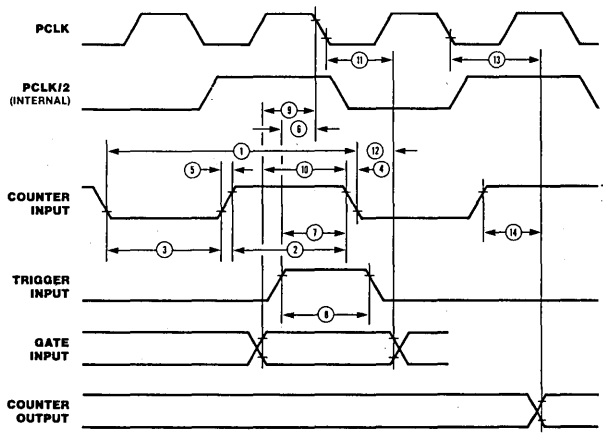
† Units in nanoseconds (ns), except as noted.

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Setup Time	0		0		
2	ThDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Hold Time—Strobed Handshake	500		330		
3	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		0		
4	TwACK1	$\overline{ACKIN}$ Low Width—Strobed Handshake	250		165		
5	TwACKh	$\overline{ACKIN}$ High Width—Strobed Handshake	250		165		
6	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		0		
7	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	25		20		1
8	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		0		
9	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↓ Hold Time	2		2		2
10	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	2		2		2
11	THDI(RFD)	Data Input to RFD ↓ Hold Time—Interlocked Handshake					
12	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay Interlocked Handshake	0		0		
13	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handwshake	0		0		
14	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑)—Interlocked and 3-Wire Handshake	0		0		
15	TdACK(DAV)	$\overline{ACKIN}$ ↑ (RFD ↑) to $\overline{DAV}$ ↓ Delay—Interlocked and 3-Wire Handshake	0		0		
16	TdDAVIf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay—Input 3-Wire Handshake	0		0		
17	ThDI(DAC)	Data Input to DAC ↑ Hold Time—3-Wire Handshake	0		0		
18	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay—Input 3-Wire Handshake	0		0		
19	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay—Input 3-Wire Handshake	0		0		
20	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay—Output 3-Wire Handshake	0		0		
21	ThDO(DAC)	Data Output to DAC ↑ Hold Time—3-Wire Handshake	2		2		2
22	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay—Output 3-Wire Handshake	2		2		2
23	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay—Output 3-Wire Handshake	0		0		

NOTES:
1. This time can be extended through the use of deskew timers.
2. Units equal to TcPC.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Counter/ Timer Timing

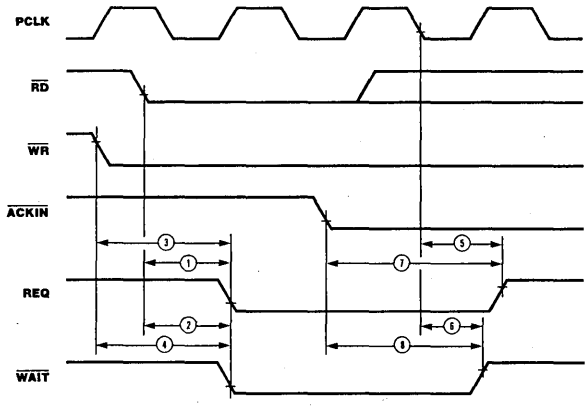


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TcCI	Counter Input Cycle Time	500		330		
2	TCIh	Counter Input High Width	230		150		
3	TWCIl	Counter Input Low Width	230		150		
4	TfCI	Counter Input Fall Time		20		15	
5	TrCI	Counter Input Rise Time		20		15	
6	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (Timer Mode)	150		120		1
7	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (Counter Mode)	150		100		1
8	TwTI	Trigger Input Pulse Width (High or Low)	200		130		
9	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (Timer Mode)	100		100		1
10	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (Counter Mode)	100		80		1
11	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (Timer Mode)	100		70		1
12	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (Counter Mode)	100		70		1
13	TdPC(CO)	PCLK to Counter Output Delay (Timer Mode)		475		320	
14	TdCI(CO)	Counter Input to Counter Output Delay (Counter Mode)		475		420	

NOTES:
1. These parameters must be met to guarantee trigger or gate are valid for the next counter/timer cycle.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

REQUEST/ WAIT Timing

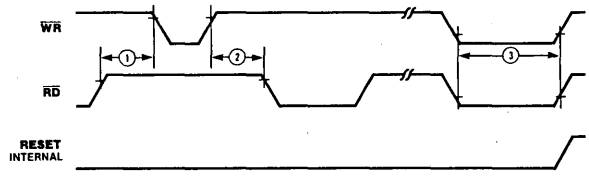


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRD(REQ)	$\overline{RD}$ ↓ to REQ ↓ Delay		500		350	
2	TdRD(WAIT)	$\overline{RD}$ ↓ to $\overline{WAIT}$ ↓ Delay		500		350	
3	TdWR(REQ)	$\overline{WR}$ ↓ to REQ ↓ Delay		500		400	
4	TdWR(WAIT)	$\overline{WR}$ ↓ to $\overline{WAIT}$ ↓ Delay		500		400	
5	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay		300		300	
6	TdPC(WAIT)	PCLK ↓ to $\overline{WAIT}$ ↑ Delay		300		300	
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to REQ ↑ Delay		8+1000		8 + 900	1,2
8	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay		10+500		10 + 500	1,2

NOTES:
1. The delay is fronm DAV ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.
2. Units equal to TcPC + ns.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Reset Timing

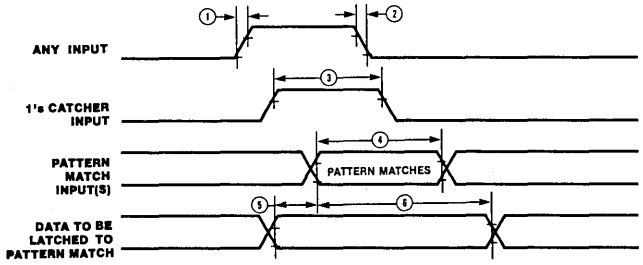


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓ for No Reset	50		50		
2	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓ for No Reset	50		50		
3	TwRES	Minimum Width of $\overline{RD}$ and $\overline{WR}$ both Low for Reset	250		250		

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Units in nanoseconds (ns).

Miscellaneous Port Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TrI	Any Input Rise Time		100		100	
2	TfI	Any Input Fall Time		100		100	
3	Tw1's	1's Catcher High Width	250		170		1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		500		
5	TsPMD	Data Latched on Pattern Match Setup Time (Bit Port)	0		0		
6	ThPMD	Data Latched on Pattern Match Hold Time (Bit Port)	1000		650		

NOTES:
1. If the input is programmed inverting, a Low-going pulse of the same width will be detected.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logiq "0".
† Units in nanoseconds (ns).

Z8038/Z8538 FIO FIFO Input/ Output Interface Unit

October 1988

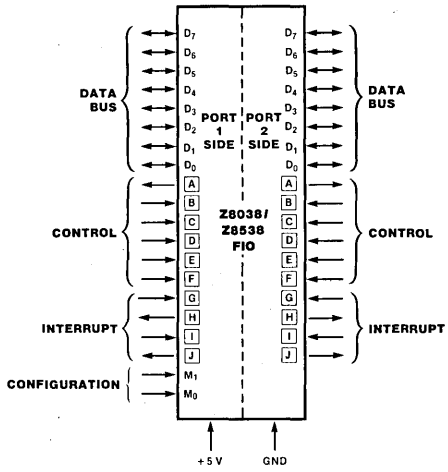
Features

- 128-byte FIFO buffer provides asynchronous bidirectional CPU/CPU or CPU/peripheral interface, expandable to any width in byte increments by use of multiple FIOs.
- Interlocked 2-Wire or 3-Wire Handshake logic port mode; Z-BUS® or non-Z-BUS interface.
- Pattern-recognition logic stops DMA transfers and/or interrupts CPU; preset byte count can initiate variable-length DMA transfers.
- Seven sources of vectored/nonvectored interrupt which include pattern-match, byte count, empty or full buffer status; a dedicated "mailbox" register with interrupt capability provides CPU/CPU communication.
- $\overline{\text{REQUEST}}/\overline{\text{WAIT}}$ lines control high-speed data transfers.
- All functions are software controlled via directly addressable read/write registers.

General Description

The Z8038/Z8538 FIO provides an asynchronous 128-byte FIFO buffer between two CPUs or between a CPU and a peripheral device. This buffer interface expands to a 16-bit or wider data path and expands in depth to add as many Z8060 FIFOs (and an additional FIO) as are needed.

The FIO manages data transfers by assuming Z-BUS, non-Z-BUS microprocessor (a generalized microprocessor interface), Interlocked 2-Wire Handshake, and 3-Wire Handshake operating modes. These modes interface dissimilar CPUs or CPUs and peripherals running under differing speeds or protocols, allowing asynchronous data transactions and improving I/O overhead by as much as two orders of magnitude. Figures 1 and 2 show how the signals controlling these operating modes are mapped to the FIO pins.



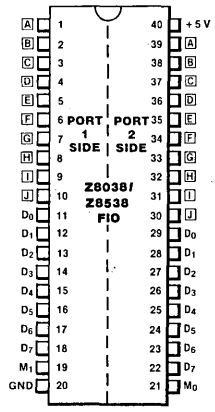


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

General Description (Continued)

The FIO supports the Z-BUS interrupt protocols, generating seven sources of interrupts upon any of the following events: a write to a message register, change in data direction, pattern match, status match, over/underflow error, buffer full and buffer empty status. Each interrupt source can be enabled or disabled, and can also place an interrupt vector on the port address/data lines.

The data transfer logic of the FIO has been specially designed to work with DMA (Direct Memory Access) devices for high-speed transfers. It provides for data transfers to or from memory each machine cycle, while the DMA device generates memory address and control signals. The FIO also supports the variably sized block length, improving system throughput when multiple variable length messages are transferred amongst several sources.

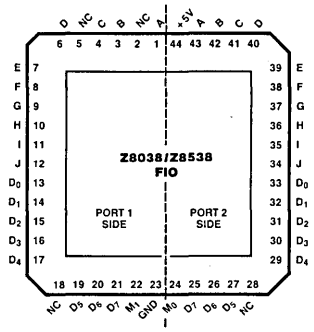


Figure 2b. 44-pin Chip Carrier, Pin Assignments

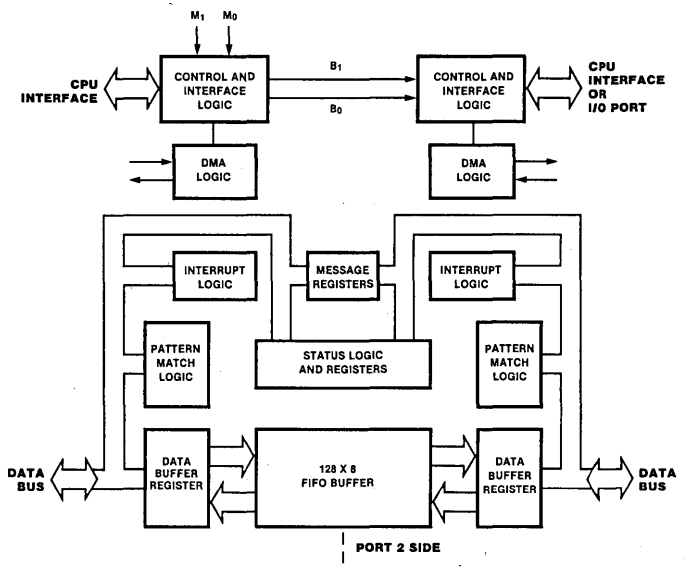


Figure 3. FIO Block Diagram

Functional Description

Operating Modes. Ports 1 and 2 operate in any of twelve combinations of operating modes, listed in Table 2. Port 1 functions in either the Z-BUS or non-Z-BUS microprocessor modes, while Port 2 functions in Z-BUS, non-Z-BUS, Interlocked 2-Wire Handshake, and 3-Wire Handshake modes. Table 1 describes the signals and their corresponding pins in each of these modes.

The pin diagrams of the FIO are identical, except for two pins on the Port 1 side, which select that port's operating mode. Port 2's operating mode is programmed by two bits in Port 1's Control register 0. Table 2 describes the combinations of operating modes; Table 3 describes the control signals mapped to pins A–J in the five possible operating modes.

Signal Pins	Z-BUS Low Byte	Z-BUS High Byte	Non-Z-BUS	Interlocked HS Port*	3-Wire HS Port*
A	$\overline{REQ}/\overline{WT}$	$\overline{REQ}/\overline{WT}$	$\overline{REQ}/\overline{WT}$	RFD/$\overline{DAV}$	RFD/$\overline{DAV}$
B	$\overline{DMASTB}$	$\overline{DMASTB}$	$\overline{DACK}$	$\overline{ACKIN}$	$\overline{DAV}$/DAC
C	$\overline{DS}$	$\overline{DS}$	$\overline{RD}$	FULL	DAC/RFD
D	R/$\overline{W}$	R/$\overline{W}$	$\overline{WR}$	EMPTY	EMPTY
E	$\overline{CS}$	$\overline{CS}$	$\overline{CE}$	$\overline{CLEAR}$	$\overline{CLEAR}$
F	$\overline{AS}$	AS	C/$\overline{D}$	DATA DIR	DATA DIR
G	$\overline{INTACK}$	A_0	$\overline{INTACK}$	IN_0	IN_0
H	IEO	A_1	IEO	OUT_1	OUT_1
I	IEI	A_2	IEI	$\overline{OE}$	$\overline{OE}$
J	$\overline{INT}$	A_3	$\overline{INT}$	OUT_3	OUT_3

*2 side only.

Table 1. Pin Assignments

Mode	M_1	M_0	B_1	B_0	Port 1	Port 2
0	0	0	0	0	Z-BUS Low Byte	Z-BUS Low Byte
1	0	0	0	1	Z-BUS Low Byte	Non-Z-BUS
2	0	0	1	0	Z-BUS Low Byte	3-Wire Handshake
3	0	0	1	1	Z-BUS Low Byte	2-Wire Handshake
4	0	1	0	0	Z-BUS High Byte	Z-BUS High Byte
5	0	1	0	1	Z-BUS High Byte	Non-Z-BUS
6	0	1	1	0	Z-BUS High Byte	3-Wire Handshake
7	0	1	1	1	Z-BUS High Byte	2-Wire Handshake
8	1	0	0	0	Non-Z-BUS	Z-BUS Low Byte
9	1	0	0	1	Non-Z-BUS	Non-Z-BUS
10	1	0	1	0	Non-Z-BUS	3-Wire Handshake
11	1	0	1	1	Non-Z-BUS	2-Wire Handshake

Table 2. Operating Modes

Functional Description (Continued)

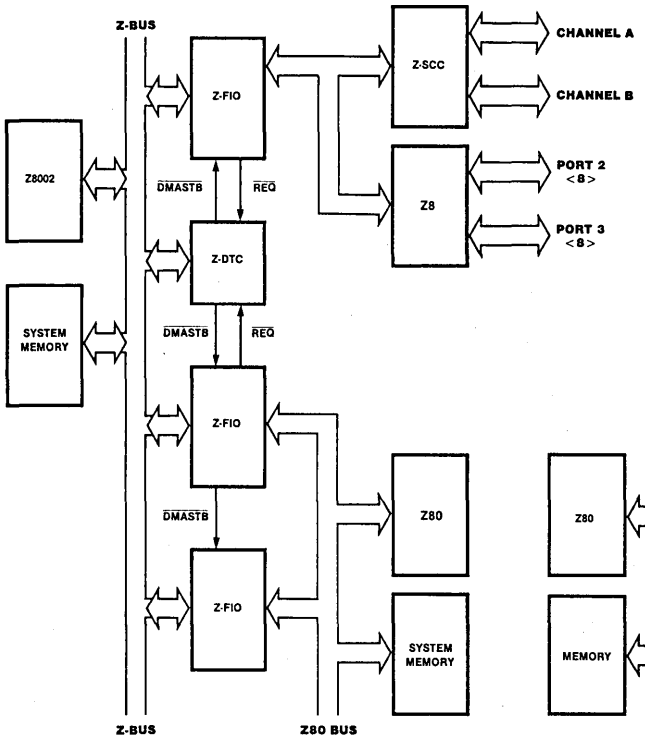


Figure 4. CPU to CPU Configuration

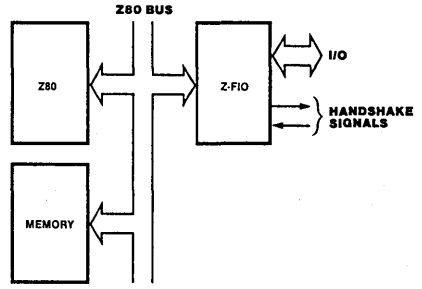


Figure 5. CPU to I/O Configuration

Pins Common To Both Sides

Pin Signals	Pin Names	Pin Numbers	Signal Description
M_0	M_0	21	M_1 and M_0 program Port 1 side CPU interface
M_1	M_1	19	
+5 Vdc	+5 Vdc	40	DC power source
GND	GND	20	DC power ground

Z-BUS Low Byte Mode

Z8038

Pin Signals	Pin Names	Pin Numbers Port 1	Pin Numbers Port 2	Signal Description
AD_0-AD_7 (Address/Data)	D_0-D_7	11-18	29-22	Multiplexed bidirectional address/data lines, Z-BUS compatible.
$\overline{REQ}/\overline{WAIT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; WAIT line (open-drain) output for synchronized CPU and FIO data transfers.
$\overline{DMASTB}$ (Direct Memory Access Strobe)	B	2	38	Input, active Low. Strobes DMA data to and from the FIFO buffer.
$\overline{DS}$ (Data Strobe)	C	3	37	Input, active Low. Provides timing for data transfer to or from FIO.
$R/\overline{W}$ (Read/Write)	D	4	36	Input; active High signals CPU read from FIO; active Low signals CPU write to FIO.
$\overline{CS}$ (Chip Select)	E	5	35	Input, active Low. Enables FIO. Latched on the rising edge of $\overline{AS}$.
$\overline{AS}$ (Address Strobe)	F	6	34	Input, active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ sampled while $\overline{AS}$ Low.
$\overline{INTACK}$ (Interrupt Acknowledge)	G	7	33	Input, active Low. Acknowledges an interrupt. Latched on the rising edge of $\overline{AS}$.
IEO (Interrupt Enable Out)	H	8	32	Output, active High. Sends interrupt enable to lower priority device IEI pin.
IEI (Interrupt Enable In)	I	9	31	Input, active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (Interrupt)	J	10	30	Output, open drain, active Low. Signals FIO interrupt request to CPU.

Z-BUS High Byte Mode

Z8038

Pin Signals	Pin Names	Pin Numbers Port 1	Pin Numbers Port 2	Signal Description
AD_0-AD_7 (Address/Data)	D_0-D_7	11-18	29-22	Multiplexed bidirectional address/data lines, Z-BUS compatible.
$\overline{REQ}/\overline{WAIT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; WAIT line (open-drain) output for synchronized CPU and FIO data transfers.
$\overline{DMASTB}$ (Direct Memory Access Strobe)	B	2	38	Input, active Low. Strobes DMA data to and from the FIFO buffer.
$\overline{DS}$ (Data Strobe)	C	3	37	Input, active Low. Provides timing for transfer of data to or from FIO.
$R/\overline{W}$ (Read/Write)	D	4	36	Input, active High. Signals CPU read from FIO; active Low signals CPU write to FIO.
$\overline{CS}$ (Chip Select)	E	5	35	Input, active Low. Enables FIO. Latched on the rising edge of $\overline{AS}$.
$\overline{AS}$ (Address Strobe)	F	6	34	Input, active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ are sampled while $\overline{AS}$ is Low.
A_0 (Address Bit 0)	G	7	33	Input, active High. With A_1, A_2, and A_3, addresses FIO internal registers.
A_1 (Address Bit 1)	H	8	32	Input, active High. With A_0, A_2, and A_3, addresses FIO internal registers.
A_2 (Address Bit 2)	I	9	31	Input, active High. With A_0, A_1, and A_3, addresses FIO internal registers.
A_3 (Address Bit 3)	J	10	30	Input, active High. With A_0, A_1, and A_2, addresses FIO internal registers.

Table 3. Signal/Pin Descriptions

Non-Z-BUS Mode

Z8538

Pin Signals	Pin Names	Pin Numbers Port 1	Port 4	Signal Description
D_0-D_7 (Data)	D_0-D_7	11-18	29-22	Bidirectional data bus.
$\overline{REQ}/\overline{WT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfer.
$\overline{DACK}$ (DMA Acknowledge)	B	2	38	Input, active Low. DMA acknowledge.
$\overline{RD}$ (Read)	C	3	37	Input, active Low. Signals CPU read from FIO.
$\overline{WR}$ (Write)	D	4	36	Input, active Low. Signals CPU write to FIO.
$\overline{CE}$ (Chip Select)	E	5	35	Input, active Low. Used to select FIO.
C/$\overline{D}$ (Control/Data)	F	6	34	Input, active High. Identifies control byte on D_0-D_7; active Low identifies data byte on D_0-D_7.
$\overline{INTACK}$ (Interrupt Acknowledge)	G	7	33	Input, active Low. Acknowledges an interrupt.
IEO (Interrupt Enable Out)	H	8	32	Output, active High. Sends interrupt enable to lower priority device IEI pin.
IEI (Interrupt Enable In)	I	9	31	Input, active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (Interrupt)	J	10	30	Output, open drain, active Low. Signals FIO interrupt to CPU.

Port 2—I/O Port Mode

Pin Signals	Pin Names	Pin Numbers	Mode	Signal Description
D_0-D_7 (Data)	D_0-D_7	29-22	2-Wire HS* 3-Wire HS	Bidirectional data bus.
RFD/$\overline{DAV}$ (Ready for Data/Data Available)	A	39	2-Wire HS 3-Wire HS	Output, RFD active High. Signals peripherals that FIO is ready to receive data. $\overline{DAV}$ active Low signals that FIO is ready to send data to peripherals.
$\overline{ACKIN}$ (Acknowledge Input)	B	38	2-Wire HS	Input, active Low. Signals FIO that output data is received by peripherals or that input data is valid.
$\overline{DAV}$/DAC (Data Available/Data Accepted)	B	38	3-Wire HS	Input; $\overline{DAV}$ (active Low) signals that data is valid on bus. DAC (active High) signals that output data is accepted by peripherals.
FULL	C	37	2-Wire HS	Output, open drain, active High. Signals that FIO buffer is full.
DAC/RFD (Data Accepted/Ready for Data)	C	37	3-Wire HS	Direction controlled by internal programming. Both active High. DAC (an output) signals that FIO has received data from peripheral; RFD (an input) signals that the listeners are ready for data.
EMPTY	D	36	2-Wire HS 3-Wire HS	Output, open drain, active High. Signals that FIFO buffer is empty.
$\overline{CLEAR}$	E	35	2-Wire HS 3-Wire HS	Programmable input or output, active Low. Clears all data from FIFO buffer.
DATA DIR (Data Direction)	F	34	2-Wire HS 3-Wire HS	Programmable input or output. Active High signals data input to Port 2; Low signals data output from Port 2.
IN_0	G	33	2-Wire HS 3-Wire HS	Input line to D_0 of Control Register 3.
OUT_1	H	32	2-Wire HS 3-Wire HS	Output line from D_1 of Control Register 3.
$\overline{OE}$ (Output Enable)	I	31	2-Wire HS 3-Wire HS	Input, active Low. When Low, enables bus drivers. When High, floats bus drivers at high impedance.
OUT_3	J	30	2-Wire HS 3-Wire HS	Output line from D_3 of Control register 3.

*Handshake

Table 3. Signal/Pin Descriptions (Continued)

Reset

The FIO can be reset under either hardware or software control by one of the following methods:

- By forcing both $\overline{AS}$ and $\overline{DS}$ Low simultaneously in Z-BUS mode (normally illegal).
- By forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously in non-Z-BUS mode.
- By writing a 1 to the Reset bit in Control register 0 for software reset.

In the Reset state, all control bits are cleared to 0. Only after clearing the Reset bit (by writing a 0 to it) can the other command bits be programmed. This action is true for both sides of the FIO when programmed as a CPU interface.

For proper system control, when Port 1 is reset, Port 2 is also reset. In addition, all Port 2's outputs are floating and all inputs are ignored. To initiate the data transfer, Port 2 must be enabled by Port 1. The Port 2 CPU can determine when it is enabled by reading Control register 0, which reads "floating" data bus if not enabled and "01_H" if enabled.

CPU Interfaces

The FIO is designed to work with both Z-BUS- and non-Z-BUS-type CPUs on both Port 1 and Port 2. The Z-BUS configuration interfaces CPUs with time-multiplexed address and data information on the same pins. The Z8001®, Z8002®, and Z8® are examples of this type of CPU. The $\overline{AS}$ (Address Strobe) pin is used to latch the address and chip select information sent out by the CPU. The R/$\overline{W}$ (Read/Write) pin and the $\overline{DS}$ (Data Strobe) pin are used for timing reads and writes from the CPU to the FIO (Figures 6 and 7).

The non-Z-BUS configuration is used for CPUs where the address and data buses are separate. Examples of this type of CPU are the Z80® and the Intel 8080. The $\overline{RD}$ (Read) and $\overline{WR}$ (Write) pins are used to time reads and writes from the CPU to the FIO (Figures 9 and 10). The C/$\overline{D}$ (Control/Data) pin is used to directly access the FIFO buffer (C/$\overline{D}$ = 0) and to access the other

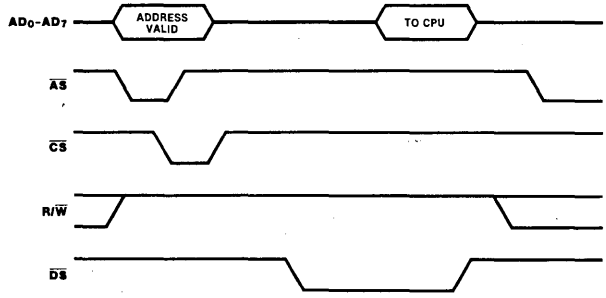


Figure 6. Z-BUS Read Cycle Timing

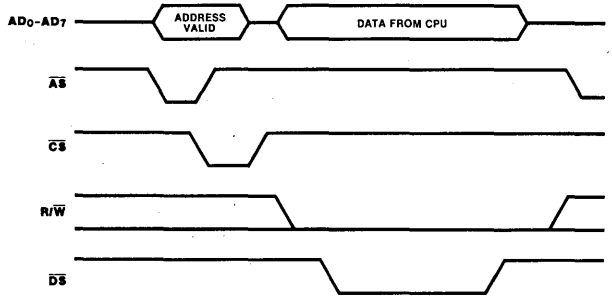


Figure 7. Z-BUS Write Cycle Timing

CPU Interfaces (Continued)

registers ($C/\overline{D} = 1$). Read and write to all registers except the FIFO buffer[1] are two-step operations, described as follows (Figure 8). First, write the address ($C/\overline{D} = 1$) of the register to be accessed into the Pointer Register (State 0); second, read or write ($C/\overline{D} = 1$) to the register pointed at previously (State 1). Continuous status monitoring can be performed in State 1 by continuous Control Read operations ($C/\overline{D} = 1$).

[1]The FIFO buffer can also be accessed by this two-step operation.

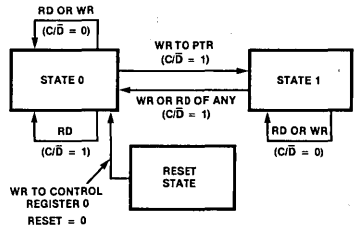


Figure 8. Register Access in Non-Z-BUS Mode

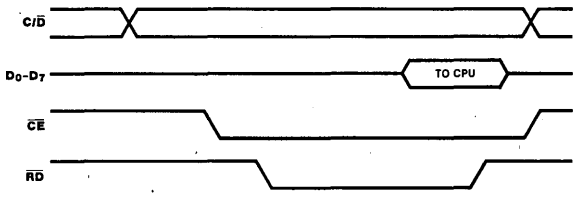


Figure 9. Non-Z-BUS Read Cycle Timing

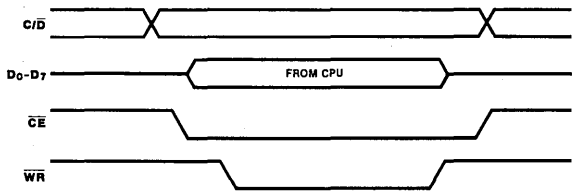


Figure 10. Non-Z-BUS Write Cycle Timing

WAIT Operation

When data is output by the CPU, the REQ/WT (WAIT) pin is active (Low) only when the FIFO buffer is full, the chip is selected, and the FIFO buffer is addressed. WAIT goes inactive when the FIFO buffer is not full.

When data is input by the CPU, the REQ/WT pin becomes active (Low) only when the FIFO buffer is empty, the chip is selected, and the FIFO buffer is addressed. WAIT goes inactive when the FIFO buffer is not empty.

Interrupt Operation

The FIO supports Zilog's prioritized daisy chain interrupt protocol for both Z-BUS and non-Z-BUS operating modes (for more details refer to the Zilog *Z-BUS Summary*).

Each side of the FIO has seven sources of interrupt. The priorities of these devices are fixed in the following order (highest to lowest): Mailbox Message, Change in Data Direction, Pattern Match, Status Match, Overflow/Underflow Error, Buffer Full, and Buffer Empty. Each interrupt source has three bits that control how it generates the interrupt. These bits are Interrupt Pending (IP), Interrupt Enable (IE), and Interrupt Under Service (IUS).

In addition, each side of the FIO has an interrupt vector and four bits controlling the FIO interrupt logic. These bits are Vector

Interrupt Operation (Continued)

Includes Status (VIS), Master Interrupt Enable (MIE), Disable Lower Chain (DLC), and No Vector (NV).

A typical Interrupt Acknowledge cycle for Z-BUS operation is shown in Figure 11 and for non-Z-BUS operation in Figure 12. The only difference is that in Z-BUS mode, $\overline{\text{INTACK}}$ is latched by $\overline{\text{AS}}$, and in non-Z-BUS mode $\overline{\text{INTACK}}$ is not latched.

When MIE = 1, reading the vector always includes status, independent of the state of the VIS bit. In this way, when VIS = 0, all information can be obtained with one additional read, thus conserving vector space. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified.

In non-Z-BUS mode, the IPs do not get set while in State 1. Therefore, to minimize interrupt latency, the FIO should be left in State 0. In Z-BUS mode IPs are set by an $\overline{\text{AS}}$ following the event.

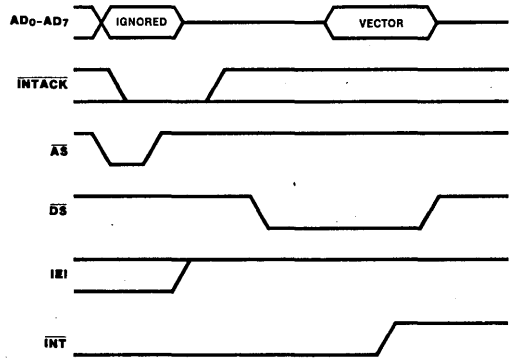


Figure 11. Z-BUS Interrupt Acknowledge Cycle

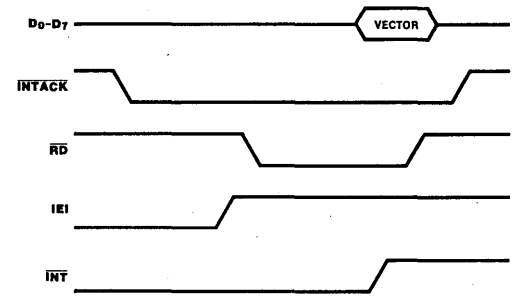


Figure 12. Non-Z-BUS Interrupt Acknowledge Cycle

CPU to CPU Operation

DMA Operation. The FIO is particularly well suited to work with a DMA in both Z-BUS and non-Z-BUS modes. A data transfer between the FIO and system memory can take place during every machine cycle on both sides of the FIO simultaneously.

In Z-BUS mode, the $\overline{\text{DMASTB}}$ pin (DMA Strobe) is used to read or write into the FIFO buffer. The R/$\overline{\text{W}}$ (Read/Write) and $\overline{\text{DS}}$ (Data Strobe) signals are ignored by the FIO; however, the $\overline{\text{CS}}$ (Chip Select) signal is not ignored and therefore must be kept invalid. Figures 13 and 14 show typical timing.

In Non-Z-BUS mode, the $\overline{\text{DACK}}$ pin (DMA Acknowledge) is used to tell the FIO that its DMA request is granted. After $\overline{\text{DACK}}$ goes Low, every read or write to the FIO goes into the FIFO buffer. Figures 15 and 16 show typical timing.

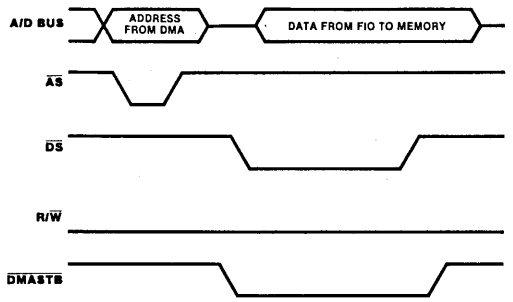


Figure 13. Z-BUS FIO to Memory Data Transaction

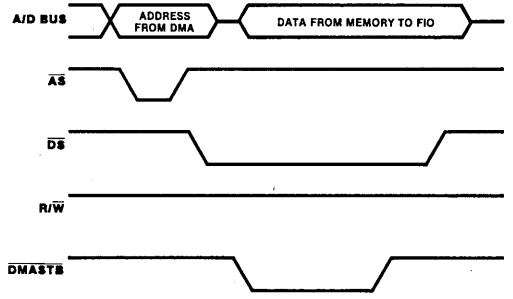


Figure 14. Z-BUS Memory to FIO Data Transaction

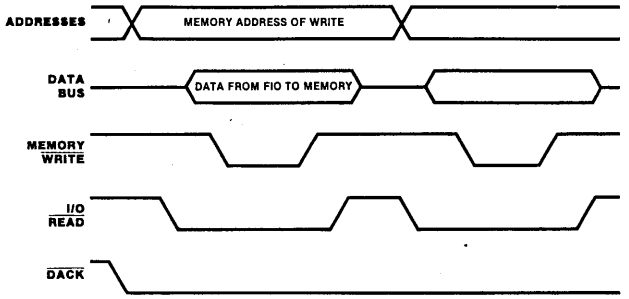


Figure 15. Non-Z-BUS FIO to Memory Transaction

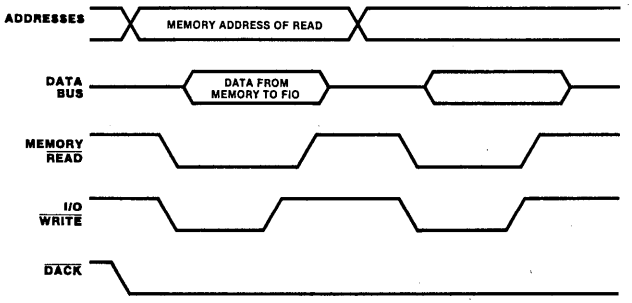
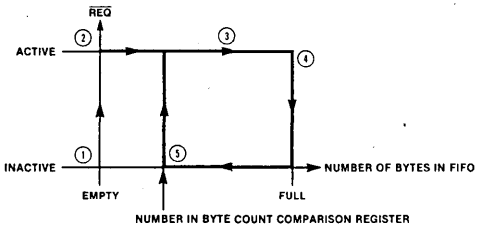


Figure 16. Non-Z-BUS Memory to FIO Data Transaction

CPU to CPU Operation (Continued)

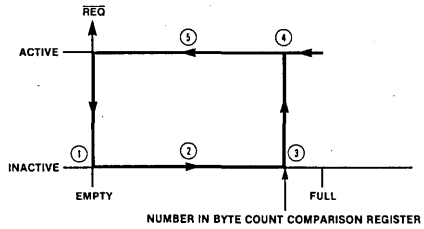
The FIO provides a special mode to enhance its DMA transfer capability. When data is written into the FIFO buffer, the REQ/WT (REQUEST) pin is active (Low) until the FIFO buffer is full. It then goes inactive and stays inactive until the number of bytes in the FIFO buffer is equal to the value programmed into the Byte Count Comparison register. Then the REQUEST signal goes active and the sequence starts over again (Figure 17).



NOTES:
1. FIFO empty.
2. REQUEST enabled, FIO requests DMA transfer.
3. DMA transfers data into the FIO.
4. FIFO full, REQUEST inactive.
5. The FIFO empties from the opposite port until the number of bytes in the FIFO buffer is the same as the number programmed in the Byte Count Comparison register.

Figure 17. Byte Count Control: Write to FIO

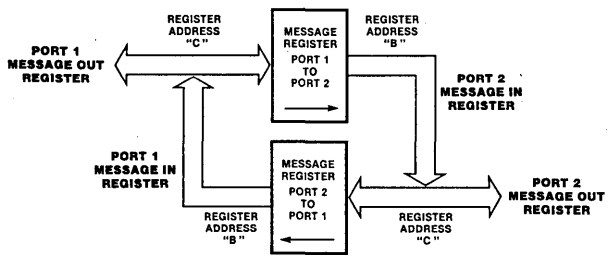
When data is read from the FIO, the REQ/WT pin (REQUEST) is inactive until the number of bytes in the FIFO buffer is equal to the value programmed in the Byte Count Comparison register. The REQUEST signal then goes active and stays active until the FIFO buffer is empty. When empty, REQUEST goes inactive and the sequence starts over again (Figure 18).



NOTES:
1. FIFO empty.
2. CPU/DMA fills FIFO buffer from the opposite port.
3. Number of bytes in FIFO buffer is the same as the number of bytes programmed in the Byte Count Comparison register.
4. REQUEST goes active.
5. DMA transfers data out of FIFO until it is empty.

Figure 18. Byte Count Control: Read from FIO

Message Registers. Two CPUs can communicate through a dedicated "mailbox" register without involving the 128 × 8 bit FIFO buffer (Figure 19). This mailbox approach is useful for transferring control parameters between the interfacing devices on either side of the FIO without using the FIFO buffer. For example, when Port 1's CPU writes to the Message Out register, Port 2's message IP is set. If interrupts are enabled, Port 2's CPU is interrupted. Port 2's message IP status is readable from the Port 1 side. When Port 2's CPU reads the data from its Message In register, the Port 2 IP is cleared. Thus, Port 1's CPU can read when the message has been read and can now send another message or follow whatever protocol that is set up between the two CPU's. The same transfer can also be made from Port 2's CPU to Port 1's CPU.



NOTE: Usable only for CPU/CPU interface.

Figure 19. Message Register Operation

CPU to CPU Operation (Continued)

CLEAR (Empty) FIFO Operation. The $\overline{\text{CLEAR}}$ FIFO bit (active Low) clears the FIFO buffer of data. Writing a 0 to this bit empties the FIFO buffer, inactivates the $\overline{\text{REQUEST}}$ line, and disables the handshake (if programmed). The $\overline{\text{CLEAR}}$ bit does not affect any control or data register. To remove the $\overline{\text{CLEAR}}$ state, write a 1 to the $\overline{\text{CLEAR}}$ bit.

In CPU/CPU mode, under program control, only one of the ports can empty the FIFO by writing to its Control Register 3, bit 6. The Port 1 CPU must program bit 7 in Control Register 3 to determine which port controls the CLEAR FIFO operation (0 = Port 1 control; 1 = Port 2 control).

Direction of Data Transfer Operation. The Data Direction bit controls the direction of data transfer in the FIFO buffer. The Data Direction bit is defined as 0 = output from CPU and 1 = input to CPU. This bit reads correctly when read by either port's CPU. For example, if Port 1's CPU reads a 0 (CPU output) in its Data Direction bit, then Port 2's CPU reads a 1 (input to CPU) in its Data Direction bit.

In CPU/CPU mode, under program control, only one of the ports can control the direction of data transfer. The Port 1 CPU must program bit 5 in Control Register 3 to determine which port controls the data direction (0 = Port 1 control; 1 = Port 2 control). Figure 20 shows FIO data transfer options.

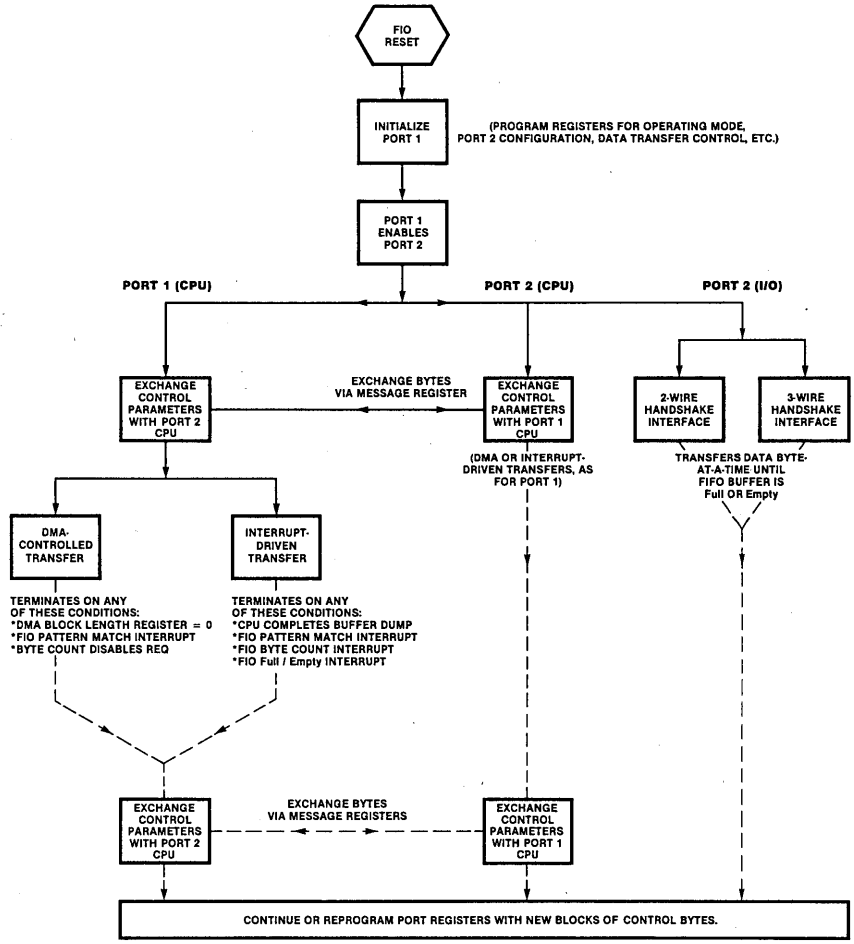


Figure 20. FIO Data Transfer Options

CPU to I/O Operation

When Port 2 is programmed in the Interlocked 2-Wire Handshake mode or the 3-Wire Handshake mode, and Port A is programmed in Z-BUS or non-Z-BUS Microprocessor mode, the FIO interfaces a CPU and a peripheral device. In the Interlocked 2-Wire Handshake mode, RFD/$\overline{\text{DAV}}$ and $\overline{\text{ACKIN}}$ strobe data to and from Port 2. In the 3-Wire Handshake mode, RFD/$\overline{\text{DAV}}$, $\overline{\text{DAV}}$/DAC, and DAC/RFD signals control data flow.

Interlocked 2-Wire Handshake. In the Interlocked Handshake, the action of the FIO must be acknowledged by the other half of the handshake before the next action can take place. In output mode, Port 2 does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, in input mode, Port 2 does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging Port 2's acceptance of the last byte. This allows the FIO to directly interface to a Z8's port, a CIO's port, a UPC's port, another FIO port, or another FIFO Z8060, with no external logic (Figures 21 and 22).

3-Wire Handshake. The 3-Wire Handshake is designed for applications in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate that an input port is ready for new data or that it has accepted the present data. In the 3-Wire Handshake, the rising edge of the RFD status line indicates that the port is ready for data, and the rising edge of the DAC status line indicates that the data has been accepted. With 3-Wire Handshake, the lines of many input ports can be bussed together with open-drain drivers and the output port knows when all of the ports are ready and have accepted the data. This handshake is the same handshake used in the IEEE-488 Instruments. Since the port's direction can be changed under software control, bidirectional IEEE-488-type transfers can be performed. Figures 23 and 24 show the timings associated with 3-Wire Handshake communications.

CLEAR FIFO Operation. In CPU-to-I/O operation, the CLEAR FIFO operation can be performed by the CPU side (Port 1) under software control as previously explained. The CLEAR FIFO operation can also be performed under hardware control by defining the $\overline{\text{CLEAR}}$ pin of Port 2 as an input (Control Register 3, bit 7 = 1).

For cascading purposes, the $\overline{\text{CLEAR}}$ pin can also be defined as an output (Control Register 3, bit 7 = 0), which reflects the current state of the $\overline{\text{CLEAR}}$ FIFO bit. It can then empty other FIOs or initialize other devices in the system.

Data Direction Control. In CPU-to-I/O mode, the direction of data transfer can be controlled by the CPU side (Port 1) under software control as previously explained. The data direction can also be determined by hardware control by defining the Data Direction pin of Port 2 as an input (Control Register 3, bit 5 = 1).

For cascading purposes, the Data Direction pin can also be defined as an output (Control Register 3, bit 5 = 0) pin which reflects the current state of the Data Direction bit. It can then be used to control the direction of data transfer for other FIOs or for external logic.

On the Port 2 side, when data direction is 0, Port 2 is in Output Handshake mode. When data direction is 1, Port 2 is in Input Handshake mode.

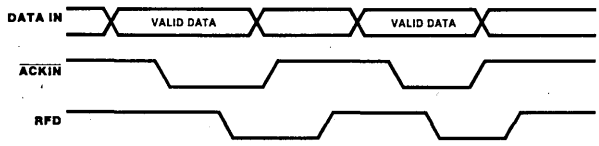


Figure 21. Interlocked Handshake Timing (Input) Port 2 Side Only

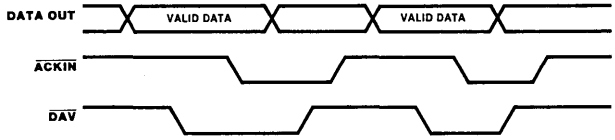


Figure 22. Interlocked Handshake Timing (Output) Port 2 Side Only

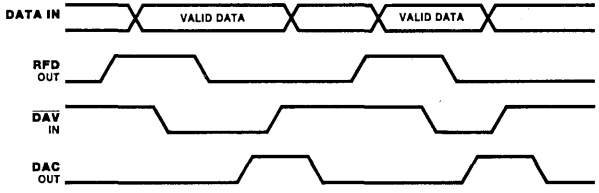


Figure 23. Input (Acceptor) Timing IEEE-488 HS Port: Port 2 Side Only

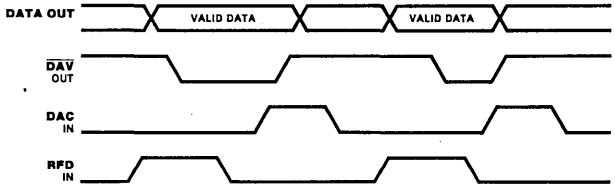


Figure 24. Output (Source) Timing IEEE-488 HS Port: Port 2 Side Only

Programming

The programming of the FIO is greatly simplified by the efficient grouping of the various operation modes in the control registers. Since all of the control registers are read/write, the need for maintaining their image in system memory is eliminated. Also, the read/write feature of the registers aids in system debugging.

Each side of the FIO has 16 registers. All 16 registers are used by the Port 1 side; Control register 2 is not used on the Port 2 side. All registers are addressable 0_H through F_H.

In the Z-BUS Low Byte mode, the FIO allows two methods for register addressing under control of the Right Justify Address (RJA) bit in Control register 0. When RJA = 0, address bus bits 1-4 are used for register addressing and bits 1, 5, 6, and 7 are ignored (Table 4). When RJA = 1, bits 0-3 are used for the register addresses, and bits 4-7 are ignored.

Control Registers. These four registers specify FIO operation. The Port 2 side control registers operate only if the Port 2 device is a CPU. The Port 2 CPU can control interface operations, including data direction, only when enabled by the setting of bit 0 in the Port 1 side of Control Register 2. A 1 in bit 1 of the same register enables the handshake logic.

Interrupt Status Registers. These four registers control and monitor the priority interrupt functions for the FIO.

Interrupt Vector Register. This register stores the interrupt service routine address. This vector is placed on D_0-D_7 when IUS is set by the Interrupt Acknowledge signal from the CPU. When bit 4 (Vector Includes Status) is set in Control Register 0, the reason for the interrupt is encoded within the vector address in bits 1, 2, and 3. If bit 5 is set in Control register 0, no vector is output by the FIO during an Interrupt Acknowledge cycle. However, IUS is set as usual.

Non Z-BUS	D_7-D_4	D_3	D_2	D_1	D_0	
Z-BUS High		A_3	A_2	A_1	A_0	
Z-BUS Low RJA=0	AD_7-AD_5	AD_4	AD_3	AD_2	AD_1	AD_0
Z-BUS Low RJA=1	AD_7-AD_4	AD_3	AD_2	AD_1	AD_0	
Description						
Control Register 0	x	0	0	0	0	x
Control Register 1	x	0	0	0	1	x
Interrupt Status Register 0	x	0	0	1	0	x
Interrupt Status Register 1	x	0	0	1	1	x
Interrupt Status Register 2	x	0	1	0	0	x
Interrupt Status Register 3	x	0	1	0	1	x
Interrupt Vector Register	x	0	1	1	0	x
Byte Count Register	x	0	1	1	1	x
Byte Count Comparison Register	x	1	0	0	0	x
Control Register 2*	x	1	0	0	1	x
Control Register 3	x	1	0	1	0	x
Message Out Register	x	1	0	1	1	x
Message In Register	x	1	1	0	0	x
Pattern Match Register	x	1	1	0	1	x
Pattern Mask Register	x	1	1	1	0	x
Data Buffer Register	x	1	1	1	1	x

x = Don't Care
*Register is only on Port 1 side

Table 4. FIO Register Address Summary

Programming (Continued)

Byte Count Compare Register. This register contains a value compared with the byte count in the Byte Count register. If the Byte Count Compare interrupt is enabled, an interrupt will occur upon compare.

Message Out Register. Either CPU can place a message in its Message Out register. If the opposite side Message register interrupt is enabled, the receiving side CPU will receive an interrupt request, advising that a message is present in its Message In register. Bit 5 in Control Register 1 on the initiating side is set when a message is written. It is cleared when the message is read by the receiving CPU.

Message In Register. This register receives a message placed in the Message Out register by the opposite side CPU.

Pattern Match Register. This register contains a bit pattern matched against the byte in the Data Buffer register. When these patterns match, a Pattern Match interrupt will be generated, if previously enabled.

Pattern Mask Register. The Pattern Mask register may be programmed with a bit pattern mask that limits comparable bits in the Pattern Match register to non-masked bits (1 = mask).

Data Buffer Register. This register contains the data to be read from or written to the FIFO buffer.

Byte Count Register. This is a read-only register, containing the byte count for the FIFO buffer. The byte count is derived by subtracting the number of bytes read from the buffer from the number of bytes written into the buffer. The count is "frozen" for an accurate reading by setting bit 6 (Freeze Status register) in Control Register 1. This bit is cleared when the Byte Count register read is completed.

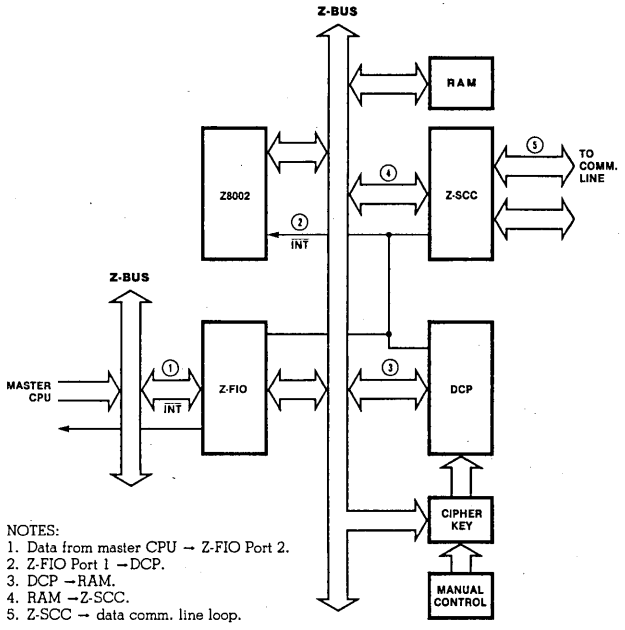
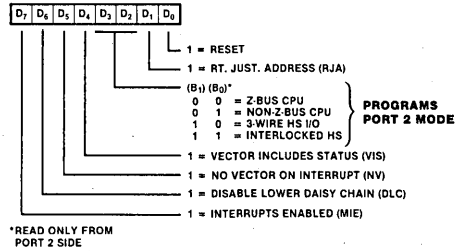


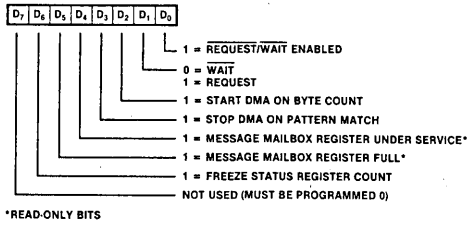
Figure 25. Typical Application: Node Controller

Registers

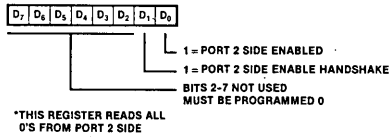
Control Register 0
Address: 0000
(Read/Write)



Control Register 1
Address: 0001
(Read/Write)



Control Register 2*
Address: 1001
(Read/Write)



Control Register 3
Address: 1010
(Read/Write)

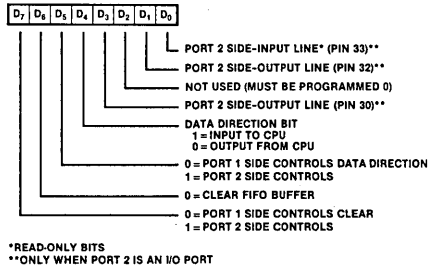


Figure 26. Control Registers

Interrupt Status Register 0
Address: 0010
(Read/Write)

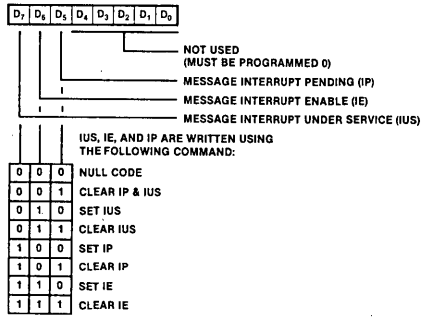
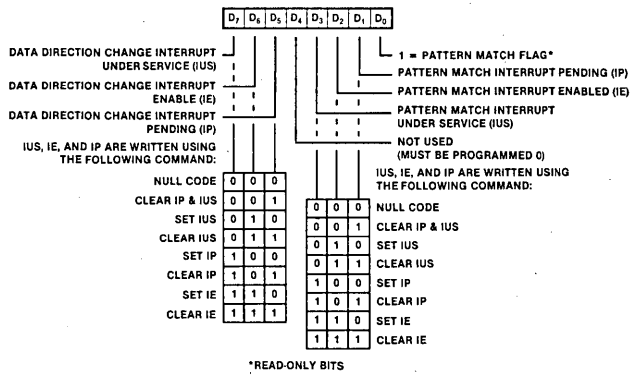
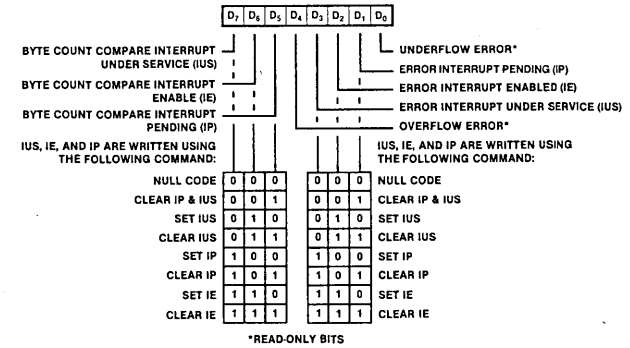


Figure 27. Interrupt Status Registers

Interrupt Status Register 1
Address: 0011
(Read/Write)



Interrupt Status Register 2
Address: 0100
(Read/Write)



Interrupt Status Register 3
Address: 0101
(Read/Write)

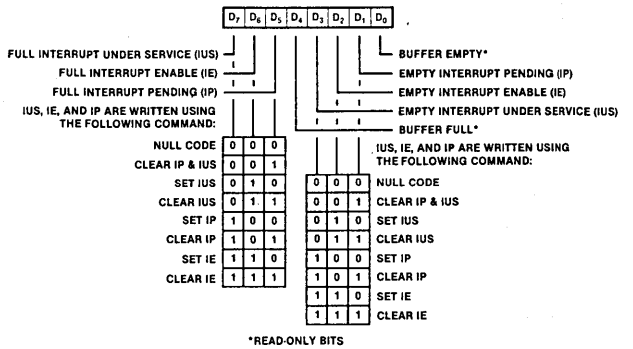


Figure 27. Interrupt Status Registers (Continued)

Registers
(Continued)

Byte Count Register
Address: 0111
(Read Only)

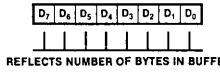


Figure 28. Byte Count Register

Interrupt Vector Register
Address: 0110
(Read/Write)

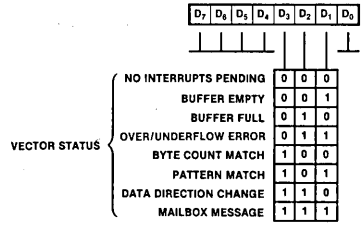


Figure 29. Interrupt Vector Register

Pattern Match Register
Address: 1101
(Read/Write)

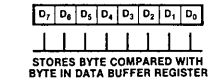


Figure 30. Pattern Match Register

Pattern Mask Register
Address: 1110
(Read/Write)

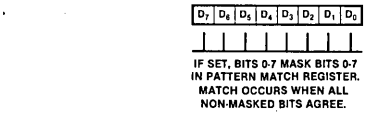


Figure 31. Pattern Mask Register

Data Buffer Register
Address: 1111
(Read/Write)

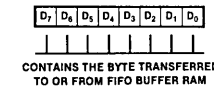


Figure 32. Data Buffer Register

Byte Count Comparison Register
Address: 1000
(Read/Write)

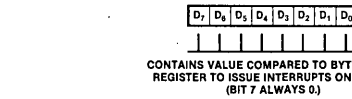


Figure 33. Byte Count Comparison Register

Message Out Register
Address: 1011
(Read/Write)

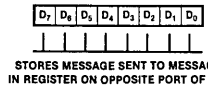


Figure 34. Message Out Register

Message In Register
Address: 1100
(Read Only)

Figure 35. Message In Register

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

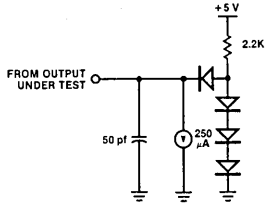
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

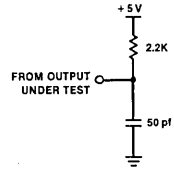
The DC characteristics and capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information



Standard Test Load



Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
			0.5	V	$I_{OL} = +3.2$ mA
I_{IL}	Input Leakage	−10.0	+10.0	µA	$0.4 \leq V_{IN} \leq +2.4V$
I_{OL}	Output Leakage	−10.0	+10.0	µA	$0.4 \leq V_{OUT} \leq +2.4V$
I_{LM}	Mode Pins Input Leakage (Pins 19 and 21)	−100	+10.0	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	V_{CC} Supply Current		200	mA	

$V_{CC} = 5\ V \pm 5\%$ unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Condition
C_{IN}	Input Capacitance		10	pf	
C_{OUT}	Output Capacitance		15	pf	
$C_{I/O}$	Bidirectional Capacitance		20	pf	

Unmeasured pins returned to ground.

Inputs

tr	Any Input Rise Time		100	ns
tf	Any Input Fall Time		100	ns

f = 1 MHz, over specified temperature range.

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwAS	$\overline{AS}$ Low Width	70		50		1
2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
4	TsCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
5	ThCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
6	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		40		1
7	TsA(DS)	Address to $\overline{DS}$ ↓ (with $\overline{AS}$ ↑ to $\overline{DS}$ ↓ = 60 ns)	120		100		
8	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
9	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
10	TwDS	$\overline{DS}$ Low Width	390		250		
11	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
12	TdDS(DRV)	$\overline{DS}$ (Read) ↓ to Address Data Bus Driven	0		0		
13	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180	
14	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		
15	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
16	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	2
17	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		
18	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
19	Trc	Valid Access Recovery Time	1000		650		3

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{DS}$ of one FIO access to $\overline{DS}$ of another FIO access (either read or write).

* All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".

† Units in nanoseconds (ns).

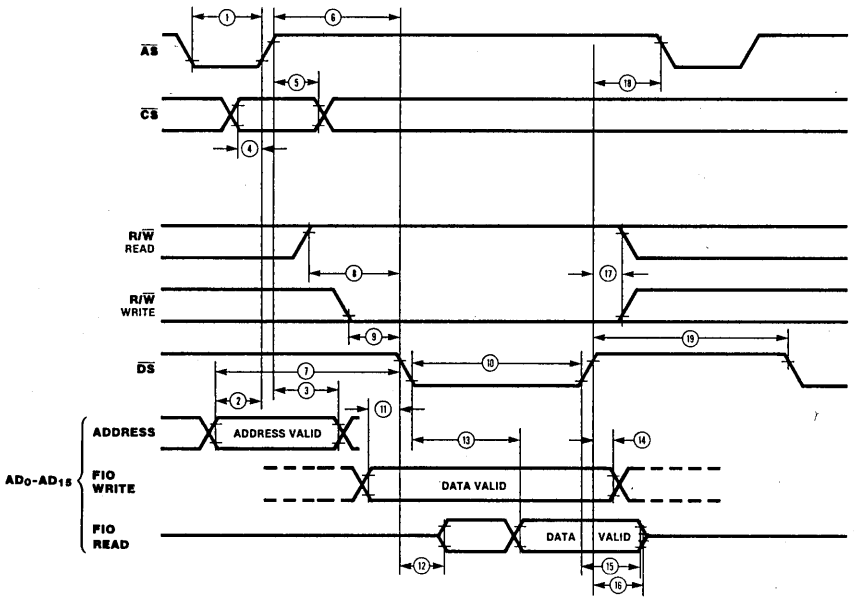


Figure 36. Z-BUS CPU Interface Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
20	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
21	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
22	TsDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		250		180	
23	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	390		250		
24	TdAS(IEO)	$\overline{AS}$ ↑ to IEO ↓ Delay ($\overline{INTACK}$ Cycle)		350		250	4
25	TdIEI(IEO)	IEI to IEO Delay		150		100	4
26	TsIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		70		
27	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↑ Hold Time	50		30		4
28	TdDS(INT)	$\overline{DS}$ ($\overline{INTACK}$ Cycle) to $\overline{INT}$ Delay		900		800	
29	TdDCST	Interrupt Daisy Chain Settle Time					4

NOTES:

4. The parameters for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{AS}$ to $\overline{DS}$ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral and TdIEI(IEO) for each peripheral, separating them in the chain.

† Units in nanoseconds (ns).

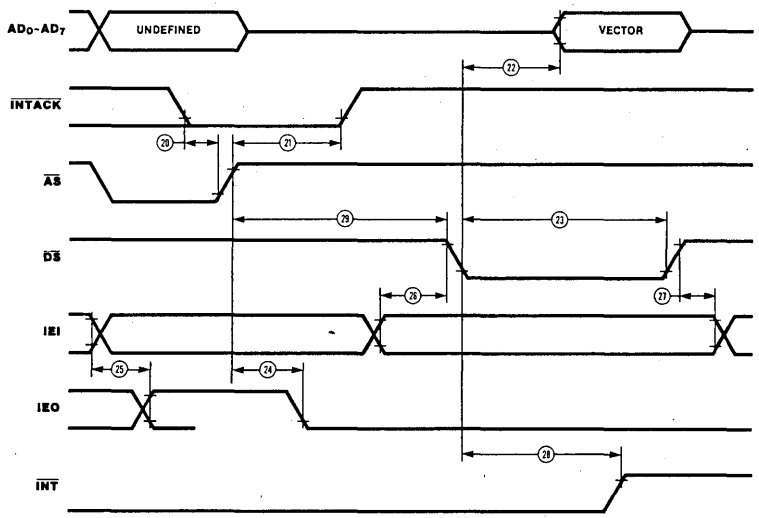


Figure 37. Z-BUS CPU Interrupt Acknowledge Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
30	TdMW(INT)	Message Write to $\overline{INT}$ Delay		1		1	5
31	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay		1		1	6
32	TdPMW(INT)	Pattern Match to $\overline{INT}$ Delay (Write Case)		1		1	
33	TdPMR(INT)	Pattern Match (Read Case) to $\overline{INT}$ Delay		1		1	
34	TdSC(INT)	Status Compare to $\overline{INT}$ Delay		1		1	6
35	TdER(INT)	Error to $\overline{INT}$ Delay		1		1	
36	TdEM(INT)	Empty to $\overline{INT}$ Delay		1		1	6
37	TdFL(INT)	Full to $\overline{INT}$ Delay		1		1	6
38	TdAS(INT)	$\overline{AS}$ to $\overline{INT}$ Delay					

NOTES:
5. Write is from the other side of FIO.
6. Write can be from either side, depending on programming of FIO.

† Units equal to $\overline{AS}$ Cycles + ns.

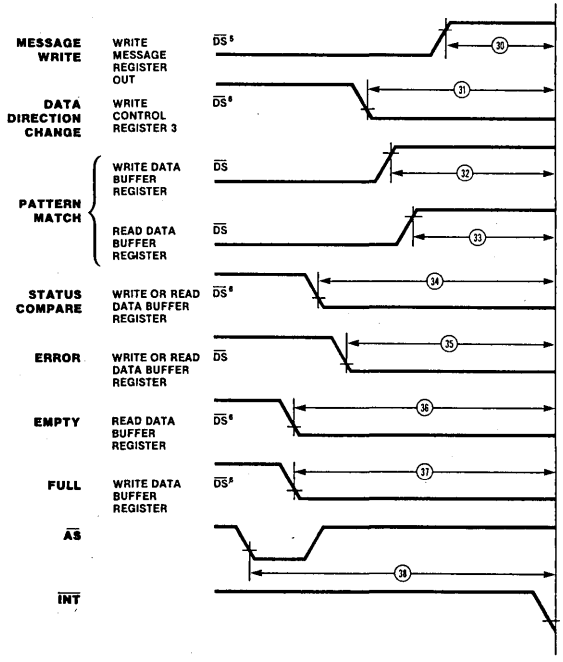


Figure 38. Z-BUS Interrupt Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TdDS(WAIT)	$\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Delay		190		160	
2	TdDS1(WAIT)	$\overline{DS1}$ ↑ to $\overline{WAIT}$ ↑ Delay		1000		1000	
3	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay		1000		1000	1
4	TdDS(REQ)	$\overline{DS}$ ↓ to $\overline{REQ}$ ↑ Delay		350		300	
5	TdDMA(REQ)	$\overline{DMASTB}$ ↓ to $\overline{REQ}$ ↑ Delay		350		300	
6	TdDS1(REQ)	$\overline{DS1}$ ↑ to $\overline{REQ}$ ↓ Delay		1000		1000	
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ ↓ Delay		1000		1000	
8	TdSU(DMA)	Data Setup Time to $\overline{DMASTB}$	200		150		
9	TdH(DMA)	Data Hold Time to $\overline{DMASTB}$	30		20		
10	TdDMA(DR)	$\overline{DMASTB}$ ↓ to Valid Data		150		100	
11	TdDMA(DRH)	$\overline{DMASTB}$ ↑ to Data Not Valid	0		0		
12	TdDMA(DR2)	$\overline{DMASTB}$ ↑ to Data Bus Float		70		45	

NOTES:
1. The delay is from $\overline{DAV}$ for 3-Wire Input Handshake. The delay is from DAC for 3-Wire Handshake.

† Units in nanoseconds (ns).

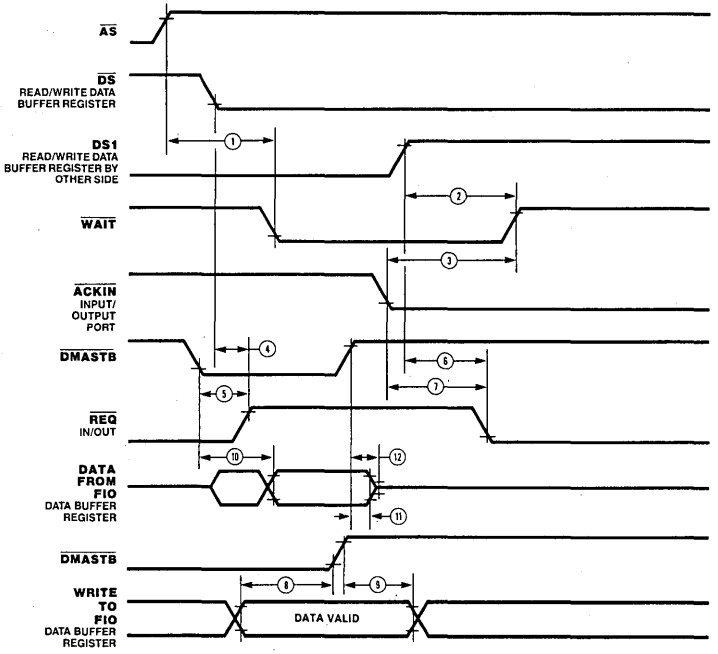


Figure 39. Z-BUS Request/Wait Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for No Reset	40		20		
2	TdASQ(DS)	Delay for $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for No Reset	50		30		
3	Tw(AS + DS)	Minimum Width of $\overline{AS}$ and $\overline{DS}$ Both Low for Reset.	500		350		1

NOTES:
1. Internal circuitry allows for the reset provided by the Z8 ($\overline{DS}$ held Low while $\overline{AS}$ pulses) to be sufficient.

† Units in nanoseconds (ns).

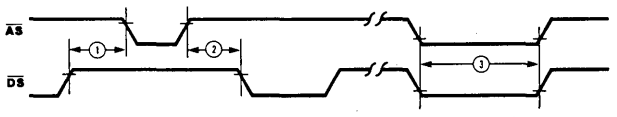


Figure 40. Z-BUS Reset Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TsA(RD)	Address Setup to $\overline{RD}$ ↓	80		80		1
2	TsA(WR)	Address Setup to $\overline{WR}$ ↓	80		80		
3	ThA(RD)	Address Hold Time to $\overline{RD}$ ↑	0		0		1
4	ThA(WR)	Address Hold Time to $\overline{WR}$ ↑	0		0		
5	TsCEI(RD)	$\overline{CE}$ Low Setup Time to $\overline{RD}$	0		0		1
6	TsCEI(WR)	$\overline{CE}$ Low Setup Time to $\overline{WR}$	0		0		
7	ThCEI(RD)	$\overline{CE}$ Low Hold Time to $\overline{RD}$	0		0		1
8	ThCEI(WR)	$\overline{CE}$ Low Hold Time to $\overline{WR}$	0		0		
9	TsCEh(RD)	$\overline{CE}$ High Setup Time to $\overline{RD}$	100		70		1
10	TsCEh(WR)	$\overline{CE}$ High Setup Time to $\overline{WR}$	100		70		
11	TwRDl	$\overline{RD}$ Low Width	390		250		
12	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
13	TdRDf(DR)	$\overline{RD}$ ↓ to Valid Data Delay		250		180	
14	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
15	TdRD(DRz)	$\overline{RD}$ ↑ to Data Bus Float		70		45	2
16	TwWRl	$\overline{WR}$ Low Width	390		250		
17	TsDW(WR)	Data Setup Time to $\overline{WR}$	0		0		
19	Trc(WR)	Write Valid Access Recovery Time	1000		650		
20	Trc(RD)	Read Valid Access Recovery Time	$1000 + WR_p$		$650 + WR_p$		3

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time the output has changed 0.5V from steady state with minimum ac load and maximum dc load.
3. Recovery time equal to Trc(WR) + write pulse width of the opposite side.

† Units in nanoseconds (ns).

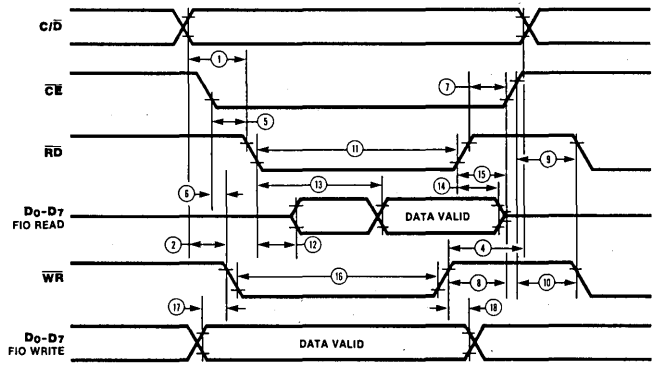


Figure 41. Non-Z-BUS CPU Interface Timing

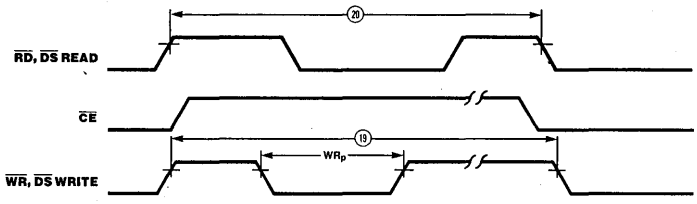


Figure 42. Z-BUS/Non-Z-BUS Recovery Time

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes†
20	TdIEI(IEO)	IEI to IEO Delay		150		100	4
21	TdI(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		350		250	4
22	TsIEI(RDA)	IEI Setup Time to $\overline{RD}$ (Acknowledge)	100		70		4
23	TdRD(DR)	$\overline{RD}$ ↓ to Vector Valid Delay		250		180	
24	TwRD1(IA)	Read Low Width (Interrupt Acknowledge)	390		250		
25	ThIA(RD)	$\overline{INTACK}$ ↑ to $\overline{RD}$ ↑ Hold Time	30		20		
26	ThIEI(RD)	IEI Hold Time to $\overline{RD}$ ↑	20		10		
27	TdRD(INT)	$\overline{RD}$ ↓ to $\overline{INT}$ ↑ Delay		900		800	
28	TdDCST	Interrupt Daisy Chain Settle Time	350		250		4

NOTES:

4. The parameter for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{INTACK}$ ↓ to $\overline{RD}$ ↓ must be greater than the sum of TdI(IEO) for the highest priority peripheral, TsIEI(RD) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.

† Units in nanoseconds (ns).

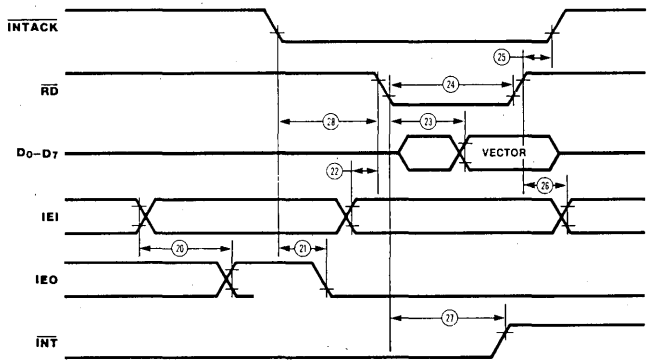


Figure 43. Non-Z-BUS Interrupt Acknowledge Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes
29	TdMW(INT)	Message Write to $\overline{INT}$ Delay					
30	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay					
31	TdPMW(INT)	Pattern Match (Write Case) to $\overline{INT}$ Delay					
32	TdPMR(INT)	Pattern Match (Read Case) to $\overline{INT}$ Delay					
33	TdSC(INT)	Status Compare to $\overline{INT}$ Delay					
34	TdER(INT)	Error to $\overline{INT}$ Delay					
35	TdEM(INT)	Empty to $\overline{INT}$ Delay					
36	TdFL(INT)	Full to $\overline{INT}$ Delay					
37	TdSO(INT)	State 0 to $\overline{INT}$ Delay		600		500	

Note: Parameter values for numbers 29 through 36 were left blank as they are software dependent.

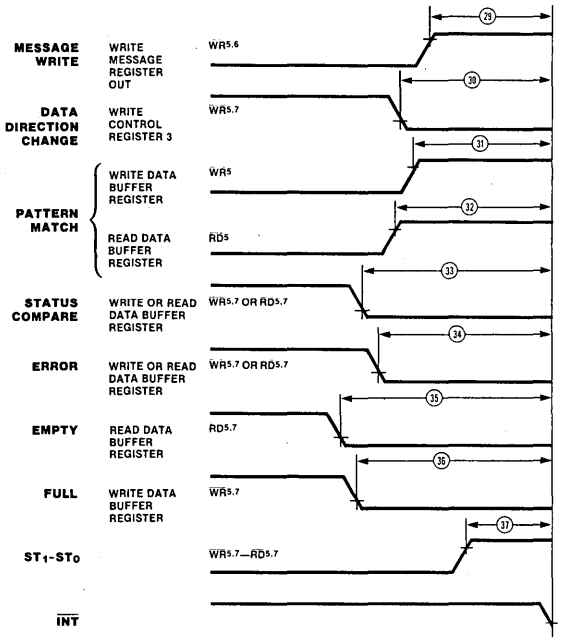


Figure 44. FIO Non-Z-BUS Interrupt Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TdCE(WT)	$\overline{CE}$ ↓ to $\overline{WAIT}$ Active		200		170	
2	TdRD1(WT)	$\overline{RD1}$ ↑ or $\overline{WR1}$ ↑ to $\overline{WAIT}$ Inactive		1000		1000	
3	TdACK(WT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ Inactive		1000		1000	1
4	TdRD(REQ)	$\overline{RD}$ ↓ or $\overline{WR}$ ↓ to $\overline{REQ}$ Inactive		350		300	
5	TdRD1(REQ)	$\overline{RD1}$ ↑ or $\overline{WR1}$ ↑ to $\overline{REQ}$ Active		1000		1000	
6	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ Active		1000		1000	
7	TdDAC(RD)	$\overline{DACK}$ ↓ to $\overline{RD}$ ↓ or $\overline{WR}$ ↓	100		80		
8	TSU(WR)	Data Setup Time to $\overline{WR}$	200				
9	Th(WR)	Data Hold Time to $\overline{WR}$	30			20	
10	TdDMA	$\overline{RD}$ ↓ to Valid Data		150		100	2
11	TdDMA(DRH)	$\overline{RD}$ ↑ to Data Not Valid	0		0		2
12	TdDMA(DRZ)	$\overline{RD}$ ↑ to Data Bus Float		70		45	2

NOTES:
1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from $\overline{DAC}$ ↑ for 3-Wire Output Handshake.
2. Only when $\overline{DACK}$ is active.

† Units in nanoseconds (ns).

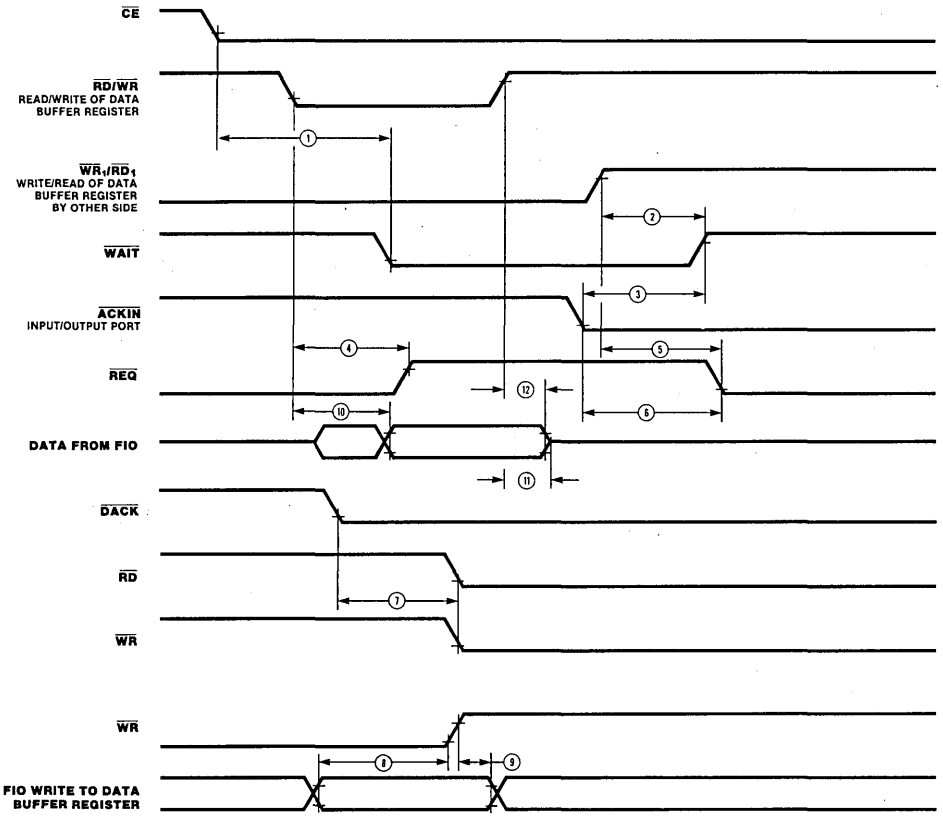


Figure 45. Non-Z-BUS Request/Wait Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓	100		70		
2	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓	100		70		
3	TwRD + WR	Width of $\overline{RD}$ and $\overline{WR}$, both Low for Reset	500		350		

NOTES:

† Units in nanoseconds (ns).

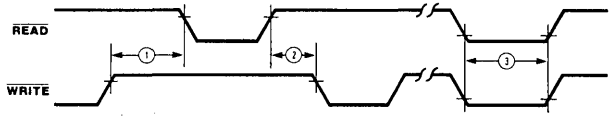


Figure 46. Non-Z-BUS Reset Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TwCLR	Width of Clear to Reset FIFO	700		700		
2	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven		210		210	
3	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float		150		150	
4	TdCLR(ACK)	$\overline{CLEAR}$ ↑ to $\overline{ACKIN}$ ↓	800		800		

NOTES:

† Units in nanoseconds (ns).

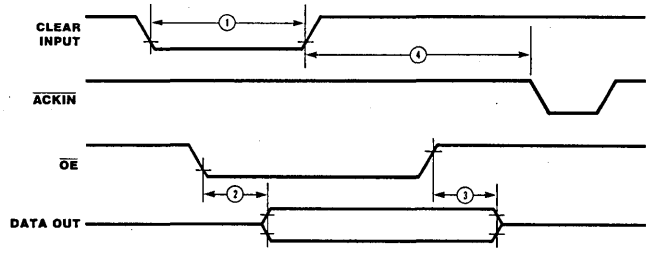


Figure 47. Port 2 Side Operation

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ to Setup Time	50		50		
2	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0	500	0	500	
3	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		0		
4	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	50		25		
5	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		0		
6	ThDO(ACK)	Data Out to $\overline{ACKIN}$ Hold Time	50		50		
7	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	0	500	0	500	
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		0		
9	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay	0		0		
10	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handshake	0	400	0	400	
11	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑)	0		0		
12	TdACKr(DAV)	$\overline{ACKIN}$ ↑ to $\overline{DAV}$ ↓	0	800	0	800	
13	TdACKf(Empty)	$\overline{ACKIN}$ ↓ to Empty	0		0		
14	TdACKf(Full)	$\overline{ACKIN}$ ↓ to Full	0		0		
15	TcACK	$\overline{ACKIN}$ Cycle Time	1		1		1

NOTES:

† Units in nanoseconds (ns), except as noted.
1. Units in microseconds.

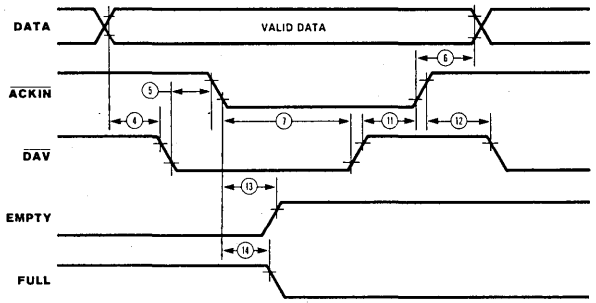


Figure 48. 2-Wire Handshake (Port 2 Side Only) Output

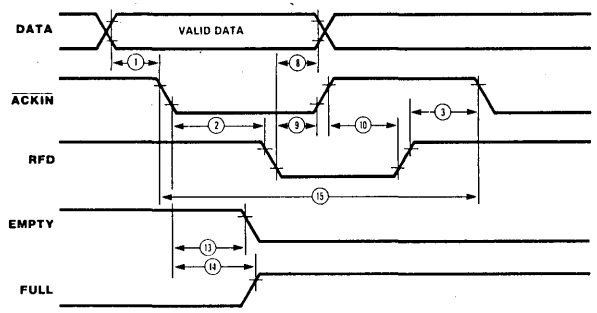


Figure 49. 2-Wire Handshake (Port 2 Side Only) Input

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes †
1	TsDI(DAV)	Data Input to $\overline{DAV}$ ↓ Setup Time	50		50		
2	TdDAVIf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0	500	0	500	
3	TdDAVf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0	500	0	500	
4	ThDI(DAC)	Data In to DAC ↑ Hold Time	0		0		
5	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay	0		0		
6	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0	500	0	500	
7	TdDAVIr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0	500	0	500	
8	TdRFDI(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0		0		
9	TsDO(DAC)	Data Out to $\overline{DAV}$ ↓					
10	TdDAVOf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0		0		
11	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0		0		
12	ThDO(DAC)	Data Out to DAC ↑ Hold Time					
13	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay		400		400	
14	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0		0		
15	TdDAVOr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0		0		
16	TdRFDO(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0	800	0	800	

NOTES:

† Units in nanoseconds (ns).

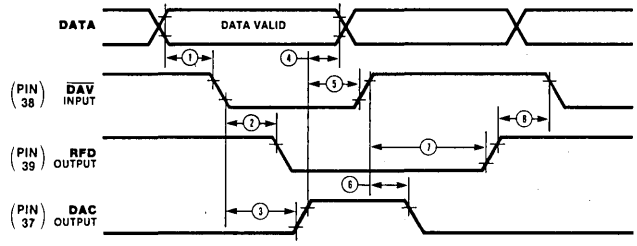


Figure 50. 3-Wire Handshake Input

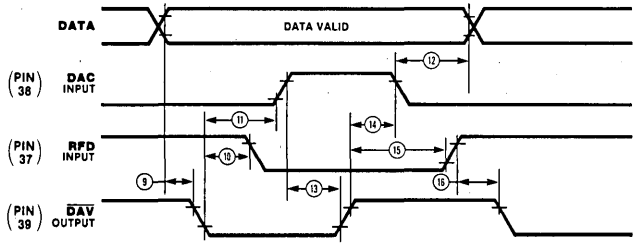


Figure 51. 3-Wire Handshake Output

Z8060/Z8560
FIFO Buffer Unit

October 1988

FEATURES

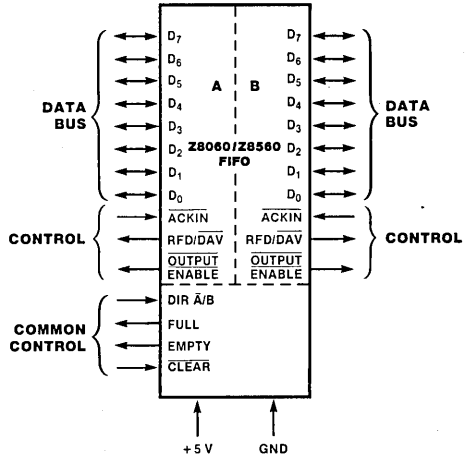
- Bidirectional, asynchronous data transfer capability.
- Large 128-bit-by-8-bit buffer memory.
- Two-wire, interlocked handshake protocol.
- Wire-ORing of empty and full outputs for sensing of multiple-unit buffers.
- 3-state data outputs.
- Connects any number of FIFOs in series to form buffer of any desired length.
- Connects any number of FIFOs in parallel to form buffer of any desired width.

GENERAL DESCRIPTION

The Z8060/Z8560 First-In First-Out (FIFO) Buffer Units consist of a 128-bit-by-8-bit memory, bidirectional data transfer and handshake logic. The structure of the FIFO unit is similar to that of other available buffer units. FIFO is a general-purpose unit; its handshake logic is compatible with that of other members of Zilog's Z8® and Z8000® Families.

FIFOs can be cascaded end-to-end without limit to form a parallel 8-bit buffer of any desired length (in 128-byte increments). Any number of single- or multiple-unit FIFO serial buffers can be connected in parallel to form buffers of any desired width (in 8-bit increments).

The FIFO buffer units are available as 28-pin packages. Figures 1 and 2 show the pin functions and pin assignments, respectively, of the FIFO device. A block diagram is shown in Figure 3.



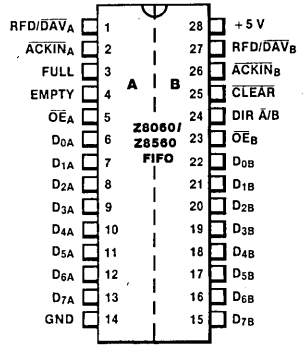


Figure 2. FIFO Pin Assignments

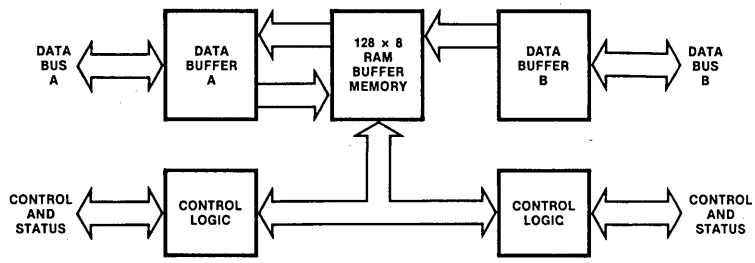


Figure 3. Functional Block Diagram

PIN DESCRIPTIONS

$\overline{\text{ACKIN}}$. *Acknowledge Input* (input, active Low). This line signals the FIFO that output data has been received by peripherals or that input data is valid.

$\overline{\text{CLEAR}}$. *Clear Buffer* (input, active Low). When set to Low, this line causes all data to be cleared from the FIFO buffer.

D_0-D_7. *Data Bus* (inputs/outputs, bidirectional). These bidirectional lines are used by the FIFO to receive and to transmit data.

DIR $\overline{\text{A}}$/B. *Direction Input A/B* (input, two control states). A High on this line signals that input data is to be received at Port B. A Low on this line signals that input data is to be received at Port A.

EMPTY. *Buffer Status* (output, active High, open-drain). A High on this line indicates that the FIFO buffer is empty.

FULL. *Buffer Status* (output, active High, open-drain). A High on this line indicates that the FIFO buffer is full.

$\overline{\text{OEA}}$, $\overline{\text{OEB}}$. *Output Enable A, Output Enable B* (inputs, active Low). When Low, $\overline{\text{OEA}}$ enables the bus drivers for Port A; when High, $\overline{\text{OEA}}$ causes the bus drivers to float to a high-impedance level. Input $\overline{\text{OEB}}$ controls the bus drivers for Port B in the same manner as $\overline{\text{OEA}}$ controls those for Port A.

RFD/$\overline{\text{DAV}}$. *Ready-for-Data/Data Available* (outputs RFD, active High; $\overline{\text{DAV}}$ active Low). RFD, when High, signals to the peripherals involved that the FIFO is ready to receive data. $\overline{\text{DAV}}$, when Low, signals to the peripherals involved that FIFO has data available to send.

FUNCTIONAL DESCRIPTION

Interlocked 2-Wire Handshake. In interlocked 2-wire handshake operation, the action of FIFO must be acknowledged by the other half of the handshake before the next action can occur. In an Output Handshake mode, the FIFO indicates that new data is available only after the external device has indicated that it is ready for the data. In an Input Handshake mode, the FIFO does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the acceptance of the last byte. This control feature allows the FIFO, with no external logic, to directly interface with the port of any CPU in the Z8 Family—a CIO, a UPC, an Z-FIO, or another FIFO. The timing for the input and output handshake operations is shown in Figures 4 and 5, respectively.

Resetting or Clearing the FIFO. The $\overline{\text{CLEAR}}$ input is used to initialize and clear the FIFO. A Low level on this input clears all data from the FIFO, allows the EMPTY output to go High and forces both outputs RFD/$\overline{\text{DAV}}_A$ and RFD/$\overline{\text{DAV}}_B$ High. A High level on $\overline{\text{CLEAR}}$ allows the data to transfer through the FIFO.

Bidirectional Transfer Control. The FIFO has bidirectional data transfer capability under control of the DIR $\overline{\text{A}}$/B input. When DIR $\overline{\text{A}}$/B is set Low, Port A becomes input handshake and Port B becomes output handshake; data transfers are then made from Port A to Port B. Setting DIR $\overline{\text{A}}$/B High reverses the handshake assignments and the direction of transfer. This bidirectional control is illustrated in Table 1.

Table 1. Bidirectional Control Function Table

DIR $\overline{\text{A}}$/B	Port A Handshake	Port B Handshake	Transfer
0	Input	Output	A to B
1	Output	Input	B to A

The FIFO buffer must be empty before the direction of transfer is changed; otherwise, the results of the change will be unpredictable. If FIFO status is unknown when a transfer direction change is to be made, the recommended procedure is:

(1) Force and hold $\overline{\text{CLEAR}}$ Low.

(2) Set DIR $\overline{\text{A}}$/B to the level required for the desired direction.

(3) Force $\overline{\text{CLEAR}}$ High.

Empty and Full Operation. The EMPTY and FULL output lines can be wire-ORed with the EMPTY and FULL lines of other FIFOs and Z-FIOs. This capability enables the user to determine the empty/full status of a buffer consisting of multiple FIFOs, Z-FIOs, or a combination of both. Table 2 shows the various states of EMPTY and FULL.

Table 2. Signals EMPTY and FULL Operation Table

Number of Bytes in FIFO	EMPTY	FULL
0	High	Low
1-127	Low	Low
128	Low	High

Interconnection Example. Figure 6 illustrates a simplified block diagram showing the manner in which FIFOs can be interconnected to extend a Z-FIO buffer.

Output Enable Operation. The FIFO provides a separate Output Enable ($\overline{\text{OE}}$) signal for each port of the buffer. An $\overline{\text{OE}}$ output is valid only when its port is in the Output Handshake mode. The control of this output function is shown in Table 3. Signal $\overline{\text{OE}}$ operates with lines DIR $\overline{\text{A}}$/B. A High on a valid $\overline{\text{OE}}$ line 3-states its port's data bus but does not affect the handshake operation. A Low level on a valid $\overline{\text{OE}}$ enables the data bus outputs if its port is in the Output Handshake mode. Note that the handshake operation is unaffected by the Output Enable pin.

Table 3. Output Control Function Table

DIR $\overline{\text{A}}$/B	$\overline{\text{OE}}_A$	$\overline{\text{OE}}_B$	Function
0	X	0	Disable Port A Output Enable Port B Output
0	X	1	Disable Port A Output Disable Port B Output
1	0	X	Enable Port A Output Disable Port B Output
1	1	X	Disable Port A Output Disable Port B Output

NOTE: X = Don't care.

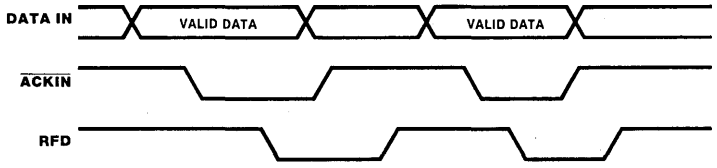


Figure 4. Two-Wire Interlocked Handshake Timing (input)

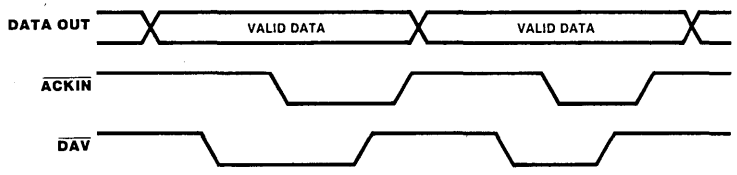


Figure 5. Two-Wire Interlocked Handshake Timing (output)

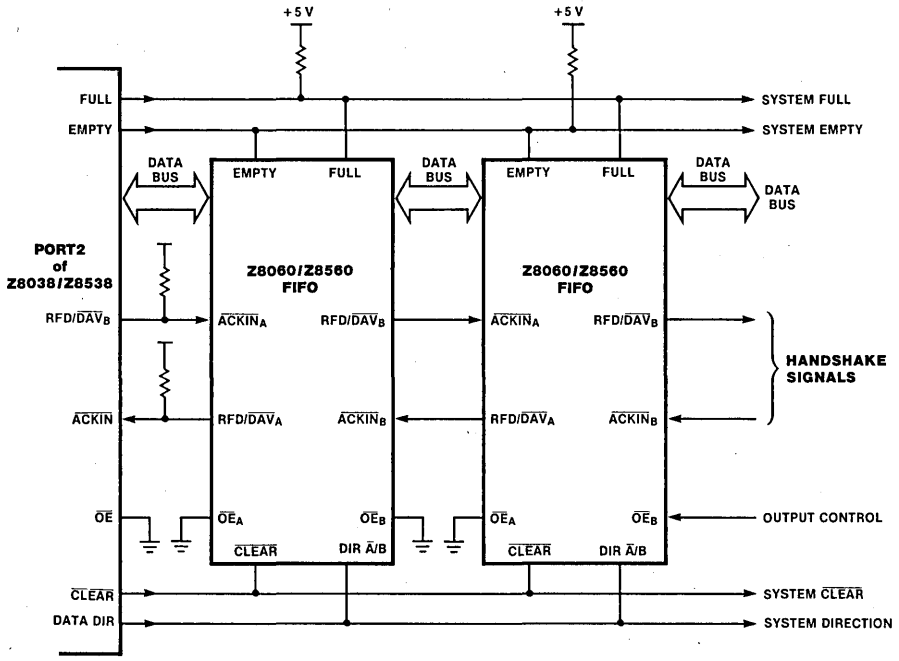


Figure 6. Typical Interconnection (Simplified Diagram)

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7V
Operating Ambient Temperature. . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The AC characteristics and capacitance sections listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Standard conditions are as follows:

- $+4.75V \leq V_{CC} \leq +5.25V$
- GND = 0V
- T_A as specified in Ordering Information. All AC parameters assume a load capacitance of 50 pf max.

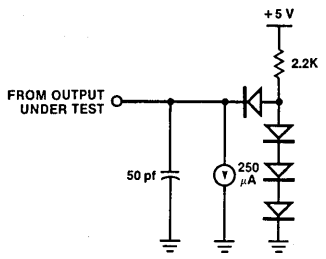


Figure 7. Standard Test Load

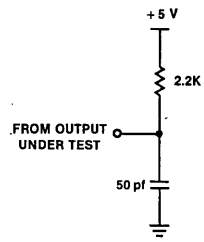


Figure 8. Open-Drain Test Load

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\,\mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = 2.0$ mA
			0.5	V	$I_{OL} = 3.2$ mA
I_{IL}	Input Leakage		±10	μA	$0.4 \leq V_{IN} \leq 2.4V$
I_{OL}	Output Leakage		±10	μA	$0.4 \leq V_{OUT} \leq 2.4V$
I_{CC}	V_{CC} Supply Current		200	mA	

NOTE: $V_{CC} = +5V \pm 5\%$ unless otherwise specified over specified temperature range.

CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C_{IN}	Input Capacitance		10	pf
C_{OUT}	Output Capacitance		15	pf
$C_{I/O}$	Bidirectional Capacitance		20	pf
Input				
tr	Any input rise time		100	ns
tf	Any input fall time		100	ns

Over specified temperature range; f = 1 MHz.
Unmeasured pins returned to ground.

2-WIRE INTERLOCKED HANDSHAKE TIMING

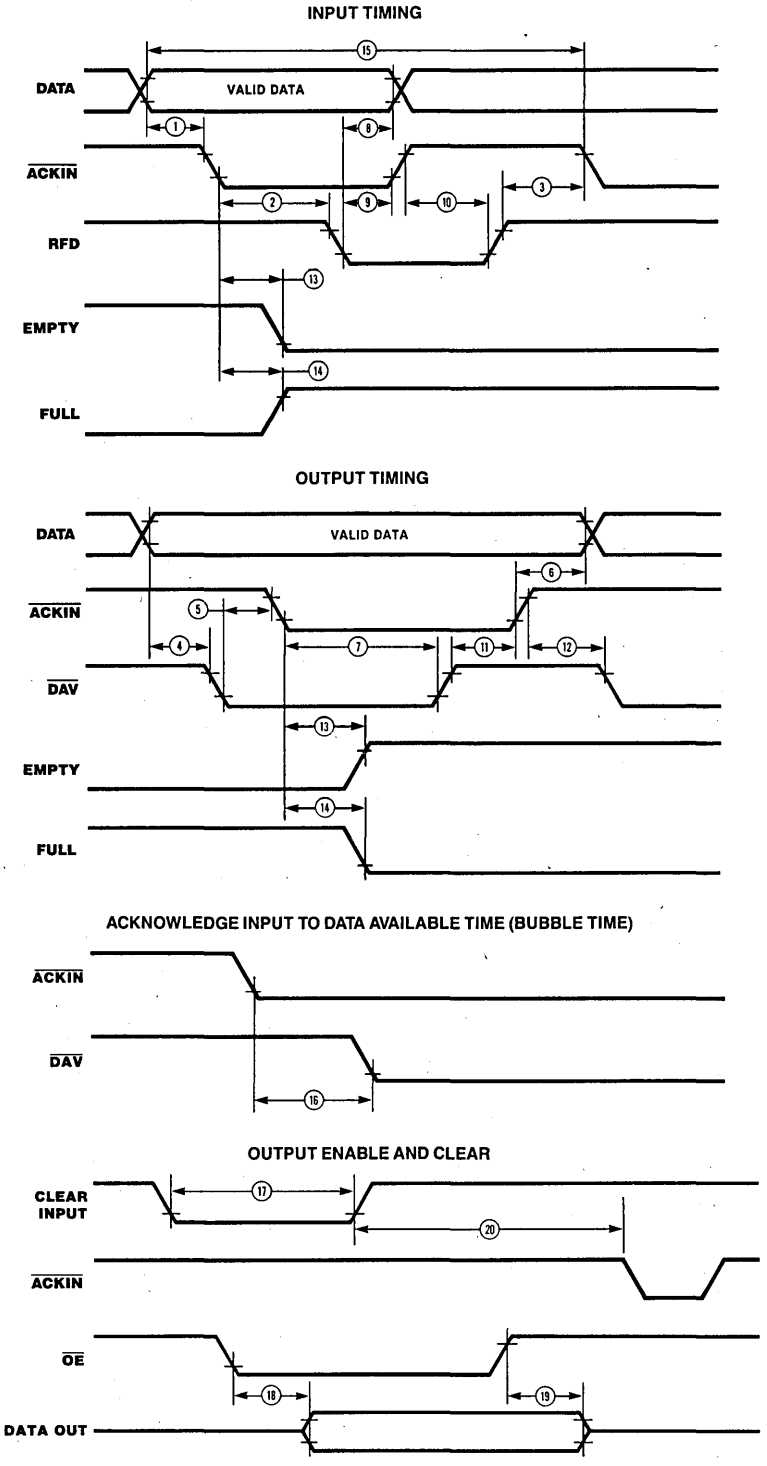


Figure 9. Timing Diagrams

FIFO 2-Wire Handshake Timing. Timing for 2-wire interlocked handshake operation is shown in Figure 9. The symbol, description and values for the numbered parameters (Figure 9) are given in AC Characteristics.

AC CHARACTERISTICS

Number	Symbol	Parameter	Min	Max	Units*
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ to Setup ime	50		ns
2	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay		500	ns
3	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		ns
4	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	50		ns
5	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		ns
6	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↑ Hold Time	50		ns
7	TdACK(DAV)	$\overline{ACKIN}$ ↓ to DAV ↑ Delay		500	ns
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		ns
9	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay	0		ns
10	TdACKr(RFD)	$\overline{ACKIN}$ ↑ to RFD ↑ Delay		400	ns
11	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑	0		ns
12	TdACKr(DAV)	$\overline{ACKIN}$ ↑ to $\overline{DAV}$ ↓		800	ns
13	TdACKINf(EMPTY)	(Input) $\overline{ACKIN}$ ↓ to EMPTY ↓ Delay (Output) $\overline{ACKIN}$ ↓ to EMPTY ↑ Delay		600	ns
14	TdACKINf(FULL)	(Input) $\overline{ACKIN}$ ↓ to FULL ↑ Delay (Output) $\overline{ACKIN}$ ↓ to FULL ↓ Delay		600	ns
15	ACKIN Clock Rate	(Input or Output)		1.0	MHz
16	TdACKINf(DAVf)	(Bubble Time)		1000	ns
17	TwCLR	Width of Clear to Reset FIFO	700		ns
18	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven		210	ns
19	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float		150	ns
20	TdCLR(ACK)	$\overline{CLEAR}$ ↑ to $\overline{ACKIN}$ ↓		800	ns

*All timing references assume 2.0V for a logic 1 and 0.8V for a logic 0. Timings are preliminary and subject to change.

Z8068/Z9518 Z-DCP
Data Ciphering Processor

October 1988

Features

- Encrypts and decrypts data using the National Bureau of Standards encryption algorithm.
- Supports three standard ciphering modes: Electronic Code Book, Chain Block and Cipher Feedback.
- Three separate registers for encryption, decryption, and master keys improve system security and throughput by eliminating frequent reloading of keys.
- Three separate programmable ports (master, slave, and key data) provide hardware separation of encrypted data, clear data, and keys.
- Data rates greater than 1M bytes per second can be handled.
- Key parity check.

General Description

The Z8068/Z9518 Data Ciphering Processors (DCP) are n-channel, silicon-gate LSI devices, which contains the circuitry to encrypt and decrypt daya using national Bureau of Standards encryption algorithms. It is designed to be used in a variety of environments, including dedicated controllers, communication concentrators, terminals, and peripheral task processors in general processor systems.

The DCP provides a high throughput rate using Cipher Feedback, Electronic Code Book, or Cipher Block Chain operating moces. The provisions of separate ports for key input, clear data, and enciphered data enhances security.

The host system communicates with the DCP using commands entered in the master port or through auxiiliary control lines. Once set up, data can flow through the DCP at high speeds because input, output and ciphering activities can be performed concurrently. External DMA control can easily be used to enhance throughput in some system configurations.

The Z8068 and Z9518 DCP are designed to interface directly to Zilog's Z-BUS® . Device signal/pin functions are shown in Figure 1; actual pin number assignments are shown in Figure 2.

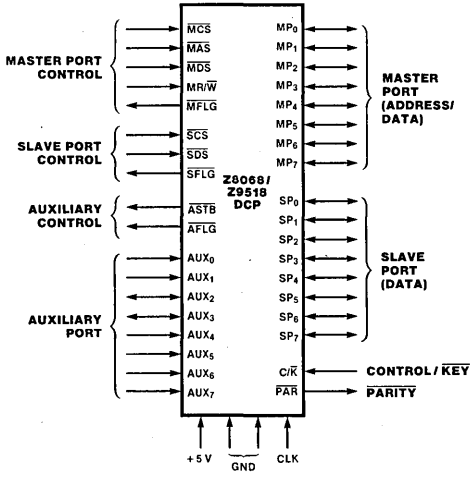
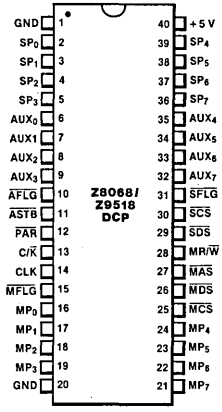


Figure 1. Pin Functions



(DIP) Pin Assignments

Pin Descriptions

$\overline{AFLG}$. *Auxiliary Port Flag* (output, active Low). This output signal indicates that the DCP is expecting key data to be entered on pins AUX_0-AUX_7. This can occur only when C/$\overline{K}$ is Low and a "Load Key Through AUX Port" command has been entered. $\overline{AFLG}$ remains active (Low) during the input of all eight bytes and will go inactive with the leading edge of the eighth strobe ($\overline{ASTB}$).

$\overline{ASTB}$. *Auxiliary Port Strobe* (input, active Low). In Multiplexed Control mode (C/$\overline{K}$ Low), the rising (trailing) edge of $\overline{ASTB}$ strobes the key data on pins AUX_0-AUX_7 into the appropriate internal key register. This input is ignored unless $\overline{AFLG}$ and C/$\overline{K}$ are both Low. One byte of key data is entered on each $\overline{ASTB}$ with the most significant byte entered first.

AUX_0-AUX_7. *Auxiliary Port Bus* (bidirectional, active High). When the DCP is operated in Multiplexed Control mode (C/$\overline{K}$ Low), these eight lines form a key-byte input port, which can be used to enter the master and session keys. This port is the only path available for entering the master key. (Session keys can also be entered via the master port.) AUX_0 is the low-order bit and is considered to be the parity bit in key bytes. The most significant byte is entered first.

When the DCP is operated in Direct Control mode (C/$\overline{K}$ High), the auxiliary port's key-entry function is disabled and five of the eight lines become direct control/status lines for interfacing to high-speed microprogrammed controllers. In this case, AUX_0, AUX_1 and AUX_4 have no function, and the other pins are defined as follows:

AUX_2-$\overline{BSY}$. *Busy* (output, active Low). This status output gives a hardware indication that the ciphering algorithm is in operation. AUX_2-$\overline{BSY}$ is driven by the BSY bit in the Status register such that when the BSY bit is 1 (active), AUX_2-$\overline{BSY}$ is Low.

AUX_3-$\overline{CP}$. *Command Pending* (output, active Low). This status output gives a hardware indication that the DCP is ready to accept the input of key bytes following a Low-to-High transition on AUX_7-K/$\overline{D}$. AUX_3-$\overline{CP}$ is driven by the CP bit in the Status register such that when the CP bit is 1 (active), AUX_3-$\overline{CP}$ is Low.

AUX_5-S/$\overline{S}$. *Start/Stop* (input, Low = Stop). When this pin goes Low (Stop), the DCP follows the normal Stop command sequence. When this pin goes High, a sequence equivalent to a Start Encryption or Start Decryption command is followed. When AUX_5-S/$\overline{S}$ goes High, the level on AUX_6-E/$\overline{D}$ selects either the start encryption or start decryption operation.

AUX_6-E/$\overline{D}$. *Encrypt/Decrypt* (input, Low = Decrypt). When AUX_5-S/$\overline{S}$ goes High, it initiates a normal data ciphering operation whose input specifies whether the ciphering algorithm is to encrypt (E/$\overline{D}$ High) or decrypt (E/$\overline{D}$ Low).

When AUX_7-K/$\overline{D}$ goes High, initiating the entry of key bytes, the level on AUX_6-E/$\overline{D}$ specifies whether the bytes are to be written into the E Key register (E/$\overline{D}$ High) or the D key Register (E/$\overline{D}$ Low).

The AUX_6-E/$\overline{D}$ input is not latched internally and must be held constant whenever one or more of AUX_5-S/$\overline{S}$, AUX_7-K/$\overline{D}$, AUX_2-$\overline{BSY}$, or AUX_3-$\overline{CP}$ are active. Failure to maintain the proper level on AUX_6-E/$\overline{D}$ during loading or ciphering operations results in scrambled data in the internal registers.

AUX_7-K/$\overline{D}$. *Key/Data* (input, Low = Data). When this signal goes High, the DCP initiates a key-data input sequence as if a Load Clear E or D Key Through Master Port command had been entered. The level on AUX_6-E/$\overline{D}$ determines whether the subsequently entered clear-key bytes are written into the E key register (E/D High) or the D key register (E/$\overline{D}$ Low)

AUX_7-K/$\overline{D}$ and AUX_5-S/$\overline{S}$ are mutually exclusive control lines; when one goes active (High), the other must remain inactive (Low) until the first returns to an inactive state. In addition, both lines must be inactive (Low) whenever a transition occurs on C/$\overline{K}$ (entering or exiting Direct Control mode).

C/$\overline{K}$. *Control/Key Mode Control.* (input, Low = Key). This input determines the operating characteristics of the DCP. A Low input on C/$\overline{K}$ puts the DCP into the Multiplexed Control mode, enabling programmed access to internal registers through the master port and enabling input of keys through the master or auxiliary port. A High input on C/$\overline{K}$ specifies operation in Direct Control mode. In this mode, several of the auxiliary port pins become direct control status signals which can be driven/sensed by high-speed controller logic, and access to internal registers through the master port is limited to the Input or Output register.

CLK. *Clock* (input, TTL compatible). An external timing source is input via the CLK pin. The Data Strobe signals ($\overline{MDS}$, $\overline{SDS}$) must change synchronously with this clock input, as must Master Port Address Strobe ($\overline{MAS}$) in Multiplexed Control mode (C/$\overline{K}$ Low), and also AUX_7-K/$\overline{D}$ and AUX_5-S/$\overline{S}$ in Direct Control mode (C/$\overline{K}$ High). In addition, the Auxiliary, Master and Slave Port Flag outputs ($\overline{AFLG}$, $\overline{MFLG}$, and $\overline{SFLG}$) change synchronously with the clock. When using the DCP with the Z8000 CPU in Multiplexed Control mode, the clock input must agree in frequency and phase with the processor clock; however, the DCP does not require the high voltage levels of the processor clock.

Pin Descriptions (Continued)

$\overline{\text{MAS}}$. *Master Port Address Strobe* (input, active Low). In Multiplexed Control mode (C/$\overline{\text{K}}$ Low), an active (Low) signal on this pin indicates the presence of valid address and chip select information at the master port. This information is latched internally on the rising edge of Master Port Address Strobe ($\overline{\text{MAS}}$). When C/$\overline{\text{K}}$ is High (Direct Control mode), $\overline{\text{MAS}}$ can be High or Low without affecting DCP operation, except that, regardless of the state of C/$\overline{\text{K}}$, if both Master Port Address Strobe ($\overline{\text{MAS}}$) and Data Strobe ($\overline{\text{MDS}}$) are Low simultaneously, the DCP Mode register will be reset to ECB mode. The master port is assigned to clear data, the slave port is assigned to enable data, and all flags remain inactive.

$\overline{\text{MCS}}$. *Master Port Chip Select* (input, active High). This signal is used to select the master port. In Multiplexed Control mode (C/$\overline{\text{K}}$ Low), the level on $\overline{\text{MCS}}$ is latched internally on the rising edge of Master Port Address Strobe ($\overline{\text{MAS}}$). This latched level is retained as long as $\overline{\text{MAS}}$ is High; when $\overline{\text{MAS}}$ is Low, the latch becomes invisible and the internal signal follows the $\overline{\text{MCS}}$ input. In Direct Control mode (C/$\overline{\text{K}}$ High), no latching of Master Port Chip Select occurs; the level on $\overline{\text{MCS}}$ is passed directly to the internal select circuitry, regardless of the state of Address Strobe ($\overline{\text{MAS}}$).

$\overline{\text{MDS}}$. *Master Port Data Strobe* (input, active Low). When $\overline{\text{MDS}}$ is active and Master Port Chip Select ($\overline{\text{MCS}}$) is valid, it indicates that valid data is present on MP_0-MP_7 during output. $\overline{\text{MDS}}$ and Master Port Address Strobe ($\overline{\text{MAS}}$) are normally mutually exclusive; if both go Low simultaneously, the DCP is reset to ECB mode and all flags remain inactive.

$\overline{\text{MFLG}}$. *Master Port Flag* (output, active Low). This flag is used to indicate the need for a data transfer into or out of the master port during normal ciphering operation. Depending upon the control bits written to the Mode register, the master port is associated with either the Input register or the Output register.

If data is to be transferred through the master port to the Input register, the $\overline{\text{MFLG}}$ reflects the contents of the Input register; after any start command is entered, $\overline{\text{MFLG}}$ goes active (Low) whenever the Input register is not full. $\overline{\text{MFLG}}$ is forced High by any command other than a start. Conversely, if the master port is associated with the Output register, $\overline{\text{MFLG}}$ reflects the contents of the Output register (except in single-port configuration). $\overline{\text{MFLG}}$ goes active (Low) whenever the Output register is not empty. In single-port configuration, $\overline{\text{MFLG}}$ reflects the contents of the Input register, while the Slave Port Flag ($\overline{\text{SFLG}}$) is associated with the Output register.

MP_0-MP_7. *Master Port Bus* (input/output, active High). These eight bidirectional lines are used to specify internal register addresses in Multiplexed Control mode (see C/$\overline{\text{K}}$) and to input and output data. The master port provides software access to the Status, Command and Mode registers as well as the Input and Output registers. The 3-state master port outputs are enabled only when the master port is selected by Master Port Chip Select ($\overline{\text{MCS}}$) being Low, with Master Port Read/Write (MR/$\overline{\text{W}}$) High, and strobed by a Low on the Master Port Data Strobe ($\overline{\text{MDS}}$). MP_0 is the low-order bit. Data and key information is entered into this port with most significant byte input first.

MR/$\overline{\text{W}}$. *Master Port Read/Write* (input, Low = Write). This signal indicates to the DCP whether the current master port operation is a read (MR/$\overline{\text{W}}$ is High) or a write (MR/$\overline{\text{W}}$ is Low), thereby indicating whether data is to be transferred from or to an internal register. MR/$\overline{\text{W}}$ is not latched internally and must be held stable while Master Port Data Strobe ($\overline{\text{MDS}}$) is Low.

$\overline{\text{PAR}}$. *Parity* (output, active Low). The DCP checks all key bytes for correct (odd) parity as they are entered through either the master port (Multiplexed or Direct Control mode) or the auxiliary port (Multiplexed Control mode only). If any key byte contains even parity, the PAR bit in the Status register is set to 1 and PAR goes Low. The least significant bit of key bytes is the parity.

$\overline{\text{SCS}}$. *Slave Port Chip Select* (input, active Low). This signal is logically combined with Slave Port Data Strobe ($\overline{\text{SDS}}$) to facilitate slave port data transfers in a bus environment. $\overline{\text{SCS}}$ is not latched internally and can be permanently tied to Low without impairing slave port operation.

$\overline{\text{SDS}}$. *Slave Port Data Strobe* (input, active Low). When both $\overline{\text{SDS}}$ and $\overline{\text{SCS}}$ are Low, it indicates to the DCP either that valid data is on the SP_0-SP_7 lines for an input operation, or that data is to be driven onto the SP_0-SP_7 lines for output. The direction of data flow is determined by the control bits in the Mode register.

$\overline{\text{SFLG}}$. *Slave Port Flag* (output, active Low). This output indicates the status of either the Input register or the Output register, depending on the control bits in the Mode register. In single-port configuration, $\overline{\text{SFLG}}$ goes active during normal processing whenever the Output register is not empty. In dual-port configuration, $\overline{\text{SFLG}}$ reflects the content of whichever register is associated with the slave port. If the input register is assigned to the slave port, $\overline{\text{SFLG}}$ goes active whenever the Input register is not full, once any of the start commands has been entered; $\overline{\text{SFLG}}$ is forced

Pin Descriptions (Continued)

inactive if any other command is entered. If the slave port is assigned to the Output register, $\overline{SFLG}$ goes active whenever the Output register is not empty. In this case, $\overline{SFLG}$ goes inactive if any command is aborted.

SP_0-SP_7. *Slave Port Bus* (bidirectional). The slave port provides a second data input/output interface to the DCP, allowing overlapped input, output, and ciphering operations. The 3-state slave port outputs are driven only when Slave Port Chip Select ($\overline{SCS}$) and Slave Port Data Strobe ($\overline{SDS}$) are both Low, $\overline{SFLG}$ is 0, and the internal port control configuration allows output to the slave port. SP_0 is the low order bit. The most significant byte of data blocks is entered or retrieved through this port first.

Functional Description

The overall design of the DCP, as shown in Figure 3, is optimized to achieve high data throughput. Data bytes can be transferred through both the master and slave ports, and key bytes can be written through both the auxiliary and master ports. Three 8-bit buses (input, output and C bus) carry data and key bytes between the ports and the internal registers. Three 56-bit, write-only key registers are provided for the Master (M) Key, the Encryption (E) Key and the Decryption (D) Key. Parity checking is provided on incoming key bytes. Two 64-bit registers are provided for initializing vectors (IVE and IVD) that are required for chained (feedback) ciphering modes. Three 8-bit registers (Mode, Command and Status) are accessible through the master port.

Algorithm Processing. The algorithm processing unit of the DCP (Figure 3) is designed to encrypt and decrypt data according to the National Bureau of Standards' Data Encryption Standard (DES), as specified in Federal Information Processing Standards Publication 46. The DES specifies a method for encrypting 64-bit blocks of clear data ("plain text") into corresponding 64-bit blocks of "cipher text."

The DCP offers three ciphering methods, selected by the cipher type field of the Mode register: Electronic Code Book (ECB), Cipher Block Chain (CBC) and Cipher Feedback (CFB). These methods are implemented in accordance with Federal Information Processing Standards, Publication 46.

Electronic Code Book (ECB) is a straightforward implementation of the DES: 64 bits of clear data in, 64 bits of cipher text out, with no cryptographic dependence between blocks.

Cipher Block Chain (CBC) also operates on blocks of 64 bits, but it includes a feedback step which chains consecutive blocks so that repetitive data in the plain text (such as ASCII blanks) does not yield repetitive cipher text. CBC also provides an error extension characteristic which protects against fraudulent data insertions and deletions.

Cipher Feedback (CFB) is an additive stream cipher method in which the DES algorithm generates a pseudorandom binary stream, which is then exclusive-ORed with the clear data to form the cipher text. The cipher text is then fed back to form a portion of the next DES input block. The DCP implements 8-bit cipher feedback, with data input, output,

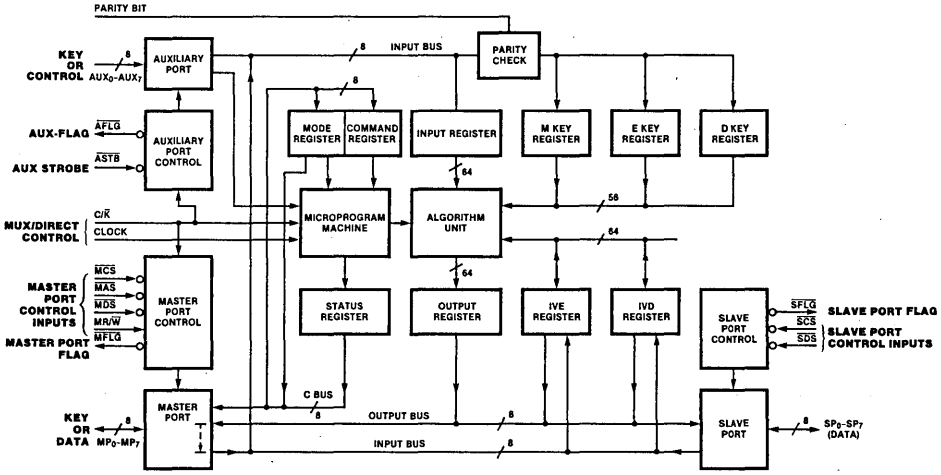


Figure 3. Z8068/Z9518 Block Diagram

Functional Description (Continued)

and feedback paths of one byte wide. This method is useful for low speed, character-at-a-time, serial communications.

Multiple Key Registers. The DCP provides the necessary registers to implement a multiple-key or master-key system. In such an arrangement, a single master key, stored in the DCP M key register, is used to encrypt session keys for transmission to remote DES equipment and to decrypt session keys received from such equipment. The M Key register may be loaded (with plain text) only through the auxiliary port, using the Load Clear Master Key command. In addition to the M Key register, the DCP contains two session key registers: the E key register, used to encrypt clear text, and the D key register, used to decrypt cipher text. All three registers are loaded by writing commands such as Load Clear E Key, through master port, into the Command register, and then writing the eight bytes of key data to the port when the Command Pending bit in the Status register is 1.

Operating Modes: Multiplexed Control vs. Direct Control. The DCP can be operated in either of two basic interfacing modes, determined by the logic level on the C/$\overline{K}$ input pin. In Multiplexed Control mode (C/$\overline{K}$ Low), the DCP is configured internally to allow a master CPU to address five of the internal control/status/data registers directly, thereby controlling the device via mode and command values written to these registers. Also, in this mode, the auxiliary port is enabled for key-byte input.

If the logic level on C/$\overline{K}$ is brought High, the DCP enters Direct Control mode, and the auxiliary port pins are converted into direct hardware status or control signals capable of instructing the DCP to perform a functionally complete subset of its cipher processing at very high throughputs. This operating mode is particularly well suited for ciphering data for high-speed peripheral devices such as magnetic disk or tape.

Data Flow. Bits M_2 and M_3 of the Mode register control the flow of data into and out of the DCP through the master and slave ports. Three basic configurations are provided: one single-port and two dual-port.

Single-Port Configuration. The simplest configuration occurs when the Mode register configuration bits are set to master port only (Figure 4). In this operating configuration, the encrypt/decrypt bit (M_4) controls the processing of data. Data to be encrypted or decrypted is written to the master port Input register address. To facilitate monitoring of the Input register status, the $\overline{MFLG}$ signal goes Low when the Input register is not full. Data is read by the master CPU through the master port Output register address. Pin $\overline{SFLG}$ goes Low when the Output register is not empty. $\overline{MFLG}$ is then redefined as a master input flag and $\overline{SFLG}$ is redefined as a master output flag.

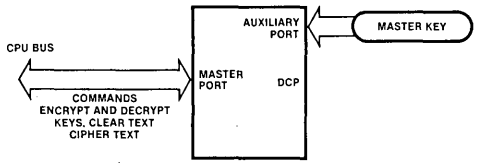


Figure 4. Single-Port Configuration, Multiplexed Control

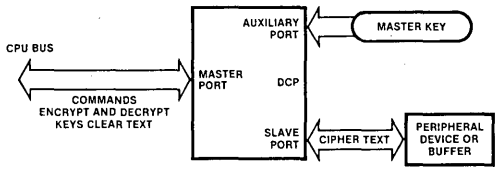


Figure 5a. Dual-Port Configuration, Multiplexed Control

Dual Port, Master Port Clear Configuration. In the dual-port configurations, both the master and slave ports are used for data entry and removal (Figures 5a and 5b). In the master port clear configuration, clear text for encryption can be entered only through the master port, and clear text resulting from decryption can be read only through the master port. Cipher text can be handled only through the slave port. The actual direction of data flow is controlled either by the encrypt/decrypt bit (M_4) in the Mode register or by the Start Encryption or Start Decryption commands. If encryption is specified, clear data will flow through the master port to the Input register, and cipher data will be available at the slave port when it is ready to be read from the Output register. For decryption, the process is reversed, with cipher data written to the Input register

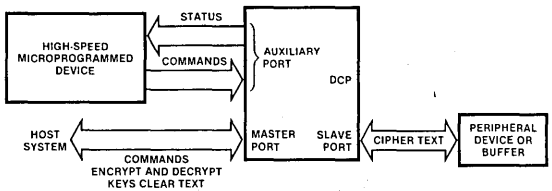


Figure 5b. Dual-Port Configuration, Direct Control

Functional Description (Continued)

through the master port. Slave port and clear text read from the Master port.

In both dual-port configurations, the Master Port Flag ($\overline{MFLG}$) and the Slave Port Flag ($\overline{SFLG}$) are used to indicate the status of the data register associated with the master port and slave port, respectively. For example, during encryption in the master port clear configuration, $\overline{MFLG}$ goes Low (active) when the Input register is not full; $\overline{SFLG}$ goes Low (active) when the Output register is not empty. If cyphering operation changes direction, $\overline{MFLG}$ and $\overline{SFLG}$ switch their register association (see Table 1).

Mode Register Bits				
Encrypt/ Decrypt Bit M4	Port Configuration Bit M3	Bit M2	Input Register Flag	Output Register Flag
0	0	0	$\overline{MFLG}$	$\overline{SFLG}$
0	0	1	$\overline{SFLG}$	$\overline{MFLG}$
0	1	0	$\overline{MFLG}$	$\overline{SFLG}$
1	0	0	$\overline{SFLG}$	$\overline{MFLG}$
1	0	1	$\overline{MFLG}$	$\overline{SFLG}$
1	1	0	$\overline{MFLG}$	$\overline{SFLG}$

Table 1. Association of Master Port Flag (MFLG) and Slave Port Flag (SFLG) with Input and Output Registers

Dual Port, Slave Port Clear Configuration. This configuration is identical to the previously described dual-port, master port clear configuration except that the direction of ciphering is reversed. That is, all data flowing in or out of the master port is cipher text, and all data at the slave port is clear text.

Master Port Read/Write Timing. The master port of the DCP is designed to operate directly with a multiplexed address/data bus such as the Zilog Z-BUS. Several features of the master port logic are:

- The level on Master Port Chip Select ($\overline{MCS}$) is latched internally on the rising (trailing) edge of Master Port Address Strobe ($\overline{MAS}$). This action relieves external address decode circuitry of the responsibility for latching chip select at address time.
- The levels on MP_1 and MP_2 are also latched internally on the rising edge of $\overline{MAS}$ and are subsequently decoded to enable reading and writing of the DCP's internal registers (Mode, Command, Status, Input and Output). This action also eliminates the need for external address latching and decoding.
- Data transfers through the master port are controlled by the levels and transitions on Master Port Data Strobe ($\overline{MDS}$) and Master Port Read/Write (MR/$\overline{W}$). The former controls the timing and the latter controls the transfer direction. Data transfers disturb neither the chip-select nor address latches, so once the DCP and a particular register have been selected, any number of reads or writes of that register can be accomplished without intervening address cycles. This feature greatly speeds up the loading of keys and data, given the necessary transfer control external to the DCP.

Loading Keys and Initializing Vector (IV) Registers. Because the key and Initializing Vector (IV) registers are not directly addressable through any of the DCP's ports, keys and vector data must be loaded (and in the case of vectors, read) via "command data sequences." Most of the commands recognized by the DCP are of this type. A load or read command is written to the Command register through the master port. The command processor responds by asserting the Command Pending output. The user then either writes eight bytes of key or vector data through the master or auxiliary port, as appropriate to the specific command, or reads eight bytes of vector data from the master port.

In Direct Control mode, only the E Key and D Key registers can be loaded; the M Key and IV registers are inaccessible. Loading the E and D Key registers is accomplished by placing the proper state on the AUX_6-E/$\overline{D}$ input (High for E Key, Low for D Key) and then raising the AUX_7-K/$\overline{D}$ input—indicating that key loading is required. The command processor attaches the proper key register to the master port and asserts the AUX_3-$\overline{CP}$ (Command Pending) signal (active Low). The eight key bytes can then be written to the master port. In the Multiplexed Control mode, all key and vector registers can be written to and all but the Master (M) Key register can be loaded with encrypted, as well as clear, data. If the operation is a Load Encrypt command, the subsequent data written to the master or auxiliary port (as appropriate) is routed first to the Input register and decrypted before it is written into the specified key or Initializing Vector register.

Parity Checking of Keys. Key bytes contain seven bits of key information and one parity bit. By DES designation, the low-order bit is the parity bit. The parity-check circuit is enabled whenever a byte is written to one of three key registers. The output of the parity-check circuit is connected to $\overline{PAR}$ and the state of this signal is reflected in Status register bit PAR (S_3). Status register bit PAR goes to 1 whenever a byte with even parity (an even number of 1s) is detected. In addition to the PAR bit, the Status register has a Latched Parity bit (LPAR, S_4) that is set to 1 whenever the Status register PAR bit goes to 1. Once set, the LPAR bit is not cleared until a reset occurs or a new Load Key command is issued.

Functional Description (Continued)

When an encrypted key is entered, the parity-check logic operates only after the decrypted key is available. The encrypted data is not checked for parity. The $\overline{PAR}$ signal reflects the state of the decrypted bytes on a byte-to-byte basis as they are clocked through the parity-check logic on their way to the key register. Thus, the time during which $\overline{PAR}$ indicates the status of a byte of decrypted key data may be as short as four clock cycles. The LPAR bit in the Status register indicates if any erroneous bytes of key data were entered.

Programming

Initialization. The DCP can be reset in several ways:

- By the "Software Reset" command.
- By a hardware reset, which occurs whenever both $\overline{MAS}$ and $\overline{MDS}$ go Low simultaneously.
- By writing to the Mode register.
- By aborting any command.

These sequences initiate the same internal operations, except that loading the Mode register or aborting any command does not subsequently reset the Mode register. Once a reset process starts, the DCP is unable to respond to further commands for approximately five clock cycles. If a power-up hardware reset is used, the leading edge of the reset signal should not occur until approximately 1 ms after V_{CC} has reached normal operating voltage. This delay time is needed for internal signals to stabilize.

Registers. The registers in the DCP that can be addressed directly through the master port are shown with their addresses in Table 2. A brief description of these registers and those not directly accessible follows.

C/$\overline{K}$	MP2	MP1	MR/$\overline{W}$	$\overline{MCS}$	Register Addressed
0	X	0	0	0	Input Register
0	X	0	1	0	Output Register
0	0	1	0	0	Command Register
0	0	1	1	0	Status Register
0	1	1	X	0	Mode Register
X	X	X	X	1	No Register Accessed
1	X	X	0	0	Input Register
1	X	X	1	0	Output Register

Table 2. Master Port Register Addresses

Hex Code	Command
90	Load Clear M Key Through Auxiliary Port
91	Load Clear E Key Through Auxiliary Port
92	Load Clear D Key Through Auxiliary Port
11	Load Clear E Key Through Master Port
12	Load Clear D Key Through Master Port
B1	Load Encrypted E Key Through Auxiliary Port
B2	Load Encrypted D Key Through Auxiliary Port
31	Load Encrypted E Key Through Master Port
32	Load Encrypted D Key Through Master Port
85	Load Clear IVE Through Master Port
84	Load Clear IVD Through Master Port
A5	Load Encrypted IVE Through Master Port
A4	Load Encrypted IVD Through Master Port
8D	Read Clear IVE Through Master Port
8C	Read Clear IVD Through Master Port
A9	Read Encrypted IVE Through Master Port
A8	Read Encrypted IVD Through Master Port
39	Encrypt With Master Key
41	Start Encryption
40	Start Decryption
C0	Start
E0	Stop
00	Software Reset

Table 3. Command Codes in Multiplexed Control Mode

Command Register. Data written to the 8-bit, write-only Command register through the master port is interpreted as an instruction. A detailed description of each command is given in the Commands section; the commands and their hexadecimal representations are summarized in Table 3. A subset of these commands can be entered implicitly in Direct Control mode (C/$\overline{K}$ High)—even though the Command register cannot be addressed in that mode—by transitions on auxiliary lines AUX_5-S/$\overline{S}$, AUX_6-E/$\overline{D}$, and AUX_7-K/$\overline{D}$. These implicit commands are summarized in Table 4.

	Pins			
C/$\overline{K}$	AUX_7-K/$\overline{D}$	AUX_6-E/$\overline{D}$	AUX_5-S/$\overline{S}$	Command Initiated
H	L	L	↑	Start Decryption
H	L	H	↑	Start Encryption
H	L	X	↓	Stop
H	↑	L	L	Load D Key Clear through master port
H	↑	H	L	Load E Key Clear through master port
H	↓	X	L	End Load Key command
H	H	X	H	Not allowed
L	Data	Data	Data	AUX pins become Key-Byte inputs

Table 4. Implicit Command Sequences in Direct Control Mode

Programming (Continued)

Status Register. The bit assignments in the read-only Status register are shown in Figure 6. The PAR, AFLG, SFLG and MFLG bits indicate the status of the corresponding output pins, as do the busy and command pending bits when the DCP is in a Direct Control mode (C/$\overline{K}$ High). In each case, the output signal will be active Low when the corresponding status bit is a 1. The parity bit indicates the parity of the most recently entered key byte. The LPAR bit indicates whether any key byte with even parity has been encountered since the last Reset or Load Key command.

The Busy bit is 1 whenever the ciphering algorithm unit is actively encrypting or decrypting data, either as a response to a command such as Load Encrypted Key (in which case the Command Pending bit is 1) or in the ciphering of regular text (indicated by the Start/Stop bit being 1). If the ciphered data cannot be transferred to the Output register because that register still contains output from a previous ciphering cycle, the Busy bit remains 1 even after the ciphering is complete. Busy is 0 at all other times, even when ciphering is not possible because data has not been written to the Input register.

The Command Pending bit is set to 1 by any command whose execution requires the transfer of data to or from a nonaddressable internal register, such as when writing key bytes to the E key register or reading bytes from the IVE register. Thus, the Command Pending bit is set following all commands except the three start commands, the Stop command and the Software Reset command. The Command Pending bit returns to 0 after all eight bytes have been transferred following Load Clear, Read Clear, or Read Encrypted commands; and after data has been transferred, decrypted, and loaded into the desired register following Load Encrypt commands.

The Start/Stop bit is set to 1 when one of the start commands is entered and it is reset to 0 whenever a reset occurs or when a new command other than a Start is entered.

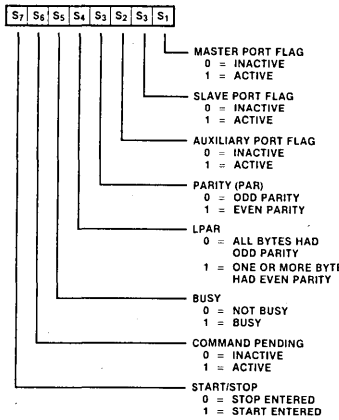


Figure 6. Status Register Bit Assignments

Mode Register. Bit assignments in this 5-bit read/write register are shown in Figure 7. The cipher type bits (M_1 and M_0) indicate to the DCP which ciphering algorithm is to be used. On reset, the Cipher Type mode defaults to Electronic Code Book mode.

Configuration bits (M_3 and M_2) indicate which data ports are to be associated with the Input and Output registers and flags. When these bits are set to the single-port, master-only configuration (M_3 M_2 = 10), the slave port is disabled and no manipulation of Slave Port Chip Select ($\overline{SCS}$) or Slave Data Strobe ($\overline{SDS}$) can result in data movement through the slave port; all data transfers are accomplished through the master port, as previously described in the Functional Description. Both $\overline{MFLG}$ and $\overline{SFLG}$ are used in this configuration; $\overline{MFLG}$ gives the status of the Input register and $\overline{SFLG}$ gives the status of the Output register.

When the configuration bits are set to one of the dual-port configurations (M_3 M_2 = 00 or 01), both the master and slave ports are available for input and output. When M_3, M_2 = 01 (the default configuration), the master port handles clear data while the slave port handles encrypted data. Configuration M_3, M_2 = 00 reverses this assignment. Actual data direction at any particular moment is controlled by the Encrypt/Decrypt bit.

The Encrypt/Decrypt bit (M_4) instructs the DCP algorithm processor to encrypt or decrypt the data from the Input register using the ciphering method specified by the Cipher Type bits. The Encrypt/Decrypt bit also controls data flow within the DCP. For example, when the configuration bits are 0,1 (dual-port, master clear, slave encrypted) and the Encrypt/Decrypt bit is 1 (encrypt), clear data will flow into the DCP through the master port and encrypted data will flow out through the slave port. When the Encrypt/Decrypt bit is set to 0 (decrypt), data flow is reversed.

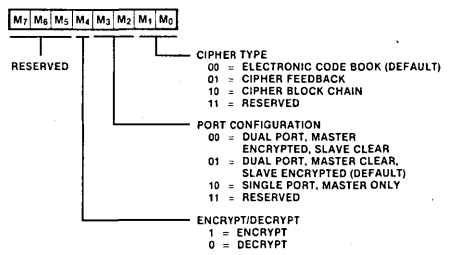


Figure 7. Mode Register Bit Assignments

Programming (Continued)

Input Register. The 64-bit, write-only Input register is organized to appear to the user as eight bytes of pushdown storage. A status circuit monitors the number of bytes that have been stored. The register is considered empty when the data stored in it has been or is being processed; it is considered full when one byte of data has been entered in Cipher Feedback mode or when eight bytes of data have been entered in Electronic Code Book or Cipher Block Chain mode. If the user attempts to write data into the Input register when it is full, the Input register disregards the attempt; no data in the register is destroyed.

Output Register. The 64-bit, read-only Output register is organized to appear to the user as eight bytes of pop-up storage. A status circuit detects the number of bytes stored in the Output register. The register is considered empty when all the data stored in it has been read by the master CPU and is considered full if it still contains one or more bytes of output data. If a user attempts to read data from the Output register when it is empty, the buffers driving the output bus remain in a 3-state condition.

M, E, D Key Registers. The following multibyte key registers cannot be addressed directly, but are loaded in response to commands written to the Command register.

There are three 64-bit, write-only key registers in the DCP: the Master (M) Key register, the Encrypt (E) key register, and the Decrypt (D) key register. The Master key register can be loaded only with clear data through the auxiliary port. The Encrypt and Decrypt Key registers can be loaded in any of four ways: (1) as clear data through the auxiliary port, (2) as clear data through the master port, (3) as encrypted data through the auxiliary port, or (4) as encrypted data through the master port. In the last two cases, the encrypted data is first routed to the Input register, decrypted using the M Key, and finally written to the target key register from the Output register.

Initializing Vector Registers (IVE and IVD). Two 64-bit registers are provided to store feedback values for cipher feedback and chained block ciphering methods. One initializing vector register (IVE) is used during encryption, the other (IVD) is used during decryption. Both registers can be loaded with either clear or encrypted data through the master port (in the latter case, the data is decrypted before being loaded into the IV register), and both may be read out either clear or encrypted through the master port.

Commands

All operations of the DCP result from command inputs, which are entered in Multiplexed Control mode by writing a command byte to the Command register. Command inputs are entered in Direct Control mode by raising and lowering the logic levels on the AUX_7-K/$\overline{D}$, AUX_6-E/$\overline{D}$, and AUX_5-S/$\overline{S}$ pins. Table 3 shows all commands that can be given in Multiplexed Control mode. Table 4 shows a subset of the implicit commands that can be executed in the Direct Control mode.

Load Clear M Key Through Auxiliary Port (90H).
Load Clear E Key Through Auxiliary Port (91H).
Load Clear D Key Through Auxiliary Port (92H).

These commands may be used only for multiplexed operations; they override the data flow specifications set in the Mode register and cause the Master (M) Key, Encrypt (E) Key, or Decrypt (D) Key register to be loaded with eight bytes written to the auxiliary port. After the Load command is written to the Command register, the Auxiliary Port Flag ($\overline{AFLG}$) goes active (Low) and the corresponding bit in the Status register (S_2) becomes 1, indicating that the device is able to accept key bytes at the auxiliary port pins. Additionally, the Command Pending bit (S_6) becomes 1 during the entire loading process.

Each byte is written to its respective key register by placing an active Low signal on the Auxiliary Port Strobe ($\overline{ASTB}$) once data has been set up on the auxiliary port pins. The actual write process occurs on the rising (trailing) edge of $\overline{ASTB}$. (See Switching Characteristics section for exact setup, strobe width, and hold times.)

The Auxiliary Port Flag ($\overline{AFLG}$) goes inactive immediately after the eighth strobe goes active (Low). However, the Command Pending bit (S_6) remains 1 for several more clock cycles, until the key loading process is completed. All key bytes are checked for correct (odd) parity as they are entered.

Load Clear E Key Through Master Port (11H).
Load Clear D Key Through Master Port (12H).

These commands are available in both Multiplexed Control and Direct Control modes. They override the data flow specifications set in the Mode register and attach the master port inputs to the Encrypt (E) Key or Decrypt (D) Key register, as appropriate, until eight key bytes have been written. In Multiplexed Control mode, the command is initiated by writing the Load command to the Command register. In Direct Control mode, the command is initiated by raising the AUX_7-K/$\overline{D}$ control input while the AUX_5-S/$\overline{S}$

Commands (Continued)

input is Low. In this latter case, the level on AUX_6-E/$\overline{D}$ determines which key register is written (High = E register).

Once the command has been recognized, the Command Pending bit (S_6 in the Status register) becomes 1. In Direct Control mode, AUX_3-$\overline{CP}$ goes active (Low), indicating that key entry may proceed. The host system then writes exactly eight bytes to the master port (at the Input register address in Multiplexed Control mode). When the key register has been loaded, the Command Pending bit returns to 0. In Direct Control mode, the AUX_3-$\overline{CP}$ output goes inactive, indicating that the DCP can accept the next command.

Load Encrypted E Key Through Auxiliary Port (B1H).
Load Encrypted D Key Through Auxiliary Port (B2H).

These commands are used in Multiplexed Control mode only. Their execution is similar to that of the Load Clear E (D) Key Through Auxiliary Port command, except that key bytes are first decrypted using the electronic code book algorithm and the Master (M) Key register. The key bytes are then loaded into the appropriate key register, after having passed through the parity-check logic.

The Command Pending bit (S_6) is 1 during the entire decrypt-and-load operation. In addition, the Busy bit (S_5) is 1 during the actual decryption process.

Load Encrypted E Key Through Master Port (31H).
Load Encrypted D Key Through Master Port (32H).

These commands are used in Multiplexed Control mode only. Their execution is similar in effect to that of the Load Clear E (D) Key Through Master Port command. The commands differ in that key bytes are initially decrypted using the electronic code book algorithm and the Master (M) Key register. Once decrypted, they are loaded byte-by-byte into the target key register, after having passed through the parity-check logic.

The command pending bit (S_6) is 1 during the entire decrypt-and-load operation. In addition, the busy bit (S_5) is 1 during the actual decryption process.

Load Clear IVE Register Through Master Port (85H)
Load Clear IVD Register Through Master Port (84H)

These commands are used in Multiplexed Control mode only. Their execution is virtually identical to that of the Load Clear E (or D) Key Through Master Port command. The commands differ in that the data written to the input register address is routed to either the Encryption Initializing Vector (IVE) or Decryption Initializing Vector (IVD) register instead of a key register. No parity checking occurs. The Command Pending bit (S_6) is 1 during the entire loading process.

Load Encrypted IVE Register Through Master Port (A5H).
Load Encrypted IVD Register Through Master Port (A4H).

These commands are analogous to the Load Encrypted E (or D) Key Through Master Port command. The data flow specifications set in the Mode register are overridden and the eight vector bytes are decrypted using the Decryption (D) Key register and the electronic code book algorithm. The resulting clear vector bytes are loaded into the target Initializing Vector register. No parity checking occurs. The Busy bit (S_5) does not become 1 during the decryption process, but the Command Pending bit (S_6) is 1 during the entire decryption-and-load operation.

Read Clear IVE Register Through Master Port (8DH).
Read Clear IVD Register Through Master Port (8CH).

In the Multiplexed Control mode, these commands override the data flow specifications set in the Mode register and connect the appropriate Initializing Vector register to the master port at the Output register address. In this state, each IV register appears as eight bytes of FIFO storage. The first byte of data is available six clocks after loading the Command register. The Command Pending bit in the Status register remains a 1 until sometime after the eighth byte is read out. The host system is responsible for reading exactly eight bytes.

Read Encrypted IVE Register Through Master Port (A9H).
Read Encrypted IVD Register Through Master Port (A8H).

In the Multiplexed Control mode only, these commands override the specifications set in the Mode register and encrypt the contents of the specified Initializing Vector register using the electronic code book algorithm and the Encrypt (E) key. The resulting cipher text is placed in the output register, where it can be read as eight bytes through the master port. During the actual encryption process, the Busy bit (S_5) is 1. When the Busy bit becomes 0, the encrypted vector bytes are ready to be read out. The Command Pending bit (S_6) is 1 during the entire encryption and output process; it becomes 0 when the eighth byte is read out. The host system is responsible for reading exactly eight bytes.

Encrypt with Master (M) Key (39H).

In the Multiplexed Control mode, this command overrides the data flow specifications set in the Mode register and causes the DCP to accept eight bytes from the master port, which are written to the Input register. When eight bytes have been received, the DCP encrypts

Commands (Continued)

the input using the Master (M) Key register. The encrypted data is loaded into the Output register, where it can be read out through the master port. The Command Pending bit (S_6) and the Busy (S_5) bit are used as status indicators in the three phases of this operation.

The Command Pending bit becomes 1 as soon as the Input register can accept data. When exactly eight bytes have been entered, the Busy bit becomes and remains 1 until the encryption process is complete. When Busy becomes 0, the encrypted data is available to be read out. The Command Pending bit returns to 0 when the eighth byte has been read.

Start Encryption (41H)
Start Decryption (40H)
Start (C0H).

The three start commands begin normal data ciphering by setting the Status register's Start/Stop bit (S_7) to 1. The Start Encryption and Start Decryption commands explicitly specify the ciphering direction by forcing the Encrypt or Decrypt bit (M_4) in the Mode register to 1 or 0, respectively. The Start command, however, uses the current state of the Encrypt/Decrypt bit, as specified in a previous Mode register load.

When a start command has been entered, the port status flag ($\overline{MFLG}$ or $\overline{SFLG}$) associated with the Input register becomes active (Low), indicating that data may be written to the Input register to begin ciphering.

In Direct Control mode, the Start command is issued by raising the level on the AUX_5-$S/\overline{S}$ input (Table 4). The ciphering direction is specified by the level on AUX_6-$E/\overline{D}$. If AUX_6-$E/\overline{D}$ is High when AUX_5-$S/\overline{S}$ goes High, the command is Start Encryption; if AUX_6-$E/\overline{D}$ is Low, it is Start Decryption.

Stop (E0H).

The Stop command clears the Start/Stop bit (S_7) in the Status register. This action causes the input flag ($\overline{MFLG}$ or $\overline{SFLG}$) to become inactive and inhibits the loading of any further input into the algorithm unit. If ciphering is in progress [Busy bit (S_5) is 1 or AUX_2-$\overline{BSY}$ is active], it is allowed to finish, and any data in the Output register remains accessible.

In Direct Control mode, the Stop command is implied when the signal level on the AUX_5-$S/\overline{S}$ input goes from High to Low (Table 4).

Software Reset (00).

This command has the same effect as a hardware reset ($\overline{MAS}$ and $\overline{MDS}$ Low): it forces the DCP back to its default configuration, and all processing flags go into Inactive mode. The default configuration includes setting the Mode register to Electronic Code Book ciphering mode and establishes a dual-port configuration with master port clear and slave port encrypted.

Timing Requirements

The control and/or data signals and the timing requirements for clock/reset, Direct Control mode, Multiplexed Control mode (master port), master (slave) port read/write, and auxiliary port key entry functions are illustrated in Figures 8 through 12. The ac switching characteristics of the signals involved in the above functions are described in the AC Characteristics. The specific timing periods described are identified by numerics (1 through 48), which are referenced in both the timing diagrams and in the AC Characteristics.

A two-to-seven character symbol is listed in AC Characteristics for each period described. The symbol specifies the signal(s) involved, the state of each signal, and optionally, the port associated with a signal. Symbols are encoded as follows:

General Form: Ta Ab (Cb)

Where:

(1) T is a constant.

(2) a represents any one of the following symbols:

Symbol	*Meaning*
c	Clock
d	Delay
f	Fall Time
h	Hold Time
r	Rise Time
s	Setup Time
w	Width

(3) A,C represent any of the following signal names:

Symbol	*Signal Name*
A	Address Strobe
B	$\overline{BSY}$, Busy
C	Clock
D*	Data In or the address at the master port.
E	$E/\overline{D}$, Enable/Disable
F*	Flag ($\overline{MFLG}$, $\overline{SFLG}$, or ($\overline{AFLG}$)
G*	Data Strobe ($\overline{MDS}$, $\overline{SDS}$, or $\overline{ASTB}$)
K	$K/\overline{D}$, Key/Data
M	$C/\overline{K}$, Control/Key Mode
N	$S/\overline{S}$, Start/Stop
P	$\overline{PAR}$, Parity
Q*	Data Out (master or slave port)
R	$\overline{CP}$, Clock Pulse
S*	Chip Select (master or slave port)
W	$MR/\overline{W}$, Master Port read/write

Timing Requirements (Continued)

(4) b represents any one of the following signal state descriptors (symbol).

Symbol	*State Indicated*
h	High
l	Low
v	Valid
x	Invalid
z	High Impedance

*These signal names may be modified by the following optional numeric port identifiers:

Identifier	*Port*
1	Master Port
2	Slave Port
3	AUX (Key) Port

For example: D1 specifies data in at Master Port; F2 specifies Slave Port flag-SFLG.

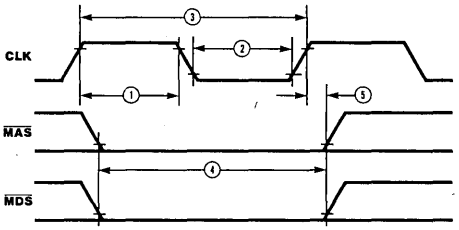


Figure 8. Clock and Reset

MAXIMUM RATINGS (Above which useful life may be impaired)

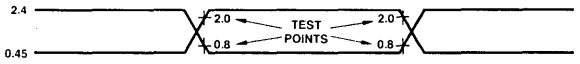
Storage Temperature	−65 to +150°C
Ambient Temperature Under Bias	0 to +70°C
Voltage on Any Pin with Respect to Ground	−0.5 to +7.0V
Power Dissipation	1.5W

The products described by this specification include internal circuitry designed to protect input devices from damaging accumulations of static charge. It is suggested nevertheless, that conventional precautions be observed during storage, handling and use in order to avoid exposure to excessive voltages.

Z8068, Z9518 ELECTRICAL CHARACTERISTICS (over operating range unless otherwise specified)

T_A = 0 to 70°C, V_{CC} = +5.0V ±5%, V_{SS} = 0V

Parameters	Description	Test Conditions	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		−0.5		.8	Volts
V_{IH}	Input High Voltage		2.2		V_{CC}	Volts
V_{OL}	Output Low Voltage	I_{OL} = 3.2mA			.40	Volts
V_{OH}	Output High Voltage	I_{OH} = −400μA	2.4			Volts
I_I	Input Leakage Current	$V_{SS} \leq V_{IN} \leq V_{CC}$			±10	μA
I_{OZ}	Output Leakage Current	V_{SS} + .40 ≤ $V_{IN} \leq V_{CC}$			±10	μA
I_{CC}	Supply Current (AVER.)			150	250	mA



INPUT WAVEFORMS FOR A.C. TESTS

Z8068/Z9518 SWITCHING CHARACTERISTICS (Note 1)

The table below specifies the guaranteed performance of this device over the commercial operating range of 0 to +70°C with V_{CC} from 4.75 to 5.25V. All data are in nanoseconds. Switching tests are made with inputs and outputs measured at 0.8V for a LOW and 2.0V for a HIGH. Outputs are fully loaded, with $C_L \geq$ 50pF. See switching waveform figures following table for graphic illustration of timing parameters.

SWITCHING CHARACTERISTICS over operating range

Parameter	Number	Description	Z8068 Min	Z8068 Typ	Z8068 Max	Z9518 Min	Z9518 Typ	Z9518 Max	Units
Clock									
TWH	1	Clock Width (HIGH)	115			150			ns
TWL	2	Clock Width (LOW)	115			150			ns
TC	3	Clock HIGH to Next Clock HIGH (Clock Cycle)	250		1000	320		1000	ns
Reset									
TG1LG1H	5	$\overline{MDS} \cdot \overline{MAS}$ LOW to $\overline{MDS} \cdot \overline{MAS}$ HIGH (Reset Pulse Width)	TC			TC			ns
TCHG1H	6	Clock HIGH to $\overline{MDS} \cdot \overline{MAS}$ HIGH	0		50	0		50	ns
Direct Control Mode									
TNLMH	9	S/$\overline{S}$ LOW to C/$\overline{K}$ HIGH (Setup)	3TC			3TC			ns
TKLMH	10	K/$\overline{D}$ LOW to C/$\overline{K}$ HIGH (Setup)	3TC			3TC			ns
TMHNH	11	C/$\overline{K}$ HIGH to S/$\overline{S}$ HIGH	6TC			6TC			ns
TMHKH	12	C/$\overline{K}$ HIGH TO K/$\overline{D}$ HIGH	6TC			6TC			ns
TEVKH	14	E/$\overline{D}$ VALID to K/$\overline{D}$ HIGH (Setup)	3TC			3TC			ns
TKHRL	15	K/$\overline{D}$ HIGH to $\overline{CP}$ LOW			300			300	ns
TKLEX	17	K/$\overline{D}$ LOW to E/$\overline{D}$ INVALID (Hold)	TC			TC			ns
TCLNV	19	Clock LOW to S/$\overline{S}$ VALID	20		80	20		80	ns
TEVNH	20	E/$\overline{D}$ VALID to S/$\overline{S}$ HIGH (Setup)	3TC			3TC			ns
TNHF1L	21	S/$\overline{S}$ HIGH to $\overline{MFLG}$ ($\overline{SFLG}$) LOW (Port Input Flag)			230			300	ns
TCHF1L	22	Clock HIGH to $\overline{MFLG}$ ($\overline{SFLG}$) LOW (Port Input Flag) (Note 2)			230			300	ns
TCHBL	24	Clock HIGH to $\overline{BSY}$ LOW			300			400	ns
TCLBH	25	Clock LOW to $\overline{BSY}$ HIGH			230			300	ns
TCHF1L	27	Clock HIGH to $\overline{MFLG}$ ($\overline{SFLG}$) LOW (Port Output Flag)			230			300	ns
TNLF1H	28	S/$\overline{S}$ LOW to $\overline{MFLG}$ ($\overline{SFLG}$) HIGH (Port Input Flag) (Note 3)			230			300	ns
Multiplexed Control Mode – Master Port									
TWA	32	$\overline{MAS}$ Width (LOW)	80			115			ns
TS1LAH	34	$\overline{MCS}$ LOW to $\overline{MAS}$ HIGH (Setup)	0			0			ns
TAHS1H	35	$\overline{MAS}$ HIGH to $\overline{MCS}$ HIGH (Hold)	60			60			ns
TD1VAH	36	Address-In VALID to $\overline{MAS}$ HIGH (Address Setup Time)	55			90			ns
TAHD1X	37	$\overline{MAS}$ HIGH to Address-In INVALID (Address Hold Time)	60			60			ns

AC SWITCHING CHARACTERISTICS

Parameter	Number	Description		Z8068 Min	Z8068 Typ	Z8068 Max	Z9518 Min	Z9518 Typ	Z9518 Max	Units
Master (Slave) Port Read/Write										
TS1LG1L	40	$\overline{MCS}$ ($\overline{SCS}$) LOW to $\overline{MDS}$ ($\overline{SDS}$) LOW (Select Setup) (Note 4)		100			100			ns
TG1HS1H	41	$\overline{MDS}$ ($\overline{SDS}$) HIGH to $\overline{MCS}$ ($\overline{SCS}$) HIGH (Select Hold Time) (Note 4)		25			25			ns
TWVG1L	42	MR/$\overline{W}$ VALID to $\overline{MDS}$ LOW (Setup)		100			100			ns
TG1HWX	43	$\overline{MDS}$ HIGH to MR/$\overline{W}$ INVALID (Hold)		25			25			ns
TG1LG1H	44	$\overline{MDS}$ ($\overline{SDS}$) LOW to $\overline{MDS}$ ($\overline{SDS}$) HIGH	Width – Write, Data Read	125		1000	160		1000	ns
			Width – Status Register Read	200		1000	300		1000	
TCLG1H	45	Clock LOW to $\overline{MDS}$ ($\overline{SDS}$) HIGH (Note 11)		0		TWL – 65	0		TWL – 100	
TGIHG1L	46	$\overline{MDS}$ ($\overline{SDS}$) HIGH to $\overline{MDS}$ ($\overline{SDS}$) LOW (Data Strobe Recovery Time)		125			160			ns
TD1VG1H	47	Write-Data VALID $\overline{MDS}$ ($\overline{SDS}$) HIGH	Setup Time – Key Load (Note 8)	125			160			ns
			Setup Time – Data Write	125			160			
			Setup Time – Command/ Mode Register Write	125			160			
TG1HD1X	48	$\overline{MDS}$ ($\overline{SDS}$) HIGH to Write-Data INVALID (Hold Time – All Writes)		25			25			ns
TG1LQ1V	49	$\overline{MDS}$ ($\overline{SDS}$) LOW to Read-Data VALID	Read Access Time – Status Register			200			300	ns
			Read Access Time – Data			120			150	
TG1HQ1V	50	$\overline{MDS}$ ($\overline{SDS}$) HIGH to Read-Data INVALID (Read Hold Time)		5			5			ns
TG1LF1H	51	$\overline{MDS}$ ($\overline{SDS}$) LOW to $\overline{MFLG}$ ($\overline{SFLG}$) HIGH (Last Strobe) (Note 5)				125			160	ns
TG1LRH	52	$\overline{MDS}$ HIGH to CP HIGH	Last Strobe, Key Load			TC+500			TC+500	ns
TG1HNL	53	$\overline{MDS}$ ($\overline{SDS}$) HIGH to S/$\overline{S}$ LOW (Hold Time) (Note 9)		4TC			4TC			ns
TG1HPV	54	$\overline{MDS}$ HIGH to $\overline{PAR}$ VALID (Key Write)				200			250	ns
Auxiliary Port Key Entry										
TG3LG3H	61	$\overline{ASTB}$ LOW to $\overline{ASTB}$ HIGH (Width)		160			160			ns
TCLG3H	62	Clock LOW to $\overline{ASTB}$ HIGH		0		50	0		50	ns
TG3HG3L	63	$\overline{ASTB}$ HIGH to Next $\overline{ASTB}$ LOW (Recovery Time)		250			320			ns
TD3VG3H	64	Write-Data VALID to $\overline{ASTB}$ HIGH (Data Setup Time)		200			300			ns
TG3HD3X	65	$\overline{ASTB}$ HIGH to Write-Data INVALID (Data Hold Time)		80			80			ns
TG3HPV	66	$\overline{ASTB}$ HIGH to $\overline{PAR}$ VALID				200			300	ns
TG3LF3H	67	$\overline{ASTB}$ LOW to $\overline{AFLG}$ HIGH (Last Strobe)				230			300	ns

Notes:
1. All input transition times assumed ≤20ns.
2. Parameter TCHF1L applies to all input blocks except the first (when S/$\overline{S}$ first goes HIGH).
3. When S/$\overline{S}$ goes inactive (LOW) in direct control mode, the flag associated with the input port will turn off.
4. Direct control mode only.
5. In Cipher Feedback, the port flag ($\overline{MFLG}$ or $\overline{SFLG}$) will go inactive following the leading edge of the first data strobe ($\overline{MDS}$ or $\overline{SDS}$); in all other modes and operations, the flags go inactive on the eighth data strobe.
6. Do not remove K/$\overline{D}$ until CP is inactive (HIGH).
7. Do not change E/$\overline{D}$ until $\overline{MFLG}$ ($\overline{SFLG}$) is inactive (HIGH).
8. 300ns Min if parity check is needed.
9. In Cipher Feedback mode $\overline{BSY}$ must be inactive before S/$\overline{S}$ goes LOW.
10. $\overline{AFLG}$ must go active (LOW) before $\overline{ASTB}$ goes active (LOW).
11. This limit is valid when the clock frequency is 4MHz. At slower clock rates, the range is wider.

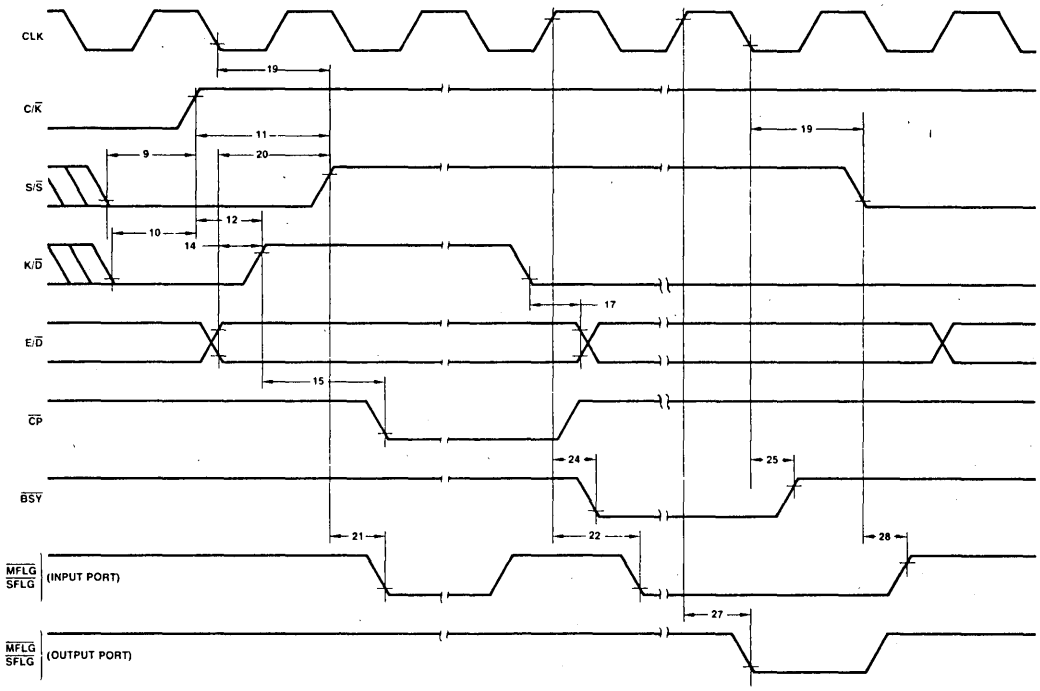


Figure 9. Control and Status Signals (Direct Control Mode)

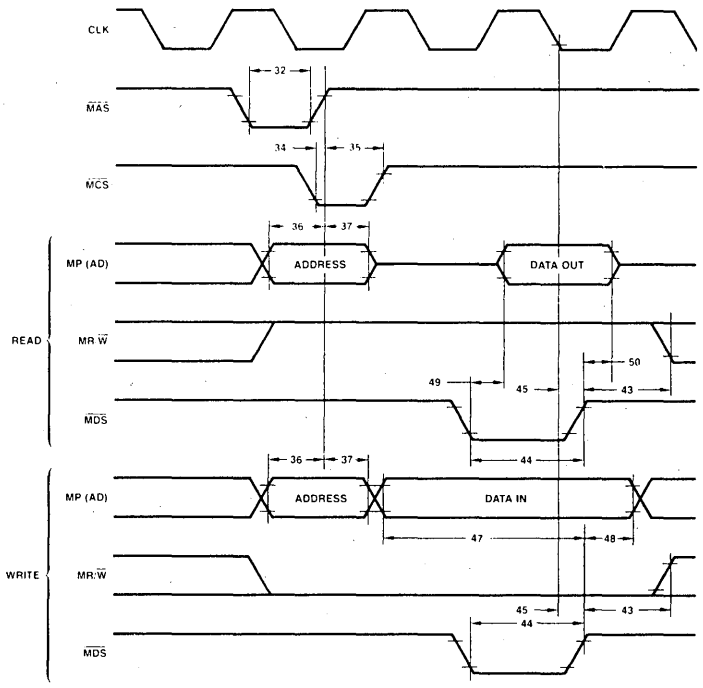


Figure 10. Master Port, Multiplexed Control Mode Read/Write Timing

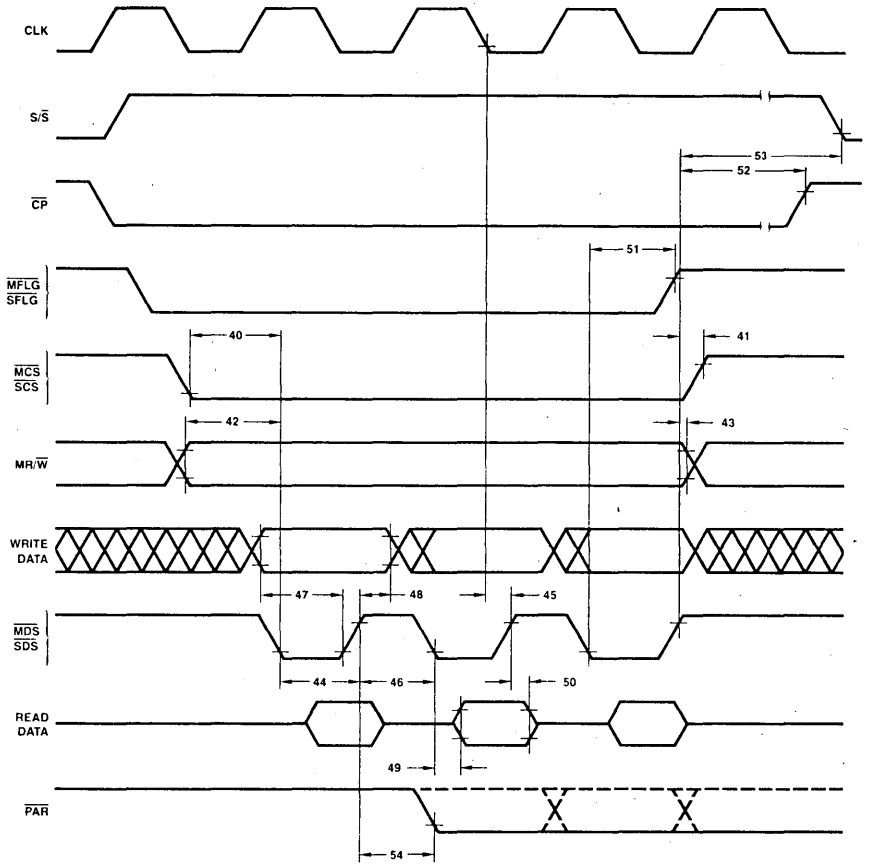


Figure 11. Master (Slave) Port Read/Write

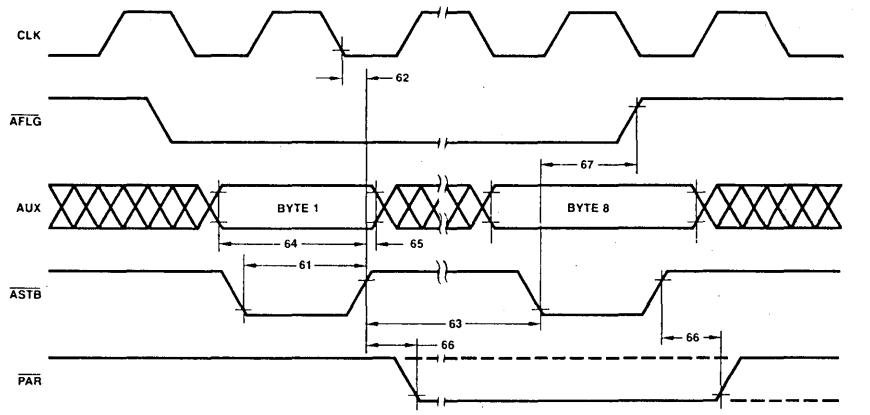


Figure 12. Auxiliary Port Key Entry

Z8516/Z9516 DMA Transfer Controller (DTC)

October 1988

FEATURES

- Two independent multi-function channels
- Transfer Modes: single, demand dedicated with bus hold, demand dedicated with bus release, demand interleave
- Memory/peripheral transfers up to 2.66 Megabyte/second at 4MHz and 4 Megabyte/second at 6MHz
- Memory/memory flowthrough transfer up to 1.33 Megabyte/second at 4MHz and 2 Megabyte/second at 6 MHz
- 16 Megabyte physical addressing range in each address space
- Data types: byte-to-byte, word-to-word, byte/word funneling
- Automatic loading/reloading of control parameters by each channel
- Optional automatic chaining of operations
- Masked data pattern matching for search operations
- Vectored interrupts on selected transfer conditions
- Software or hardware wait state insertion
- Address increment, decrement, or hold
- Channel interleave operations
- Interleave operations with system bus
- Base registers for efficient repetitive operations
- Reload word table for efficient channel initialization
- Software DMA request

GENERAL DESCRIPTION

The Z8516/Z9516 Universal DMA Transfer Controller (DTC) is a high performance peripheral interface circuit for non-Z-BUS CPUs (Figure 1). In addition to providing data block transfer capability between memory and peripherals, each of the DTC's two channels can perform peripheral-to-peripheral and memory-to-memory transfers (Figure 2). A special Search Mode of Operation compares data read from a memory or peripheral source with the contents of a pattern register.

For all DMA operations (search, transfer, and transfer-and-search), the DTC can operate with either byte or word data sizes. In some system configurations it may be necessary to transfer between word-organized memory and a byte-oriented peripheral. The DTC provides a byte packing/unpacking capability through its byte-word funnelling transfer or transfer-and-search option. Some DMA applications may continuously transfer data between the same two memory areas; these applications may not require the flexibility inherent in reloading registers from memory tables. To service these repetitive DMA operations, base registers, which reinitialize the current source and destination Address and Operation Count registers, are provided on each channel. To change the data transfer direction under CPU control, provision is made for reassigning the source address as a destination and the destination as a source, eliminating the need for actual reloading of these address registers.

DMA devices frequently must interface to slow peripherals or slow memory. In addition to providing a hardware WAIT input, the Z8516/Z9516 DTC allows the user to program the automatic insertion of either 0, 1, 2, or 4 wait states for either source or destination addresses. The user may even disable the wait pin function and exclusively use these software-programmed wait states.

High throughput and powerful transfer options are less useful if a DMA requires frequent reloading by the host CPU. The Z8516/Z9516 minimizes CPU interactions by allowing each channel to load its control parameters from memory into the channel's control registers. The only CPU action required is to load the control parameter table's address into the channel's Chain Address register and then issue a Start Chain Command to start the register loading operation. This reloading operation is called command chaining and the table is called the Chain Control Table.

The Z8516/Z9516 DTC is packaged in a 48-pin DIP and uses a single +5V power supply.

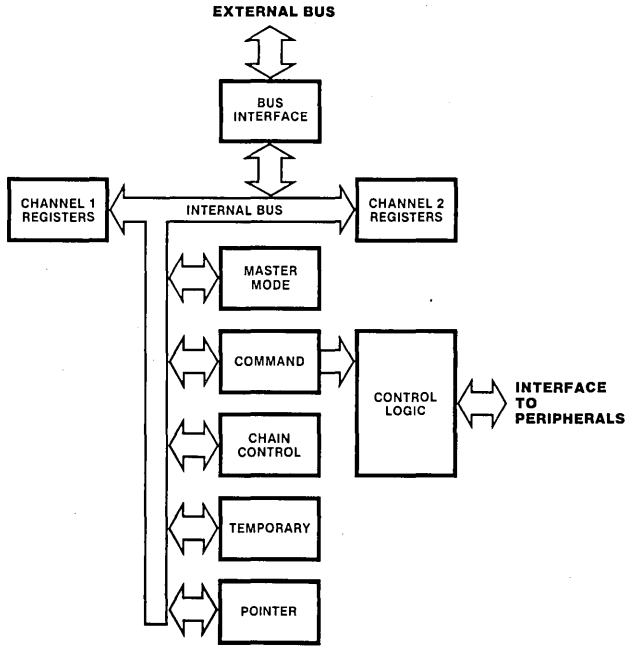


Figure 1. DTC Block Diagram

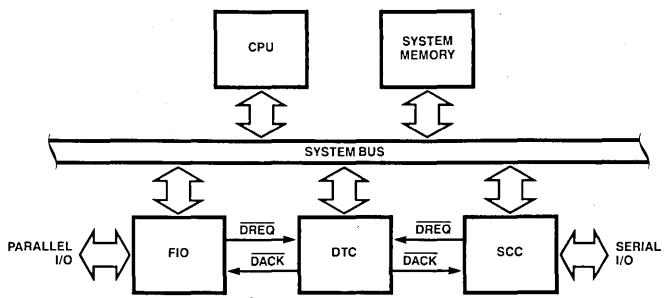


Figure 2. DTC Configuration

SIGNAL DESCRIPTIONS (Figures 3 and 4)

AD_0-AD_{15}. *Address/Data Bus* (bidirectional, active High, 3-state). The time-multiplexed bus is used for all I/O and memory transactions. AD_0 is the least significant bit position; AD_{15} is the most significant. The presence of addresses is defined by the timing edge of ALE; the asserted or requested presence of data is defined by the DS signal. When the DTC is in control of the system bus, it dominates the AD Bus; when the DTC is not in control of the system bus, the CPU or other external devices dominate the AD Bus.

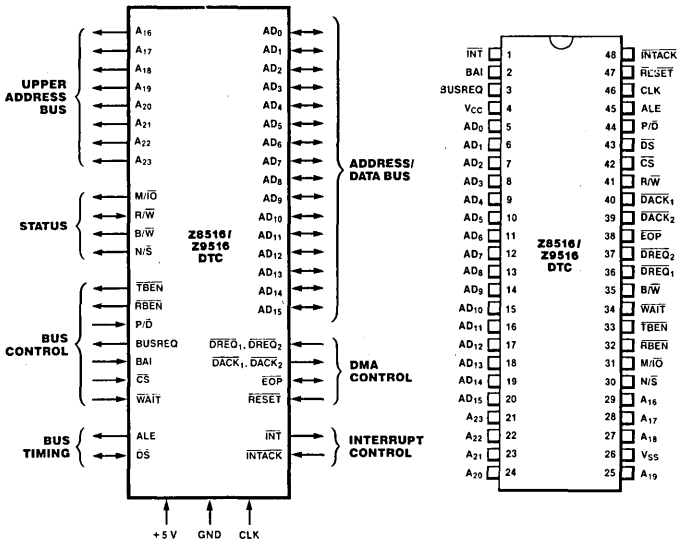


Figure 3. Pin Functions

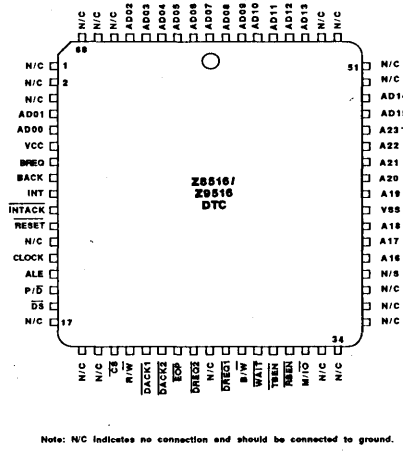


Figure 4. Pin Assignments

A_{16}-A_{23}. *Upper Address Bus* (output, 3-state). A_{16}-A_{23} are activated only when the DTC is controlling the system bus. Combined with the lower 16 address bits appearing on AD_0 through AD_{15}, this 24-bit linear address allows the DTC to access anywhere within 16 Megabytes of memory.

ALE. *Address Latch Enable* (output, active High). This signal is provided by the DTC to latch the address signals AD_0-AD_{15} into the address latch. This pin is never floated.

BAI. *Bus Acknowledge In* (input, active High). BAI is an asynchronous signal indicating that the CPU has relinquished the bus and that no higher priority device has assummed bus control. Since BAI, before being used, is internally synchronized by the DTC, transitions on BAI do not have to be synchronous with the DTC clock. The BAI input is usually connected to the HLDA line from the CPU or to the output of a priority decoder.

BUSREQ. *Bus Request* (output, active High). This signal is used by the DTC to obtain control of the bus from the CPU. BUSREQ lines from multiple devices are connected to a priority encoder.

B/$\overline{W}$. *Byte/Word* (output, 3-state). This output indicates the size of data transferred on the AD_0-AD_{15} bus. High indicates a byte (8-bit) transfer; Low indicates a word (16-bit) transfer. This output is activated when ALE is High and remains valid for the duration of the whole transaction. All word-sized data is word aligned and must be addressed by even addresses (A_0 = 0). When addressing byte read transactions, the least significant address bit determines which byte is needed; an even address specifies the most significant byte (AD_8-AD_{15}) and an odd address specifies the least significant byte (AD_0-AD_7). This addressing mechanism applies to memory accesses as well as I/O accesses. When the DTC is a slave, it ignores the B/$\overline{W}$ signal and this pin floats to 3-state OFF.

CLK. *DTC Clock* (input). The clock signal controls the internal operations and the rates of data transfers. It is usually derived from a master system clock or the associated CPU clock. The Clock input requires a high voltage input signal. Many DTC input signals can make transitions independent of the DTC clock; these signals can be asynchronous to the DTC clock. On other signals, such as $\overline{WAIT}$ inputs, transitions must meet setup and hold requirements relative to the DTC clock.

$\overline{CS}$. *Chip Select* (input, active Low). A CPU or other external device uses $\overline{CS}$ to activate the DTC for reading and writing of its internal registers. There are no timing requirements between the $\overline{CS}$ input and the DTC clock; the $\overline{CS}$ input timing requirements are only defined relative to $\overline{DS}$ signal timings. This pin is ignored when the DTC is in control of the system bus.

$\overline{DACK}_1$, $\overline{DACK}_2$. *DMA Acknowledge* (output, active Low, one per channel). $\overline{DACK}$ indicates that the channel is performing a DMA operation. $\overline{DACK}$ is pulsed, held active, or held inactive during DMA operations as programmed in the Channel Mode register. For Flowthrough operations, the peripheral is fully addressed using the conventional I/O addressing protocols and therefore may choose to ignore

$\overline{DACK}$. $\overline{DACK}$ is always output as programmed in the Channel Mode register for a DMA operation, even when the operation is initiated by a CPU software request command or as a result of chaining. $\overline{DACK}$ is not output during the chaining operations.

$\overline{DREQ_1}$, $\overline{DREQ_2}$. *DMA Request* (input, active Low, one per channel). $\overline{DREQ}$ may make transitions independent of the DTC clock; these lines are used by external logic to initiate and control DMA operations performed by the DTC.

$\overline{DS}$. *Data Strobe* (bidirectional, active Low). A Low on this signal indicates that the AD_0-AD_{15} bus is being used for data transfer. When the DTC is not in control of the system bus and the external system is transferring information to or from the DTC, $\overline{DS}$ is a timing input used by the DTC to move data to or from the AD_0-AD_{15} bus. Data is written into the DTC by the external system on the Low-to-High $\overline{DS}$ transition. Data is read from the DTC by the external system while $\overline{DS}$ is Low. There are no timing requirements between $\overline{DS}$ as an input and the DTC clock; this allows use of the DTC with a system bus which does not have a bussed clock (Figure 26). During a DMA operation when the DTC is in control of the system, $\overline{DS}$ is an output generated by the DTC and used by the system to move data to or from the AD_0-AD_{15} bus. When the DTC has bus control, it writes to the external system by placing data on the AD_0-AD_{15} bus before the High-to-Low $\overline{DS}$ transition and holding the data stable until after the Low-to-High $\overline{DS}$ transition; while reading from the external system the Low-to-High transition of $\overline{DS}$ inputs data from the AD_0-AD_{15} bus into the DTC (Figure 27).

$\overline{EOP}$. *End of Process* (bidirectional, active Low, open drain). $\overline{EOP}$ must be pulled up with an external resistor of 1.8 ohm or more. When a TC or MC termination occurs, the DTC emits an output pulse on $\overline{EOP}$. An external source may terminate a DMA operation in progress by driving $\overline{EOP}$ Low. $\overline{EOP}$ always applies to the active channel; if no channel is active, $\overline{EOP}$ is ignored.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). $\overline{INT}$ is used to interrupt the CPU. It is driven Low whenever the IP and CIE bits of the Status Register are set. It is cleared by the DTC after receiving a clear IP command.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). $\overline{INTACK}$ indicates that the request for interrupt has been granted. The DTC places a vector onto the AD bus if the No Vector on Interrupt bit (MM3) is reset.

M/$\overline{IO}$. *Memory/Input-Output* (output, 3-state). This signal specifies the type of transaction. A High on this pin indicates a memory transaction; a Low indicates an I/O transaction. It floats to a tri-state level when DTC is not in control of the system bus.

N/$\overline{S}$. *Normal/System* (output, 3-state). This signal is activated only when DTC is the master. Normal is indicated when N/$\overline{S}$ is High. This signal supplements the M/$\overline{IO}$ line and is used to indicate whether memory or I/O space is being accessed.

P/$\overline{D}$. *Pointer/Data* (input). This signal indicates information on the AD_0-AD_{15} bus only when the DTC is the bus slave. A High on this signal indicates that the information on the AD bus is an address of the internal register to be accessed. The data on the AD bus is loaded into the Pointer register of the DTC. A Low on this signal indicates that a data transfer is taking place between the bus and the internal register designated by the Pointer register. Note that if a transaction is carried out with R/$\overline{W}$ High and P/$\overline{D}$ High, the contents of the Pointer register will be read.

$\overline{RBEN}$. *Receive Buffer Enable* (output, open drain, active Low). When DTC is in control of the system bus, a Low on this output indicates that the data is being transferred from the data bus lines to the DTC through the buffer. The purpose of this signal is to eliminate bus contention. This pin floats to a tri-state level when the DTC is not in control of the system bus.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ disables the DTC and clears its Master Mode register.

R/$\overline{W}$. *Read/Write* (bidirectional, 3-state). Read polarity is High and write polarity is Low. R/$\overline{W}$ indicates the data direction of the current bus transaction, and is stable from when ALE is High until the bus transaction ends. When the DTC is not in control of the system bus and the external system is transferring information to or from the DTC, R/$\overline{W}$ is a status input used by the DTC to determine if data is entering or leaving on the AD_0-AD_{15} bus during $\overline{DS}$ time. In such a case, Read (High) indicates that the system is requesting data from the DTC and Write (Low) indicates that the system is presenting data to the DTC. There are no timing requirements between R/$\overline{W}$ as an input and the DTC clock; transitions on R/$\overline{W}$ as an input are only defined relative to $\overline{DS}$. When the DTC is in control of the system bus, R/$\overline{W}$ is an output generated by the DTC, with Read indicating that data is being requested from the addressed location or device, and Write indicating that data is being presented to the addressed location or device. Flyby DMA operations are a special case where R/$\overline{W}$ is valid for the normally addressed memory or peripheral locations and must be interpreted in reverse by the Flyby peripheral that uses it.

$\overline{TBEN}$. *Transmit Buffer Enable* (output, open drain, active Low). When DTC is a bus master, a Low on this output indicates that the data is being transferred through the buffer from the DTC to the data bus lines. The purpose of this signal is to eliminate bus contention. When DTC is not in control of the system bus, these pins float to 3-state OFF.

$\overline{WAIT}$. *Wait* (input, active Low). Slow memories and peripheral devices may use $\overline{WAIT}$ to extend $\overline{DS}$ and $\overline{RBEN}$ or $\overline{TBEN}$ during operation. Unlike the $\overline{CS}$ input, transitions on the $\overline{WAIT}$ input must meet certain timing requirements relative to the DTC clock. The Wait function may be disabled using a control bit in the Master Mode register (MM2).

REGISTER DESCRIPTION

The Z8516/Z9516 DTC block diagram (Figure 1) illustrates the internal registers. Table 1 lists each register with its size and read/write access restrictions. Registers which can be read by the CPU are either fast (F) or slow (S) readable. Fast registers can be read by a normal CPU I/O operation without additional wait states. Reading slow registers requires multiple wait states. Registers can be written (W) to by the host CPU and/or can be loaded by the DMA channel itself during chaining (C). All reads or writes must be word accesses since, in slave mode, the DTC ignores the B/W line. It is the responsibility of the user to supply the necessary external logic if slow readable registers are to be read.

Table 1. DTC Internal Registers

Name	Bit Size	Number	Access Type	Port Address CH-1/CH-2
Master Mode Register	4	1	FW	38
Pointer Register	6	1	FW	
Chain-Control Register	10	1	C	
Temporary Register	16	1	D	
Command Register	8	1	W	2E/2C*
Current Address Register—A:				
Upper-Address/Tag field	14	2	CFW	1A/18
Lower-Address field	16	2	CFW	0A/08
Current Address Register—B:				
Upper-Address/Tag field	14	2	CFW	12/10
Lower-Address field	16	2	CFW	02/00
Base Address Register—A:				
Upper-Address/Tag field	14	2	CFW	1E/1C
Lower-Address field	16	2	CFW	0E/0C
Base Address Register—B:				
Upper-Address/Tag field	14	2	CFW	16/14
Lower-Address field	16	2	CFW	06/04
Current Operation Count	16	2	CFW	32/30
Base Operation Count	16	2	CFW	36/34
Pattern Register	16	2	CSW	4A/48
Mask Register	16	2	CSW	4E/4C
Status Register	16	2	F	2E/2C
Interrupt Save Register	16	2	F	2A/28
Interrupt Vector Register	8	2	CSW	5A/58
Channel Mode Register—High	5	2	CS	56/54
Channel Mode Register—Low	16	2	CSW	52/50
Chain Address Register:				
Upper-Address/Tag field	10	2	CFW	26/24
Lower-Address field	16	2	CFW	22/20

Access Codes: C = Chain Loadable; D = Accessible by DTC channel; F = Fast Readable; S = Slow Readable; W = Writeable by CPU

NOTE: The address of the register to be accessed is stored in the Pointer register.

*The port addresses of the Command register can be used alternately for both channels except when issuing a Set/Clear IP command.

The internal registers are read or written in two steps. When the P/$\overline{D}$ input is High, the address of the register to be accessed is written to the Pointer register. When P/$\overline{D}$ input is Low, the data is read from or written into the desired register which is indicated by the Pointer register. Note that a read with P/$\overline{D}$ High causes the contents of the Pointer register to be read on AD_1 through AD_6.

The DTC registers can be categorized into chip-level registers and channel-level registers.

Chip-Level Registers

Chip-level registers are duplicated for each channel and control the overall operation and configuration of the DTC.

The five chip-level registers are:

- **Master Mode**
  selects the way the DTC chip interfaces to the system
- **Pointer**
  written to by the host CPU when the P/$\overline{D}$ input is High. The data in the Pointer register is the address of the internal register to be accessed
- **Chain Control**
  used by a channel while it is reloading its channel-level registers from memory
- **Temporary**
  used to hold data for Flowthrough Transfer/Transfer-and-Searches
- **Command**
  written to by the host CPU to initiate certain operations within the DTC chip, such as resetting the unit

Master Mode Register. The 4-bit Master Mode register (Figure 5) controls the chip-level interfaces. It can be read from and written to by the host CPU through pins AD_0-AD_3 without wait states, but it is not loadable by chaining. On a reset, the Master Mode register is cleared to all zeroes.

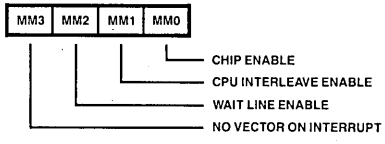


Figure 5. Master Mode Register

The Chip Enable bit, when set to 1, enables the DTC to request the bus. When enabled, the DTC can perform DMA Operations and reload registers. It can always issue interrupts and respond to interrupt acknowledges. When the Chip Enable bit is cleared to 0, the DTC is inhibited from requesting control of the system bus and, therefore, inhibited from performing chaining or DMA operations.

The CPU Interleave bit enables interleaving between the CPU and the DTC.

The Wait Line Enable bit enables sampling of the $\overline{WAIT}$ line during Memory and I/O transactions. Because the DTC provides the ability to insert software programmable wait states, users may disable sampling of the $\overline{WAIT}$ pin to eliminate the logic driving this pin. The Wait Line Enable bit provides this flexibility. The Wait States section of this document includes details on wait state insertion.

The No Vector on Interrupt bit selects whether the DTC channel or a peripheral returns a vector during interrupt acknowledge cycles. When this bit is cleared, a channel receiving an interrupt acknowledge drives the contents of its Interrupt Save register onto the AD_0-AD_{15} data bus while $\overline{INTACK}$ is Low. If this bit is set, interrupts are serviced in an identical manner, but the AD_0-AD_{15} data bus remains in a high impedance state throughout the acknowledge cycle.

Pointer Register. The Pointer register contains the address of the internal register to be accessed. It can be read or written by the CPU when the P/$\overline{D}$ line is High.

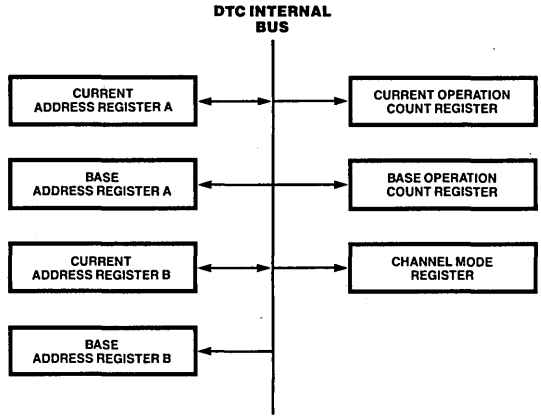
Chain Control Register. When a channel starts a chaining operation, it fetches a Reload word from the memory location pointed to by the Chain Address register (Figure 11). This word is then stored in the Chain Control register. The CPU cannot read to or write from the Chain Control register. Once a channel starts a chain operation, the channel will not relinquish bus control until all registers specified in the Reload word are reloaded unless an $\overline{EOP}$ signal is issued to the chip. Issuing an $\overline{EOP}$ to a channel during chaining prevents the chain operation from resuming and allows the contents of the Reload Word register to be discarded.

Temporary Register. The Temporary register is used to store data during Flowthrough transfers and to hold data being compared during a Search or a Transfer-and-Search. The CPU cannot read to or write from the Temporary register. In byte-word funnelling, data may be loaded into or out of the Temporary register on a byte-by-byte basis, with bytes moving between the low byte of the data bus and the high byte of the Temporary register. The Transfer section carries further details.

Command Register. The DTC Command register (Figure 25) is an 8-bit write-only register written to by the host CPU. The Command register is loaded from the data on AD_7-AD_0; the data on AD_{15}-AD_8 is disregarded. A complete discussion of the commands is given in the Command Descriptions section.

Channel-Level Registers

Each of the DTC's two channels has a complete set of channel-level registers (Figure 6), which can be divided into two subcategories: General Purpose and Special Purpose.



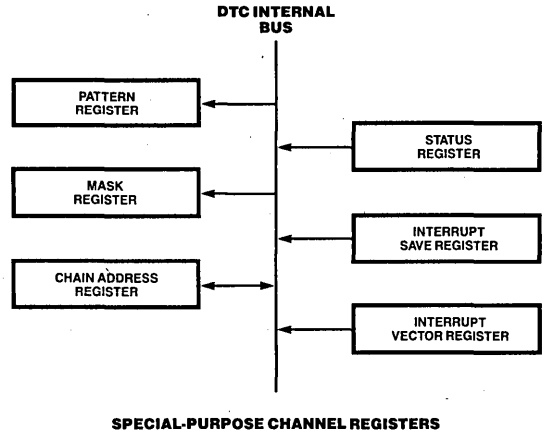


Figure 6. Channel-Level Registers

General Purpose Registers

The general purpose registers are:

- Current Address Register A (ARA)
- Current Address Register B (ARB)
- Base Address Register A (ARA)
- Base Address Register B (ARB)
- Current Operation Count
- Base Operation Count
- Channel Mode

Current and Base Address Registers A and B. The Current Address registers A and B (Current ARA and ARB) are used to point to the source and destination addresses for DMA operations. The contents of the Base Address registers A and B (Base ARA and ARB) are loaded into the Current ARA and ARB registers at the end of a DMA operation if the user enables Base-to-Current reloading in the Completion Field of the Channel Mode register. This facilitates DMA operations without reloading of the Current registers. The ARA and ARB registers can be loaded during chaining, can be written to by the host CPU without wait states, and can be read by the CPU.

Each of the Base and Current ARA and ARB registers consists of two words organized as a 6-bit Tag Field and an 8-bit Upper Address in one word and a 16-bit Lower Address in the other (Figure 7). The Tag Field selects whether the address is to be incremented, decremented, or left unchanged, as well as the status codes associated with the address. The Tag Field also allows the user to insert 0, 1, 2, or 4 wait states into memory or I/O accesses addressed by the offset and segment fields.

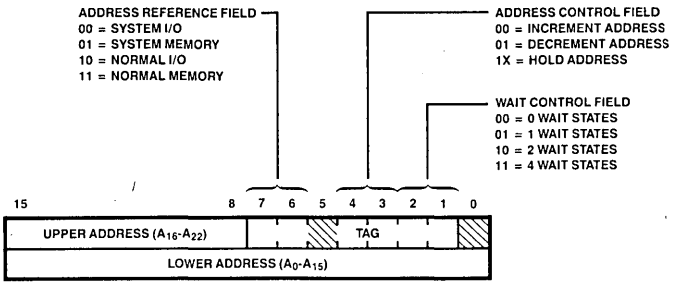


Figure 7. Address Registers A and B

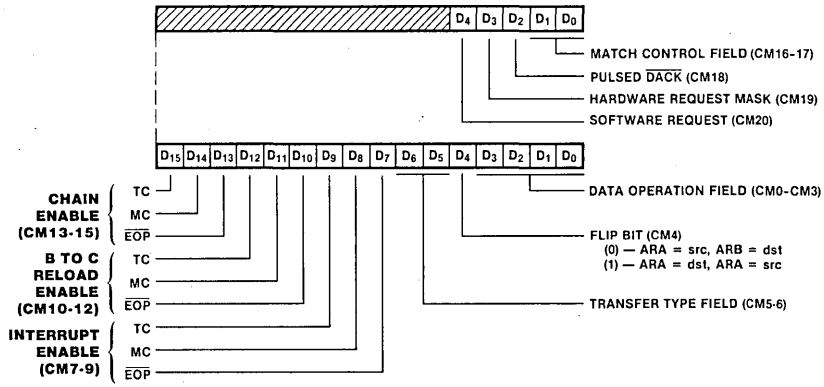


Figure 8. Channel Mode Register

The Address Reference Select Field in the Tag Field selects whether the address pertains to memory space or I/O space. Note that the N/$\overline{S}$ output pin may be either High (indicating Normal) or Low (indicating System). At the end of each iteration of a DMA Operation, the user selects to increment, decrement, or leave the address unchanged. I/O addresses, if changed, are always incremented/decremented by 2. Memory addresses are changed by 1 if the address points to a byte operand (as programmed in the Channel Mode register's Operation field) and by 2 if the address points to a word operand. For word operands, the address must be even to avoid unpredictable results. An even or odd address may be used to point to a byte operand. Since memory byte operand addresses increment/decrement by 1, they toggle between even and odd values. Since I/O byte operand addresses increment/decrement by 2, once programmed to an even or an odd value, they remain even or odd, allowing consecutive I/O operations to access the same half of the data bus. High bus is for even address; low bus is for odd.

Current and Base Operation Count Registers. Both the Current and Base Operation Count registers may be loaded during chaining, and may be written to, and read from, by the host CPU.

The 16-bit Current Operation Count register is used to specify the number of words or bytes to be transferred-and-searched. For word-to-word operations and byte-word funnelling, the Current Operation Count register must be programmed with the number of words to be transferred or searched.

Each time data is transferred or searched, the Operation Count register is decremented by 1. Once all of the data is transferred or searched: the transfer or search operation stops, the Current Operation Count register contains all zeroes, and the TC bit in the Status register is 1. If the transfer or search stops before the Current Operation Count register reaches 0, the contents of the register indicate the number of bytes or words remaining to be transferred or searched. This allows a prematurely stopped channel to be restarted where it left off without requiring reloading of the Current Operation Count register.

For byte-to-byte operations, the Current Operation Count register should specify the number of bytes to be transferred or searched. Setting the Current Operation Count register to 0000 allows the maximum number of 64K bytes to be specified.

Channel Mode Registers. The Channel Mode registers are two words wide. There are 21 bits defined in each Channel Mode register; the other 11 bits are unused. (Figure 8). The Channel Mode registers may be loaded during chaining and may be read by the host CPU. CPU reads of the Channel Mode register are slow reads and require insertion of multiple wait states. The Channel Mode

low word (bits 0-15) may be written to directly by the host CPU. The Channel Mode register selects what type of DMA operation the channel is to perform, how the operation is to be executed, and what action, if any, is to be taken when the channel finishes.

The Data Operation Field and the Transfer Field select the type of operation the channel is to perform and the operand size of bytes or words. The possible bit combinations and their interpretation are given in Table 2. The Flip bit is used to select whether the Current ARA points to the source and the Current ARB points to the destination, or vice-versa. The types of operations are described in detail in the DMA operations section.

Table 2. Channel Mode Coding

Data Operation Field			
	Operand Size		Transaction
Code	ARA	ARB	Type
	Transfer		
0001	Byte	Byte	Flowthrough
100X	Byte	Word	Flowthrough
0000	Word	Word	Flowthrough
0011	Byte	Byte	Flyby
0010	Word	Word	Flyby
	Transfer-and-Search		
0101	Byte	Byte	Flowthrough
110X	Byte	Word	Flowthrough
0100	Word	Word	Flowthrough
0111	Byte	Byte	Flyby
0110	Word	Word	Flyby
	Search		
1111	Byte	Byte	N/A
1110	Word	Word	N/A
101X	Illegal		

Match Control Field/Transfer Type		
Code	**Match Control**	**Transfer Type**
00	Stop on No Match	Single Transfer
01	Stop on No Match	Demand (Bus Hold)
10	Stop on Word Match	Demand (Bus Release)
11	Stop on Byte Match	Demand Interleave

X = Don't care

The Completion Field defines the action taken by the channel at the end of a DMA operation. This field is discussed in the Completion Options section.

The 2-bit Match Control Field selects whether matches use an 8-bit or 16-bit pattern and whether the channel is to Stop-On-Match or Stop-On-No-Match. See Table 2 and the Search section for details.

The Software Request bit and Hardware Mask bit can be set and cleared by software command by loading the Channel Mode register. These bits are described in detail in the Initiating DMA Operations section.

The $\overline{\text{DACK}}$ Control bit is used to specify when the $\overline{\text{DACK}}$ pin is driven active. When this bit is cleared, the channel's $\overline{\text{DACK}}$ pin is active whenever the channel is performing a DMA Operation, regardless of the type of transaction. If this bit is set, the $\overline{\text{DACK}}$ pin is inactive during chaining, Flowthrough Transfers, Flowthrough Transfer-and-Searches, and Searches. It is pulsed active during Flyby Transfers and Flyby Transfers-and-Searches when necessary to strobe data into or out of the Flyby peripheral. Flyby operations are discussed in detail in the Flyby Transactions section.

Special Purpose Registers

The special-purpose registers are:

- Pattern and Mask
- Status
- Interrupt Save
- Interrupt Vector
- Chain Address

Pattern and Mask Registers. The 16-bit Pattern and Mask registers are used in Search and Transfer-and-Search operations. Both the Pattern and Mask registers may be loaded by chaining, and may be written to, and read from, by the host CPU (provided wait states are inserted since these registers are slow readable). The Pattern register contains the pattern which is compared to the read data. Setting a Mask register bit to 1 specifies that the bit always matches. The Search and Transfer-and-Search sections include further details.

Status Register. The two 16-bit Status registers, depicted in Figure 9, are read-only registers which the CPU can read without wait states. Each of these registers reports on the status of its associated channel.

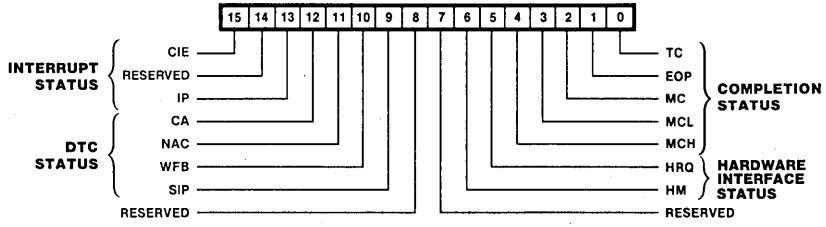


Figure 9. Status Register

The Interrupt Status Field contains the Channel Interrupt Enable (CIE) and Interrupt Pending (IP) bits. These bits are described in detail in the Interrupt section of this document.

The DTC Status Field's four bits are the Second Interrupt Pending (SIP), Waiting For Bus (WFB), No Auto-Reload or Chain (NAC), and Chain Abort (CA) bits. These bits reflect the current channel state and are accessible to the CPU.

When the channel has been properly initialized and is waiting for a command from the host CPU, all four of these bits are set to 0. If the channel requires access to the bus to carry out a DMA operation, it sets the WFB bit. Whether the channel also sets $\overline{\text{BUSREQ}}$ Low depends on the setting of MMO, the Chip Enable bit, and the current status of the bus.

If a channel completes a DMA operation and neither base-to-current reloading nor auto-chaining was enabled, the NAC is set. This bit is reset if the channel receives a Start Chain command.

When two interrupts are queued in the channel, the SIP bit is set, which prohibits any further activity until an Interrupt Acknowledge clears this bit.

Both CA and NAC bits are set by an $\overline{\text{EOP}}$ signal during chaining or if a Reset command is issued to the DTC. The CA bit is cleared when a new Chain Address Segment/Tag word or Offset word is loaded into the Channel's Address registers. NAC cannot be cleared until CA is cleared.

The Hardware Interface Field's Hardware Request ($\overline{\text{HRQ}}$) bit monitors the channel's $\overline{\text{DREQ}}$ input pin. When the $\overline{\text{DREQ}}$ pin is Low, the HRQ bit is set to 1; when the DREQ pin is High, the HRQ pin is cleared to 0. The Hardware Mask (HM) bit, when set, prevents the DTC from responding to a Low on $\overline{\text{DREQ}}$. Note, however, that the Hardware Request bit always reports the true (unmasked) status of $\overline{\text{DREQ}}$ regardless of the setting of the HM bit.

The Completion Field indicates why the most recent DMA operation ended. New data is loaded into these bits overwriting, and thereby erasing, the old setting. Three bits indicate whether the DMA operation ended as a result of a terminal count (TC), match condition (MC), or end-of-process (EOP) termination. If the DMA operation ended as a result of the Operation Count reaching 0, this is a TC termination and STO, the TC bit, is set to 1. The MC bit is set to 1 if an MC termination occurred because the match condition has been met, regardless of whether Stop-On-Match or Stop-On-No-Match was selected. The EOP bit is set to 1 only when an external $\overline{\text{EOP}}$ ends a DMA transfer; it is not set to 1 for an $\overline{\text{EOP}}$ issued during chaining. Note that two, or even all three bits, may be set if multiple reasons exist for ending the DMA operation. The MCH and MCL bits report on the match state of the upper and lower comparator bytes respectively. These bits are set to 1 when the associated comparator byte has a match and are reset otherwise, regardless of whether Stop-On-Match or Stop-On-No-Match is programmed. Regardless of the DMA operation performed, these bits determine which byte matched or did not match when using 8-bit matches with word searches and transfer-and-searches.

The three reserved bits return 0s during reads.

Interrupt Vector and Interrupt Save Registers. Each channel has an Interrupt Vector register and an Interrupt Save register. The Interrupt Vector is 8 bits wide and is written to, and read from, on AD_0-AD_7. The Interrupt Vector register contains the vector or identifier to be output during an Interrupt Acknowledge cycle. When an interrupt occurs (IP = 1) either because a DMA operation terminated or because $\overline{\text{EOP}}$ was driven Low during chaining, the contents of the Interrupt Vector register and part of the Channel Status register are stored in the 16-bit Interrupt Save register (Figure 10). The Interrupt Save register is read without wait states by the CPU.

With the vector and status safely stored, a new vector can be loaded into the Interrupt Vector register during chaining and a new DMA operation can be performed before an interrupt acknowledge cycle occurs. A second interrupt suspends activity in the channel until one of the bits is cleared.

As soon as the first clear IP command is issued, the status and vector for the second interrupt are loaded into the Interrupt Save register and channel operation resumes. The DTC can retain only two interrupts for each channel; a third operation cannot be initiated until the first interrupt has been cleared. The Interrupt section has further details.

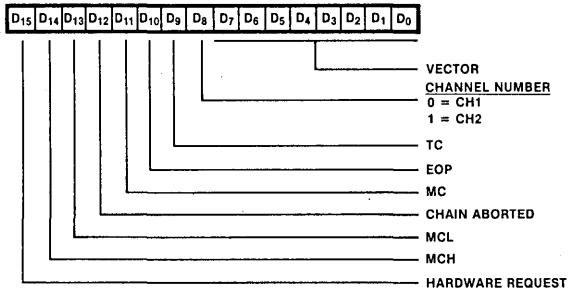


Figure 10. Interrupt Save Register

Chain Address Register. Each channel has a Chain Address register which points to the chain control table in memory containing data to be loaded into the channel's registers. The Chain Address register, as shown in Figure 11, is two words long. The first word consists of an Upper Address and Tag field. The second word contains the 16-bit Lower Address portion of the memory address. The Tag field contains 2 bits used to designate the number of wait states to be inserted during accesses to the Chain Control Table.

The Chain Address register may be loaded during chaining and the host CPU, without wait states, may read from and write to it. During chaining, if an EOP is issued to the DTC, the Chain Address register holds the old address. This is true even if the access failure occurred while new Chain Address data was being loaded, since, unless both words of the new data are successfully read, the old data is restored. EOPs that occur, when chaining and while loading a new Chain Address, cause the new data to be lost.

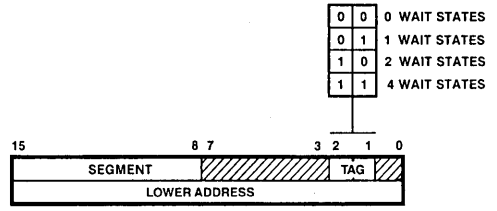


Figure 11. Chain Address Register

FUNCTIONAL DESCRIPTION

Any DMA operation, transfer, search, or transfer-and-search, consists of three phases:

- The channel's registers are initialized to specify and control the desired DMA operation.
- The DMA operation is started and performed.
- The DMA operation is terminated and actions selected to occur on termination are performed.

Reset

Either hardware or software can reset the DTC. The software reset command is described in the Commands section. Hardware resets are applied by pulling $\overline{RESET}$ Low. The DTC may be in control of the bus when a reset is applied. BAI is removed internally causing the outputs to go 3-state. If BAI remains High after reset, the DTC does not drive the bus unless BUSREQ is active. As soon as BAI goes inactive, the DTC places the AD_0-AD_{15}, AD_{16}-AD_{23}, R/$\overline{W}$, $\overline{DS}$, N/$\overline{S}$, M/$\overline{IO}$, B/$\overline{W}$, $\overline{TBEN}$ and $\overline{RBEN}$ signals in the high impedance state.

Both software and hardware resets clear the Master Mode register to all 0s, clear the CIE, IP, and SIP bits to 0, and set the CA and NAC bits to 1 in each Channel's Status register. The contents of all other DTC registers will be unchanged by a software reset. Since a hardware reset may have been applied during a DMA operation being performed by the DTC channel, the channel's registers may contain indeterminate data following a hardware reset.

The Master Mode register contains all 0s after a reset. The DTC is disabled and the CPU interleave and hardware wait are inhibited.

Because the CA and NAC bits in the Status register are set to 1 by a reset, the channel is prevented from starting a DMA operation until its Chain Address register's Segment Tag and Offset fields are programmed and the channel is issued a Start Chain Command.

Channel Initialization

The Z8516/Z9516 DTC operates with a minimum of interaction with the host CPU. This goal is achieved by having the DTC load its own control parameters from memory into each channel. The CPU has to program only the Master Mode register and each Channel's Chain Address register. All other registers are loaded by the channels themselves from a table located in the System memory space and pointed to by the Chain Address register. This reloading operation is called chaining and the table is called the Chain Control Table (Figure 12).

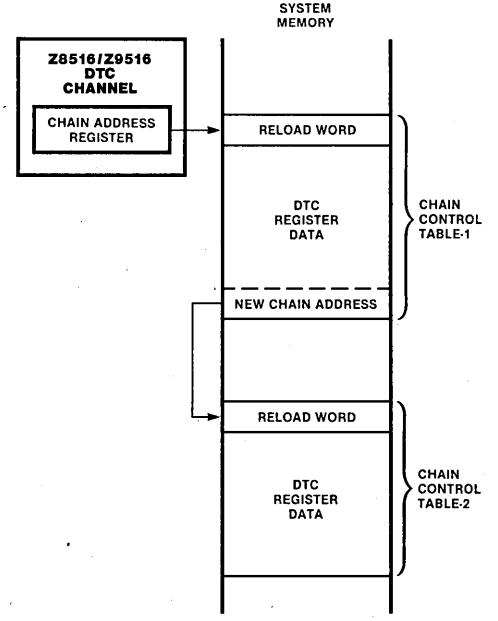


Figure 12. Chaining and Chain Control Tables

The Upper and Lower Address fields of the Chain Address Register form a 24-bit address which points to a location in system memory space. Chaining is performed by repetitively reading words from memory. Note that the Chain Address register should always be loaded with an even Address; loading an odd Address causes unpredictable results. The 2-bit Tag field facilitates interfacing to slow memory by allowing the user to select 0, 1, 2, or 4 programmable wait states. During chaining, the DTC automatically inserts the programmed number of wait states in each memory access.

The Chain Address register points to the Reload Word, the first word in the Chain Control Table. The purpose of the Reload Word is to specify which registers in the channel are to be reloaded. Reload Word bits 10-15 are undefined and may be 0 or 1. Reload Word bits 0 through 9 correspond to either one or two registers in the channel (Figure 13). When a Reload Word bit is 1, the register(s) corresponding to that bit are to be reloaded; if 0, the register(s) corresponding to that bit are not to be reloaded. The data to be loaded into the selected register(s) follow(s) the Reload Word in memory (i.e., the data is stored at successively larger memory addresses). The Chain Control Table is a variable length table; the data is packed together.

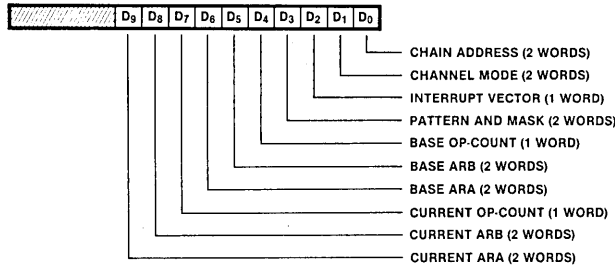
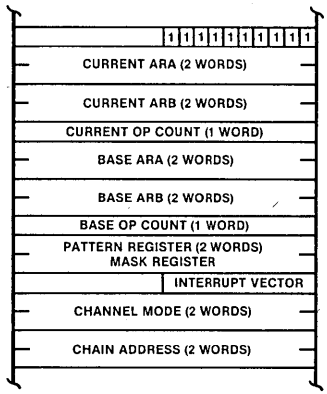
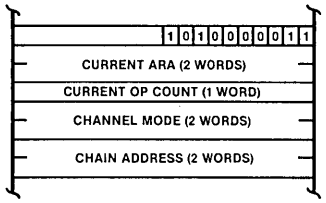


Figure 13. Reload Word/Chain Control Register

When the channel is to reload itself, it first uses the Chain Address register's contents to load the Reload Word into the DTC's Chain Control register. Next, the Chain Address register's contents are incremented by two to point to the next word in memory. The channel then scans the Reload Word register from bit 9 down to bit 0 to see which registers are to be reloaded. If no registers are specified (bits 9-0 are all 0), no registers are reloaded. If at least one of bits 9-0 are set to 1, the register(s) corresponding to the set bit are reloaded, the bit is cleared, and the Chain Address register is incremented by 2. The channel continues this operation of scanning the bits from the most significant to least significant bit position, clearing each set bit after reloading its associated registers, and incrementing the Chain Address register by 2. If all of bits 9 to 0 are set, all the registers will be reloaded in order beginning with Current ARA and ending with Chain Address. Figure 14 shows examples of Chain Control Tables. Example 14a shows the ordering of data when all registers are to be reloaded. In example 14b, only some registers are reloaded. Once the channel is reloaded, it is ready to perform a DMA operation. When loading address registers, the Upper Address and Tag word are loaded before the Lower Address word.



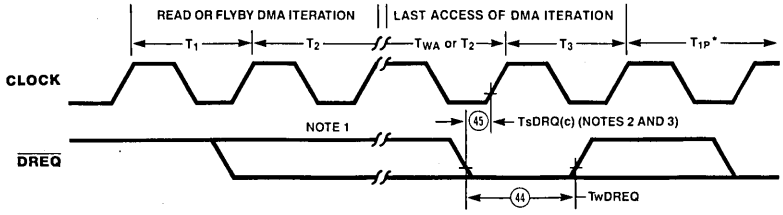


Initiating DMA Operations. DMA operations can be initiated by:

- Software request
- Hardware request
- Starting after chaining

Software Requests. The CPU can issue Software Request commands to start DMA Operations on a channel. The channel must then request control of the bus and perform transfers. See the description of the Software Request command for details.

Hardware Requests. DMA operations can be started by forcing a channel's $\overline{DREQ}$ input Low. The Channel Response describes when the Low $\overline{DREQ}$ signals are sampled and when the $\overline{DREQ}$ requests can be applied to start the next DMA operation after chaining (Figures 15 and 16).



NOTES:
1. HIGH-to-LOW DREQ transitions will only be recognized after the HIGH-to-LOW transition of the clock during T_1 of a read or flyby DMA iteration.
2. A HIGH-to-LOW DREQ transition must meet the conditions in Note 1 and TsDRQ(c) must occur before state T_3 of the last access of the DMA iteration if the channel is to retain bus control and immediately start the next iteration. DREQ may go HIGH before TsDRQ(c) if it has met the TwDRQ parameter.
3. Flyby and Search transactions have only a single access; parameter TsDRQ(c) should be referenced to the start of T_3 of the access. All other operations will always have two or three accesses per iteration.

See Appendix D for timing parameters

*State T_{1P} is a pseudo-T_1 state, without active $\overline{AS}$ generated following termination of any DMA operation.

Figure 15. Sampling DREQ During Single Transfer DMA

Starting After Chaining. If the software request bit of the Channel Mode register is loaded with a 1 during chaining, the channel performs the programmed DMA operation at the end of chaining. If the channel is programmed for Single Operation or Demand mode, it performs the operation immediately. The channel gives up the bus after chaining and before the operation if the CPU interleave bit in the Master Mode register is set. See the Channel Response section for details. Note that once a channel starts a chaining operation by fetching a Reload Word, it retains bus control at least until all of the registers specified in the Reload Word have been loaded from memory.

Bus Request/Grant

Before the DTC can perform a DMA Operation, it must gain control of the system bus. The BUSREQ and BAI interface pins provide connections between the DTC and the host CPU and other devices, if present, to arbitrate which device has control of the system bus. When the DTC wants to gain bus control, it drives BUSREQ High.

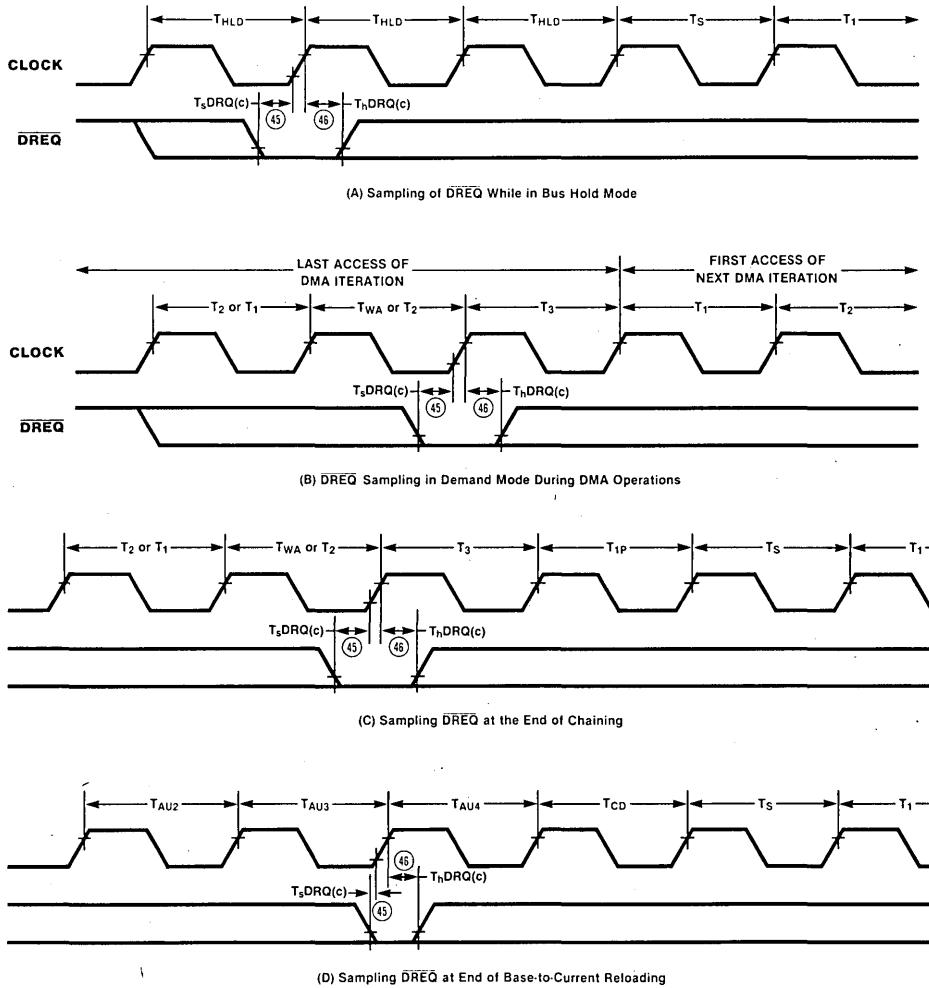
After the DTC drives BUSREQ High, the CPU relinquishes bus control and drives its bus acknowledge signal Low. When the DTC's BAI input goes High, it may begin performing operations on the system bus. When the DTC finishes its operation, it stops driving BUSREQ High.

When more than one device is used, a priority encoder/decoder or hardware daisy-chain encoder and a priority decoder are used to decide the bus grant priority.

DMA Operations

There are three types of DMA operations: Transfer, Search, and Transfer-and-Search. Transfers move data from a source location to a destination location. Two types of transfers are provided: Flowthrough and Flyby. Searches read data from a source and compare the read data to the contents of the Pattern register. A Mask register allows the user to declare "don't care" bits.

The user can program the search to stop either when the read data matches the masked pattern (Stop-On-Match) or when the read data fails to match the masked pattern (Stop-On-No-Match). Transfer-and-Search combines the two functions to facilitate the transferring of variable length data blocks. Like Transfer, Transfer-and-Search can be performed in either Flowthrough or Flyby mode.



NOTES: 1. DREQ must be LOW from the start of T_sDRQ(c) to the end of T_hDRQ(c) to ensure that the request is recognized. Failure to meet this setup time will result in the channel releasing the bus.
2. T_S is a setup state, generated before entering DMA operation cycle.
3. T_{AU2}, T_{AU3}, and T_{AU4} are auto-reload states, followed by TCD (chain decision) state.

Figure 16. DREQ Sampling Demand Mode

Transfers. The transfer operation uses four channel registers:

- Current ARA
- Current ARB
- Current Operation Count
- Channel Mode

Channel Mode register bit CM_4 is called the Flip bit and is used to select whether Current ARA is to point to the source and Current ARB is to point to the destination or whether Current ARA is to point to the destination and Current ARB is to point to the source. The Current Operation register specifies the number of words or bytes to be transferred.

Bits CM_3-CM_0 in the Channel Mode register program whether a Flowthrough or Flyby transfer is to be performed in either two or three steps. First, the channel outputs the address of the source and reads the source data into the DTC's Temporary register. In two-step Flowthrough Transfer, the channel then addresses the destination and writes the Temporary register data to the destination location. The three-step Flowthrough operation (i.e. the byte-word funnelling) is described later in this section. The source and destination for Flowthrough Transfers can be memory

locations, peripheral devices, or a memory location and a peripheral device. The $\overline{DACK}$ output for the transferring channel may be programmed to be inactive or active during the transfer. This is controlled by bit CM_{18} in the Channel Mode register.

Flyby transfers provide improved transfer throughput over Flowthrough but are restricted to transfers between memory and peripherals or between two peripherals. Flyby operations are described in detail in the Flyby Transactions section.

Transfers can use both byte- and word-sized data. Flowthrough byte-to-byte transfers are performed by reading a byte from the source and writing a byte to the destination. The Current Operation Count register must be loaded with the number of bytes to be transferred. Both the Current ARA and ARB registers, if programmed to increment/decrement, will change by 1 if the register points to a memory space (TG_6 = 2) and by 2 if the register points to an I/O space (TG_6 = 0).

Flowthrough word-to-word transfers require that the Current Operation Count specify the number of words to be transferred. Both the Current ARA and ARB registers, if programmed to increment/decrement, will change by 2 regardless of whether the register points to a memory or an I/O space.

Byte-word funnelling provides packing and unpacking of byte data to facilitate high speed transfers between byte and word peripherals and/or memory. This funnelling option can only be used in Flowthrough mode. Funnelled Flowthrough transfers are performed in three steps. For transfers from a byte source to a word destination, two consecutive byte reads are performed from the source address. The data read is assembled into the DTC's Temporary register. In the third step, the Temporary register data is written to the destination address in a word transfer. Funnelled transfers from a word source to a byte destination are performed by first loading a word from the source into the DTC's Temporary register. The word is then written out to the destination in two byte writes. For funnel operations, the byte-oriented address must be in the Current ARA register and the word-oriented address must be in the Current ARB register. The Flip bit (CM_4) in the Channel Mode register is used to specify which address is the source and which is the destination. When the byte address is to be incremented or decremented, the increment/decrement operation occurs after each of the two reads or writes. The Current Operation Count Register must be loaded with the number of words to be transferred.

In byte-to-word funnelling operations it is necessary to specify which half of the Temporary register is written out first. Table 3 summarizes these characteristics for both byte-to-word and word-to-byte funnelling operations. The criteria to determine the packing/unpacking order is based on whether the Current ARB register is programmed for incrementing or decrementing of the address. Note that if the address is to remain unchanged (i.e. if bit TG_4 on the Tag Field of the Current ARB register is 1), the increment/decrement bit (bit TG_3) still specifies the packing order.

Search. Searches use five of the Channel registers:

- Current ARA
- Current ARB
- Operation Count
- Pattern and Mask
- Channel Mode

Channel Mode register bit CM_4 is called the Flip bit and is used to select either Current ARA or Current ARB as the register specifying the source for the search. Only one of the Current Address registers is used for search operations since there is no destination address required. The Current Operation Count register specifies the maximum number of words or bytes to be searched.

Search operations involve repetitive reads from the peripheral or memory until the specified match condition is met. The search then stops. This is called a Match Condition or MC termination. Each time a read is performed, the Source address, if so programmed, is incremented or decremented by 1. If the match condition has not been met by the time the Operation Count reaches 0, the 0 value forces a TC termination, ending the search. Searches can also stop due to a Low being applied to the $\overline{EOP}$ interface pin. During a search operation, the channel's $\overline{DACK}$ output

Table 3. Byte/Word and Word/Byte Funneling

Funneling Direction	Current ARB Tag Field		Increment/Decrement and Packing/Unpacking Rules
	TG_4	TG_3	
Word-to-Byte (Flip Bit = 1)	0	0	Incremnt ARB, Write High Byte First
	0	1	Decrement ARB, Write Low Byte First
	1	0	Hold ARB, Write High Byte First
	1	1	Hold ARB, Write Low Byte First
Byte-to-Word (Flip Bit = 0)	0	0	Increment ARB, Read High Half of Word First
	0	1	Decrement ARB, Read Low Half of Word First
	1	0	Hold ARB, Read High Half of Word Written First
	1	1	Hold ARB, Read Low Half of Word Written First

will be either inactive or active throughout the search. This is controlled by bit CM_{18} in the Channel Mode register. The peripheral or memory reads performed during search follow the timing sequences described in the Flowthrough Transactions sections.

On each read during a Search operation, the DTC's Temporary register is loaded with data and compared to the Pattern register. The user can select whether the search is to stop when the Pattern and Temporary register contents match or when they do not match. This Stop-On-Match/ Stop-On-No-Match feature is programmed in bit CM_{17} of the Channel Mode register. CM_2 is an enable for the output of the comparator and allows the MC signal to be generated. A Mask register allows the user to exclude, or mask, selected Temporary register bits from the comparison by setting the corresponding Mask register bit to 1. The masked bits always are defined to match. Thus, in Stop-On-Match, successful matching of the unmasked bits, in conjunction with the always-matched masked bits, causes the search to stop. For Stop-On-No-Match, the always-matched masked bits are, by definition, excluded from not matching and therefore excluded from stopping the search.

For word reads the user may select either 8-bit or 16-bit compares through Channel Mode register bit CM_{16}. In an 8-bit, Stop-On-Match, word-read operation, successful matching of either the upper or lower byte of unmasked Pattern and Temporary registers bits stops the search. Both types do not have to match. In 16-bit Stop-On-Match with word reads, all unmasked Pattern and Temporary register bits must match to stop the search. In an 8-bit or 16-bit, Stop-On-No-Match, word-read Search operation, failure of any bit to match terminates the Search operation.

In an 8-bit Stop-On-Match with byte-reads, the Search stops if either the upper or lower byte of unmasked Pattern and Temporary register bits match. For an 8-bit Stop-On-No-Match with byte reads, failure of matching in any unmasked Pattern and Temporary register bit causes the search to stop. For 8-bit searches, the upper and lower bytes of the Pattern and Mask register should usually be programmed with the same data. Failure to set the upper and lower bytes of the Pattern and Mask registers to identical values results in different comparison criteria being used for the upper and lower bytes of the Temporary register. Users failing to program identical values for the upper and lower bytes can predict the results by recognizing that in 8-bit Stop-On-Match, the search ends if all the unmasked bits in either the upper or lower byte match, and for 8-bit Stop-On-No-Match, the failure of any unmasked bit to match ends the search. For word reads the Temporary register high and low bytes are loaded from AD_8-AD_{15} and AD_0-AD_7 respectively. In byte reads, except in funnelling, the read byte is duplicated in both halves of the Temporary register.

Transfer-and-Search. Transfer-and-Search combines the operations of the Transfer and the Search functions. The registers which control Transfer-and-Searches are:

- Current ARA
- Current ARB
- Operation Count
- Pattern and Mask
- Channel Mode

A Transfer-and-Search operation ends when the data transferred meets the match condition specified in Channel Mode register bits CM_{17}-CM_{16}. The Mask and Pattern registers indicate those bits being compared with the Temporary register contents. Like Transfers and Searches, Transfers-and-Searches are also terminated if the operation count goes to 0 or if a Low is applied to the EOP pin. Regardless of whether Transfer-and-Search stops because of a TC, MC, or EOP, it always completes the iteration by writing to the destination address before ending (writing twice for word-to-byte funnelling).

In Flowthrough mode, the Transfer-and-Search timing is identical to Flowthrough Transfer. While the data is in the Temporary register, it is masked by the Mask register and compared to the Pattern register. For word Transfer and Transfer-and-Search, the high and low bytes of the Temporary register are always written to, and read from, AD_8-AD_{15} and AD_0-AD_7 respectively. For byte Transfer and Transfer-and-Search, the byte read is always loaded into both halves of the Temporary register and the entire register is driven directly out onto the AD_0-AD_{15} bus. Transfer-and-Search can also be used with byte word funnelling. In funnelling, the match is an 8-bit match as determined by the setting of the bit CM_{16}.

Flyby Transfer-and-Search can be used to increase throughput for transfer between two peripherals or between memory and a peripheral. Memory-to-Memory Flyby is not supported. Also, in Flyby, the operand sizes of the source and destination must be the same; funnelling is not supported. A complete discussion of Flyby timing is given in the Flyby Transactions section. During a Flyby Transfer-and-Search, data is loaded into the Temporary register to facilitate the comparison operation and, at the same time, data is transferred from the source to the destination. When byte operands are used, data is loaded into both bytes of the Temporary register, from the AD_8-AD_{15} bus if the Current ARA register is even, and from AD_0-AD_7 line if the Current ARA register is odd. This alternates for memory bytes so the user must drive both halves of the bus to use the search. When word operands are used, data is loaded directly from AD_8-AD_{15} and AD_0-AD_7 into the Temporary register's high and low bytes respectively.

Channel Response

Channel Mode register bits CM_6-CM_5 select the channel's response to the request to start a DMA operation. The response falls into either of two types: Single Operation or Demand. There are three subtypes for Demand operations: Demand Dedicated with Bus Hold, Demand Dedicated with Bus Release, and Demand Interleave. For Search operations, one iteration consists of a single read operation and a comparison of the read data to the unmasked Pattern register bits. The Operation Count is decremented by 1 and the Current Address register, if so programmed, is incremented or decremented. For Transfer and Transfer-and-Search operations, a single iteration comprises reading data from the source, writing it to the destination, comparing the read data to the unmasked Pattern register bits (Transfer-and-Search only), decrementing the Operation Count by 1, and incrementing/decrementing the Current ARA amd ARB registers if so programmed. In byte-word funnelling, a single iteration consists of two reads followed by a write (Byte-to-Word funnelling) or one read followed by two writes (Word-to-Byte funnelling). In all Transfer and Transfer-and-Search cases the iteration does not stop until the data in the Temporary register is written to the destination. (Appendix B).

Single Operation. The Single Operation response is used with peripherals which transfer single bytes or words at irregular intervals. Each Software Request command causes the channel to perform a single iteration of the DMA operation. Similarly, if the Software Request bit is set by chaining, the channel performs a single iteration of the DMA operation at the end of chaining. Each application of a High-to-Low transition on the $\overline{DREQ}$ input also causes a single iteration of the DMA operation. If the hardware mask bit is set when the High to Low transition is made, the iteration is performed when the mask is cleared, providing the DMA operation has not terminated. See the Set/Clear Hardware Mask bit command in the Command section for details. Each time a Single Operation ends, the channel gives up control of the bus unless a new transition has occurred on $\overline{DREQ}$. The new transition can occur anytime after the High-to-Low ALE transition of a read or Flyby memory or I/O access of the DMA iteration. Figure 15 shows the times after which a new transition can be applied and recognized to avoid giving up the bus at the end of the current iteration.

Demand Dedicated With Bus Hold. In Demand Dedicated with Bus Hold (abbreviated Bus Hold), the application of a Software Request command, the setting during chaining of the software request bit, or applying a Low level on the $\overline{DREQ}$ input, causes the channel to acquire bus control.

If $\overline{DACK}$ is programmed as a level output (CM_{18} = 0), $\overline{DACK}$ is active while the channel controls the bus. A Software Request causes the channel to request the bus and perform the DMA operations until TC, MC, or EOP occurs.

Once the channel gains bus control due to a Low $\overline{DREQ}$ level, it samples DREQ as shown in Figure 16. If $\overline{DREQ}$ is Low, an iteration of the DMA operation is performed. If $\overline{DREQ}$ is High, the channel retains bus control and continues to drive all bus control signals active or inactive, but performs no DMA operation. Thus the user can start, or stop, execution of DMA operations by modulating $\overline{DREQ}$. Once TC, MC, or EOP occurs, the channel releases the bus, or, if chaining or Base-to-Current reloading is to occur, performs the desired operation. After chaining or Base-to-Current reloading, if the channel is still in Bus Hold mode and does not have a set Software Request bit (set either by chaining or command), the channel relinquishes bus control unless a Low $\overline{DREQ}$ level occurs within the time limits.

Demand Dedicated With Bus Release. In Demand Dedicated with Bus Release (abbreviated Bus Release), a Software Request command causes the channel to request the bus and performs the programmed DMA operation until TC, MC, or EOP occurs. If the channel was programmed for Bus Release and the Software Request bit was set during chaining, the channel starts the DMA operation as soon as chaining ends, without releasing the bus, and continues performing the operation until TC, MC, or EOP occurs.

When an active Low $\overline{DREQ}$ is applied to a channel programmed for Bus Release, the channel acquires the bus and performs DMA operations until TC, MC, or EOP occurs or until $\overline{DREQ}$ goes inactive. Figure 16 shows when $\overline{DREQ}$ is sampled to determine if the channel should perform another cycle or release the bus. Note that this sampling also occurs on the last cycle of a chaining operation. If a channel has an active $\overline{DREQ}$ at the end of chaining, it performs DMA operations immediately, without releasing the bus. When a TC, MC, or EOP occurs terminating a Bus Release mode operation, the channel, if enabled for chaining and/or Base-to-Current Reloading, performs chaining and/or reloading (assuming the Status register's SIP bit is clear) without releasing the bus.

If the SIP bit of Channel Mode register is set when a DMA termination occurs, the channel reliquishes the bus control until an Interrupt Acknowledge is received and the SIP bit is cleared. After an interrupt is serviced, the channel, if enabled for the termination, performs the Base-to-Current reloading and/or chaining.

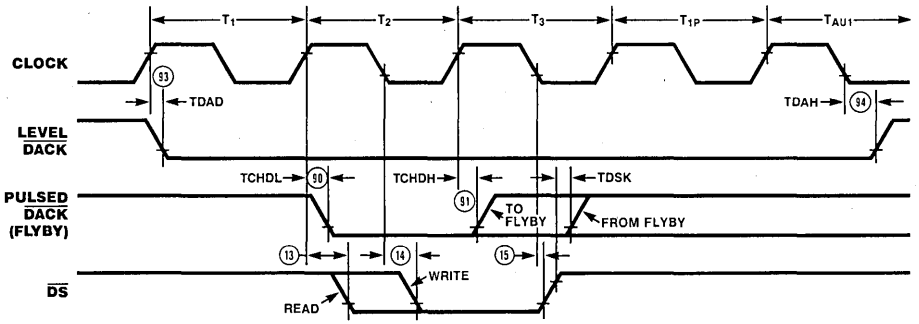
If an active request is not applied and the channel is in Demand Dedicated with Bus Hold, the channel goes into state THLD (Figure 16a). If an active request is not applied and the channel is in Demand Dedicated with Bus Release or Demand Interleave mode, it releases the bus. Note that even if an active request is applied in Demand Interleave, the channel may still release the bus. The request for Demand Interleave should continue to be applied to ensure that the channel eventually responds to the request by acquiring the bus (i.e. the request is not latched by the channel).

Demand Interleave. Demand Interleave behaves in different ways depending on the setting of Master Mode register bit MM2. If MM2 is set, the DTC always reliquishes bus control and then again requests it after each DMA iteration. This permits the CPU and other devices to gain access to the bus in the following execution control sequence: Channel 1, CPU, Channel 2, CPU, Channel 1, CPU. The CPU could be some other external device.

When MM2 is clear and both channels have active requests and are in Demand Interleave mode, control toggles between the channels after each DMA operation iteration and the DTC retains bus control until both channels are finished with the bus. Appendix B's, Figure B.2 is a flowchart of the Demand Interleave operation.

A software or hardware request causes a channel programmed for Demand Interleave to perform interleaved DMA operations until TC, MC, or EOP. If the Software Request bit is set during chaining, the channel retains the bus after chaining and immediately starts performing DMA iterations interleaving after the first operation. If $\overline{DREQ}$ is Low on the last cycle during chaining, the channel performs a single iteration immediately after chaining and interleaves thereafter until TC, MC, or EOP occurs or $\overline{DREQ}$ goes High. If the latter occurs, the channel relinquishes the bus interleaved operations. If a TC, MC, or EOP occurs, the channel first performs chaining and/or Base-to-Current reloading (assuming SIP is cleared) before interleaving.

The waveform of $\overline{DACK}$ is programmed in Channel Mode register (CM_{18}). The Pulsed $\overline{DACK}$ is only for Flyby transactions. Figure 17 shows the timing for a single Search or Flyby operation. State T_{WA} is optionally inserted if programmed. For more than one iteration, the level $\overline{DACK}$ output stays active during the time the channel has bus control. When CM_{18} is set, the $\overline{DACK}$ output is inactive during Flowthrough modes.



NOTES: 1. State T_{1P} is a pseudo-T_1 state, without active $\overline{AS}$ generated following termination of any DMA operation.
2. State T_{AU1}, is an auto-initialization state generated following the TC, MC, or EOP termination.
3. Level DACK Rising Edge occurs as shown if auto-reloading is not programmed, otherwise it stays Low for three additional clock cycles.

Figure 17. $\overline{DACK}$ Timing

Wait States

The number of wait states to be added to the memory or I/O transfer can be programmed by the user as 0, 1, 2, or 4; it can be programmed separately for the Current Address registers A and B and for the Chain Address register. This allows different speed memories and peripherals to be associated with each of these addresses. The Base Address registers A and B also have a Tag Field which is loaded into the Current ARA and ARB registers during Base-to-Current reloading. Because many users utilizing the software programmable wait states do not need the ability to generate hardware wait states through the $\overline{WAIT}$ pin, the wait function can be disabled by clearing the Wait Line Enable bit (MM2) in the Master Mode register.

During DMA transactions, the $\overline{WAIT}$ input is sampled in the middle of the T_2 state. If $\overline{WAIT}$ is High, and if no programmable wait states are selected, the DTC proceeds to state T_3. Otherwise, at least one wait state is inserted. The $\overline{WAIT}$ line is then sampled in the middle of state T_{WA}. If $\overline{WAIT}$ is High the DTC proceeds to state T_3. Otherwise additional wait states are inserted (Figure 18).

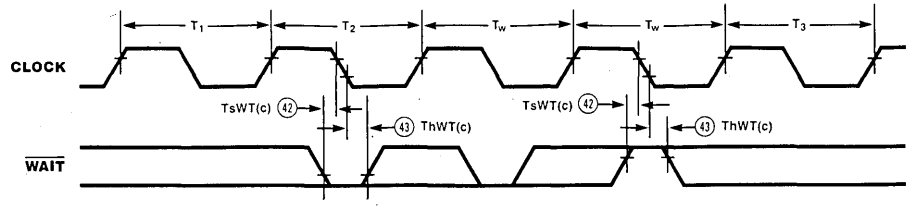


Figure 18. $\overline{WAIT}$ Timing

In a transaction when both hardware and software wait states are inserted, each time the $\overline{WAIT}$ line is sampled Low, a hardware wait state is inserted in the next cycle. The software wait state insertion is suspended until $\overline{WAIT}$ is sampled and is High. The hardware wait states may be inserted anytime during the software wait state sequence. Hardware wait states are served consecutively rather than concurrently with software wait states. For example, assume for a Flowthrough I/O Transaction that a user has programmed four software wait states. Driving a Low on the $\overline{WAIT}$ input during T_2 for two cycles would insert two hardware wait states. Driving $\overline{WAIT}$ High for three cycles would allow insertion of three of the four software wait states. Driving $\overline{WAIT}$ Low for two more cycles would insert two more hardware wait states. Finally, driving $\overline{WAIT}$ High would allow the final software wait state to be inserted. During this last software wait state, the $\overline{WAIT}$ pin would be sampled for the last time. If it is High, the channel proceeds to state T_3. If the pin is Low, the channel inserts hardware wait states until the pin goes High and the channel then enters state T_3 to complete the I/O transaction.

DMA Transactions

There are three types of transactions performed by the Z8516/Z9516 DTC: Flowthrough, Flyby, and Search.

Flowthrough Transactions. A Flowthrough Transaction (Figure 19) consists of three states: T_1, T_2, and T_3 as shown in Figure 20. The user may insert software wait states through the Tag fields of the Current ARA and ARB registers. In addition, if Master Mode register bit MM2 = 1, hardware wait states may be inserted by driving a Low signal on the $\overline{WAIT}$ pin.

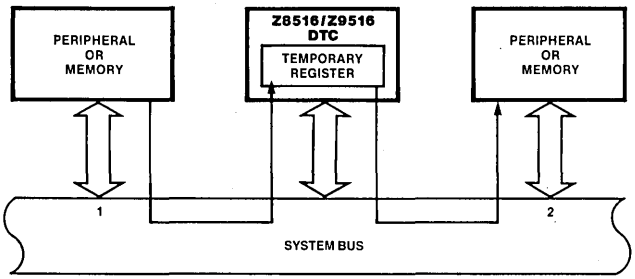


Figure 19. Flowthrough Transaction

The M/$\overline{IO}$ and N/$\overline{S}$ lines reflect the appropriate level for the current cycle early in T_1. The TG_6 and TG_7 bits of the current ARA and ARB registers should be programmed properly. The ALE output is pulsed High to mark the beginning of the cycle. The offset portion of the address for the accessed peripheral appears on AD_0-AD_{15} during T_1. The R/$\overline{W}$ and B/$\overline{W}$ lines select a read or write operation for bytes or words. The R/$\overline{W}$, N/$\overline{S}$, M/$\overline{IO}$, and B/$\overline{W}$ lines become stable during T_1 and remain stable until after T_3.

I/O address space is byte-addressed but both 8- and 16-bit data sizes are supported. During I/O transactions the B/$\overline{W}$ output is High for byte transactions and Low for word transactions. For I/O transactions, both even and odd addresses can be output, hence the address bit output on AD_0 may be 0 or 1.

The channel can perform both I/O read and write operations; the M/$\overline{IO}$ line is Low. During an I/O read, the AD_0-AD_{15} bus is placed in the high impedence state by the DTC during T_2. The DTC drives the $\overline{DS}$ output Low to signal the peripheral that data can be gated onto the bus. The DTC strobes the data into its Temporary register during T_3. DS is driven High to signal the end of the I/O transaction. During I/O write, the DTC drives the contents of the Temporary register onto the AD_0-AD_{15} bus and shortly after drives the DS output Low until T_3. Peripherals may strobe the data on the AD bus into their internal registers on either the clock's falling or rising edge. If the peripheral is also to be accessed in a Flyby transaction, data should be written only on the rising edge of $\overline{DS}$.

For byte I/O writes, the channel drives the same data on data bus lines AD_0-AD_7 and AD_8-AD_{15}. During byte I/O reads, when the address bit on AD_0 is 0, the DTC strobes data in from data lines AD_8-AD_{15}. During byte I/O reads, when the address bit on AD_0 is 1, the DTC strobes data in from data lines AD_0-AD_7. Thus, when an 8-bit peripheral is connected to the bus, its internal registers typically are mapped at either all even or all odd addresses. To simplify accesses to 8-bit peripherals, byte oriented I/O addresses are incremented/decremented by 2.

The channel can perform the I/O read and memory write operation, the memory read and I/O write operation, and the memory read and memory write operation. The timing for all Flowthrough transactions is the same.

During chaining operations the DTC reads words from an address in System memory pointed to by the active channel's Chain Address register. Those chaining operations are performed identically to the Flowthrough memory read transactions, except that the data is loaded into an internal DTC Channel register rather than the Temporary register. Chaining never causes a write or a byte read; thus all memory writes or all byte accesses are due to DMA operations. A typical memory operation consists of three states: T_1, T_2, and T_3, as shown in Figure 20. The user

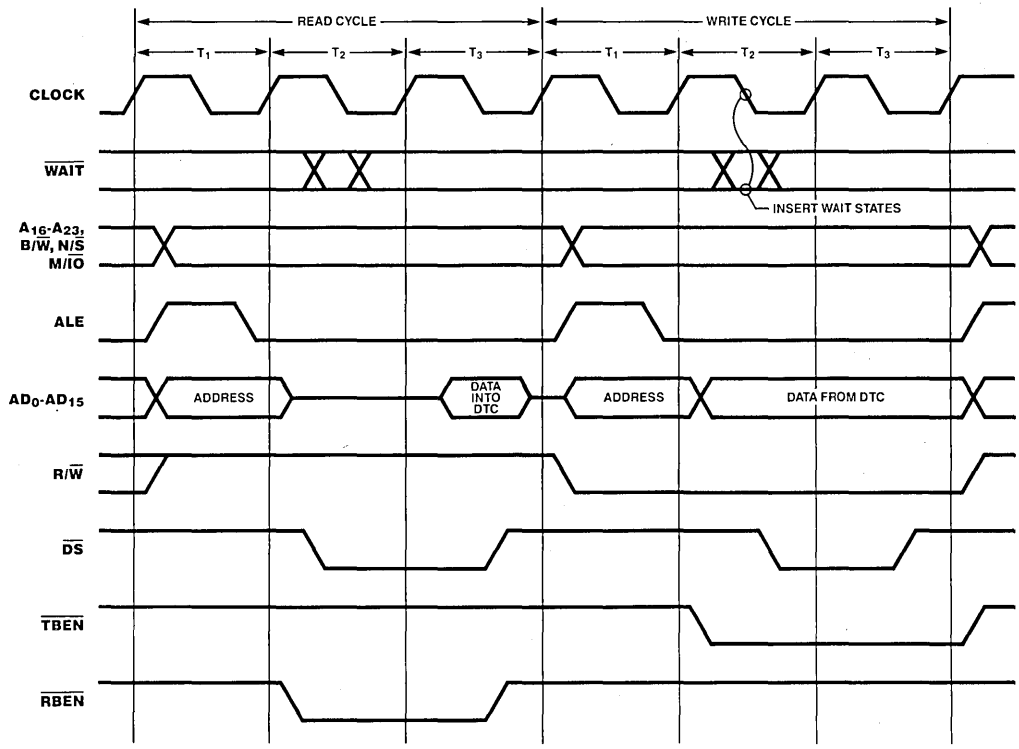


Figure 20. Flowthrough Transaction Timing

may select to insert 1, 2, or 4 software wait states between T_2 and T_3 by programming the Tag field of the Current Address register or the Chain Address register. If the Wait Line Enable bit in the Master Mode register is set, the user may also insert hardware wait states between T_2 and T_3 by driving a Low on the $\overline{WAIT}$ line. The operation of Flowthrough memory transactions is identical to Flowthrough I/O transactions.

Flyby Transactions. Flyby transfer and Flyby transfer-and-search operations are performed in a single cycle, providing a transfer rate significantly faster than Flowthrough. In transfers, Flyby mode operations can only be performed between memory and peripheral or between peripheral and peripheral. Memory-to-memory operations cannot be performed in Flyby mode (Figure 21).

The Flyby Transaction can be used only with peripherals having a special Flyby signal input or with external logic. This Flyby input is connected to the channel's $\overline{DACK}$ output. For memory-peripheral Flyby, the address of the source memory location must be programmed in the Current ARA register. The Current ARB register must be programmed with the destination memory location for peripheral-memory Flyby. For Flyby peripheral-to-peripheral transaction, if both peripherals have a Flyby input, only one (the Flyby peripheral) should be connected to $\overline{DACK}$; the other

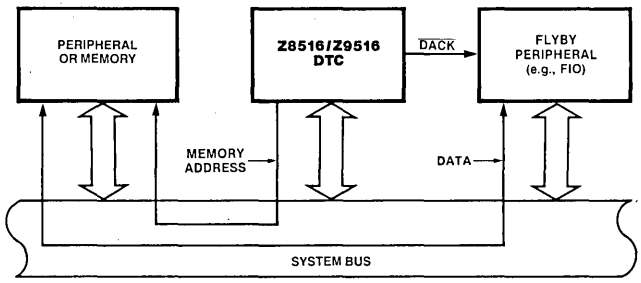


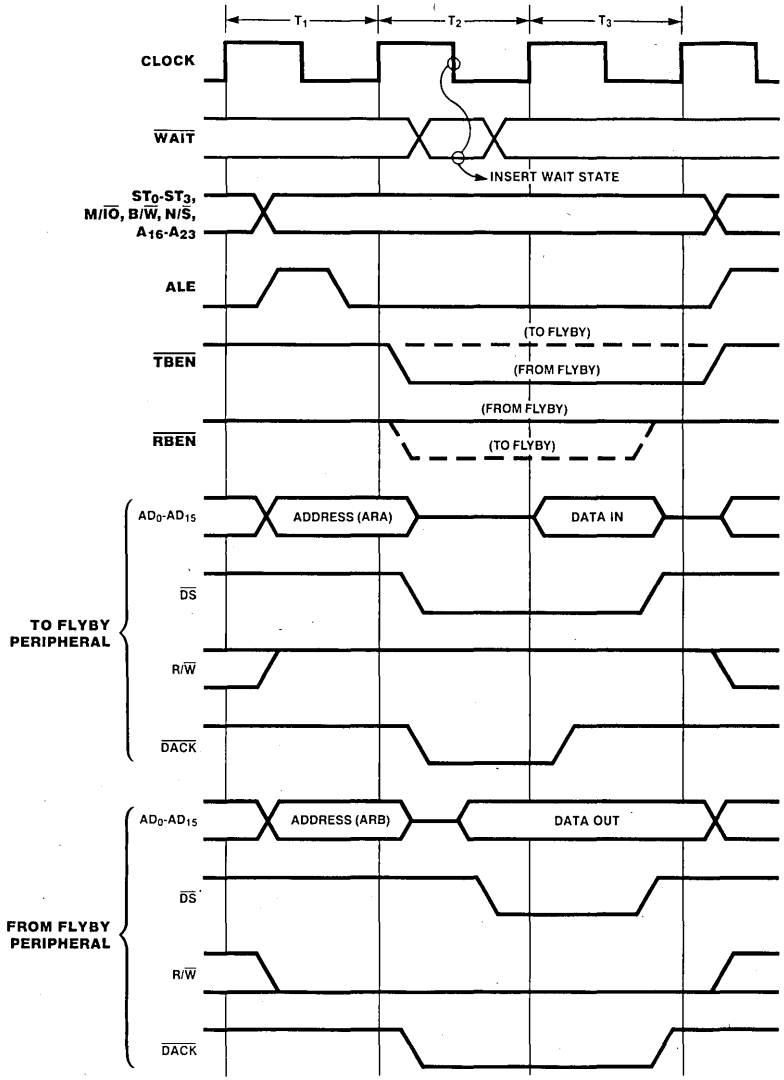
Figure 21. Flyby Transaction

(Non-Flyby) peripheral's Flyby input should be held High during the Flyby operation. When the Non-Flyby peripheral is a destination and not connected to the channel's $\overline{DACK}$ output, its address should be programmed in the current ARB register. When the Non-Flyby peripheral is a source, its address should be programmed in the current ARA register. Table 4 explains that a set Flip bit (CM_4 = 1) is for Flyby peripheral to Non-Flyby peripheral or Memory. Write transaction (From Flyby Transaction) and a clear Flip bit (CM_4 = 0) is for the Memory or Non-Flyby peripheral read to Flyby peripheral transaction (To Flyby Transaction).

A Flyby operation requires three states: T_1, T_2, and T_3. During T_1 the channel pulses ALE and outputs. The R/$\overline{W}$ line is High for To-Flyby Transaction and Low for From-Flyby Transaction (Figure 22).

Table 4. Flyby Transaction

Transaction	CM_4	R/W	Address of Memory or Non-Flyby Peripheral
To Flyby	0	HIGH	ARA
From Flyby	1	LOW	ARB



*Toggles for memory access in logical address space only.
**For physical addressing only.
***N/S will be low for I/O transactions.

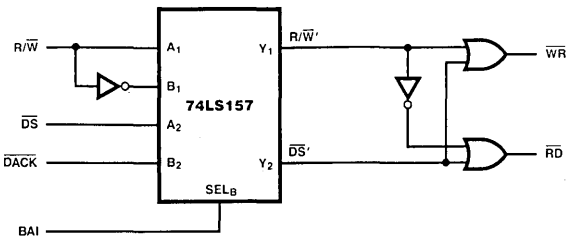
(A) Address is current ARA
(B) Address is current ARB

Figure 22. Flyby Transaction Timing

The channel's M/IO and N/S lines are coded as specified by the Current ARA or ARB Tag field. The B/$\overline{W}$ line indicates the operand size programmed in the Channel Mode register Operation field. During T_1 the channel drives R/$\overline{W}$ to indicate the transaction direction; during T_2 the channel drives both $\overline{DS}$ and $\overline{DACK}$ active. The Flyby Peripheral connected to $\overline{DACK}$ inverts the R/$\overline{W}$ signal to determine whether it is being read from or written to (Figure 23).

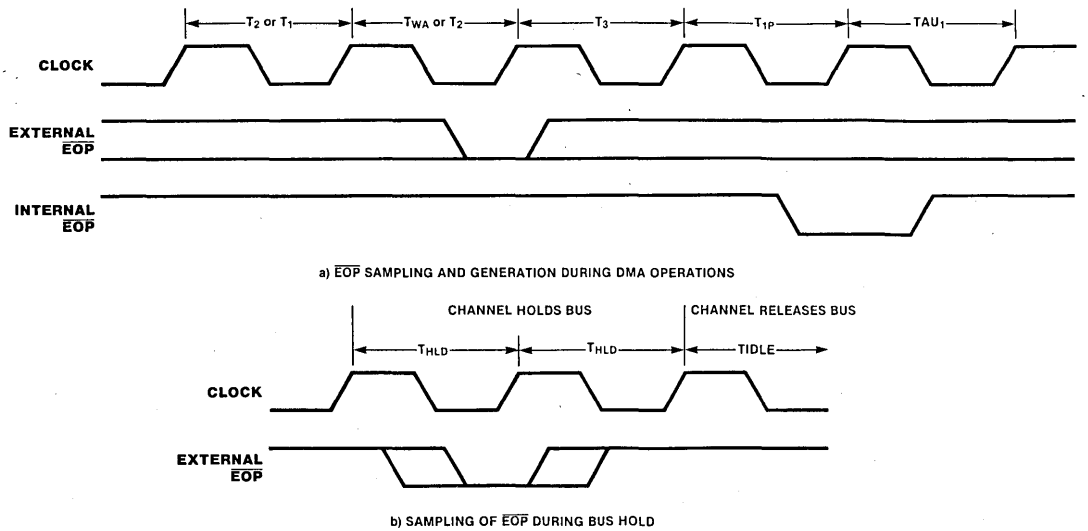
The pulsed $\overline{DACK}$ input serves two purposes: To select the peripheral for the Read/Write, and to provide timing information on when to drive data onto, or input data from, the AD_0-AD_{15} bus. Because the Flyby Peripheral never is explicitly addressed by AD_0-AD_{15}, it must know which internal register is to be loaded from, or driven onto, the AD_0-AD_{15} bus. On state T_3, the $\overline{DS}$ and $\overline{DACK}$ lines are driven inactive to conclude the transfer. In Transfer-and-Search mode, data is loaded into the DTC's Temporary register on the Low-to-High $\overline{DS}$ transition in order to perform the search function.

To provide adequate data setup time, the rising edge of $\overline{DS}$ or $\overline{DACK}$ should be used to perform the write to the transfer destination. To extend the active time of $\overline{DS}$ and $\overline{DACK}$, wait states can be inserted between T_2 and T_3. Software wait states can be inserted by programming the appropriate code in the Tag field of the Current ARA or ARB registers. Hardware wait states can be inserted by pulling $\overline{WAIT}$ Low if the Wait Line Enable bit in the Master Mode register is set. The $\overline{WAIT}$ line is sampled in the middle of the T_2 or T_{WA} state.



$R/\overline{W}' = \overline{BAI \cdot R/\overline{W}} + \overline{BAI \cdot R/W}$
$\overline{DS}' = \overline{BAI \cdot DACK} + \overline{BAI \cdot DS}$
$\overline{RD} = \overline{DACK \cdot R/\overline{W} \cdot BAI} + \overline{DS \cdot R/\overline{W} \cdot BAI}$
$\overline{WR} = \overline{DACK \cdot R/W \cdot BAI} + \overline{DS \cdot R/W \cdot BAI}$

Figure 23. Flyby Peripheral Interface



Notes:
1. The diagram lists state names for both I/O and memory accesses. Sampling of $\overline{EOP}$ will occur on the falling edge of state T_3.
2. State T_{1P} is a pseudo-T_1 state, without active $\overline{AS}$ generated following termination of any DMA operation.
3. State TAU_1 is an auto-initialization state generated following the TC, MC, or $\overline{EOP}$ termination.

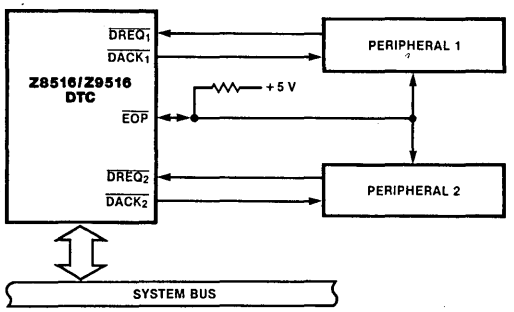
Figure 24. $\overline{EOP}$ Timing

Termination

There are three ways a Transfer-and-Search or Search operation can end and two ways a Transfer operation can end. When a channel's Current Operation Count goes to 0, the DMA operation ends; this is called a Terminal Count (TC) termination. A DMA operation can also be stopped by driving the $\overline{EOP}$ pin Low with external logic; this is called an $\overline{EOP}$ termination. Match Condition (MC) is the last method of termination which occurs when the data being Transferred-and-Searched or Searched meets the match condition programmed in Channel Mode register bits CM_{17}-CM_{16}. These bits allow the user to stop when a match occurs between the unmasked Pattern register bits and the data read from the source, or when a no-match occurs. Both byte and word matches are supported. MC terminations do not apply to Transfer operations since the pattern matching logic is disabled in Transfer mode.

End of Process

The End-of-Process ($\overline{EOP}$) interface pin is a bidirectional signal. Whenever a TC, MC, or $\overline{EOP}$ termination occurs, the DTC drives the $\overline{EOP}$ pin Low. During DMA operations, the $\overline{EOP}$ pin is sampled by the DTC to determine if $\overline{EOP}$ is being driven Low by external logic. Figure 24 shows when internal $\overline{EOP}$s are generated marking termination of all Transfers and when the $\overline{EOP}$ pin is sampled during the DMA iteration. The generation of internal $\overline{EOP}$s and sampling of external $\overline{EOP}$s for Transfer-and-Searches follows the same timing used for Transfers. Since there is a single $\overline{EOP}$ pin for both channels, $\overline{EOP}$s should only be driven Low by a channel while that channel is being serviced. This can be accomplished by selecting a level $\overline{DACK}$ output (CMR_{18} = 0) and gating each channel's $\overline{EOP}$ request with $\overline{DACK}$, as shown in Figure 25.



Notes:
1. External $\overline{EOP}$ stops channel.
2. DTC drives $\overline{EOP}$ Active on TC and MC.
3. Channel should apply $\overline{EOP}$ only if its $\overline{DACK}$ is active.

Figure 25. $\overline{EOP}$ Connection

If an $\overline{EOP}$ is detected while the channel is trying to reload the Chain Address register, the new Chain Address Offset and Segment are discarded and the old address +2 is preserved to allow inspection of the erroneous address.

Programming Completion Options. When a channel ends a DMA operation, the reason for ending is stored in the Completion Status Field of the channel's Status register (Figure 7). This information is retained until the next DMA operation ends at which time the Status register is updated to reflect the reason(s) for the latest termination. More than one bit in the Completion Field could be set to 1. All three of the channel's Status register completion bits would be set to 1 under the following conditions: If a channel decremented its Current Operation Count to 0 causing a TC termination, input data from the source generated a match causing an MC termination, and a Low on the $\overline{EOP}$ pin resulted in an $\overline{EOP}$ termination.

When a DMA operation ends, the channel can:

(a) Issue an Interrupt request (i.e., setting the IP or SIP bit of the channel's Status register)

(b) Perform Base-to-Current reloading

(c) Chain reload the next DMA operation

(d) Perform any combination of the above or

(e) None of the above

The user selects the action to be performed by the channel in the Completion option field of the Channel Mode register. For each type of termination (TC, MC or EOP) the user can choose which action or actions are to be taken. If no reloading is selected for the type of termination that occurred, the NAC bit in the Status register is set.

More than one action can occur when a DMA operation ends. This may arise because more than one action was programmed for the applicable termination. The priorities of those actions are Interrupt request, Base-to-Current reloading, and chaining. The Interrupt cannot be serviced unless the DTC has relinquished the bus.

Interrupts

To permit the DTC to begin a new DMA operation after issuing an interrupt but before the CPU acknowledges that interrupt, a two-deep interrupt queue is provided on each channel of the DTC. Interrupt handling by the Z8000 microprocessor is summarized in this section, followed by a brief discussion of the DTC's queueing capability and its implications for the system.

A complete Interrupt cycle on the Z8000 CPU consists of an Interrupt Request followed by an Interrupt Acknowledge transaction. The request, which consists of the CPU's Interrupt pin being pulled Low by a peripheral, notifies the processor that an interrupt is pending. The Interrupt Acknowledge cycle, initiated by the CPU as a result of the interrupt request, performs two functions: it selects the peripheral whose interrupt is to be acknowledged and it obtains a vector that identifies the device involved and the reason for the interrupt.

The DTC has two sources of interrupt. Each source has three bits that control its interrupt generation. These bits are the Channel Interrupt Enable (CIE), Interrupt Pending (IP), and Interrupt Under Service (IUS) bits. Since each channel on the DTC contains all three of these bits (bits CM_{15}-CM_{13}), they are seen by the CPU as two separate interrupt sources. Each channel also has its own vector register for identifying the source of the interrupt during an Interrupt Acknowledge interchange with the CPU. The Disable Lower Chain (DLC) and No Vector (NV) bits in the DTC's Master Mode register control this behavior for the entire chip.

Once a channel issues an interrupt, it is desirable to allow the channel to proceed with the next DMA operation before the interrupt is acknowledged. This could lead to problems if the DTC channel attempted to chain reload the Vector register contents. In such a situation, it may not be clear whether the old or new vector would be returned during the acknowledge. This dilemma is resolved in the DTC by providing each channel with an Interrupt Save register. When the channel sets IP as part of the procedure followed to issue an interrupt, the contents of the vector register and some of the Status register bits are saved in an Interrupt Save register (Figure 9). When an Interrupt Acknowledge cycle is performed, the contents of the Interrupt Save register are driven onto the bus. Although the use of an Interrupt Save register allows the channel to proceed with a new task, problems can still arise if a second interrupt is to be issued by the channel before the first interrupt is acknowledged. To avoid conflicts between the first and second interrupt, each channel has a Second Interrupt Pending (SIP) bit in its Status register. When a second interrupt is issued before the first interrupt is acknowledged, the SIP bit is set and the channel relinquishes the bus until an acknowledge occurs. For compatibility with polled interrupt schemes, the Interrupt Save register can be read without wait states by the host CPU. As an aid to debugging a system's interrupt logic, whenever IP is set, the Interrupt Save register is loaded from the Vector and Status registers.

Note that the SIP bit is transferred to the IP bit when IP is cleared by the host CPU. Whenever CIE is set, INT goes Low when IP is set.

Base-to-Current Reloading. When a channel finishes a DMA operation, the user may select to perform a Base-to-Current Reload. (Base-to-Current reloading is also referred to as Auto-reloading in this document.) In this type of reload, the Current Address registers A and B are loaded with the data in the Base Address registers A and B respectively, and the Current Operation Count register is loaded with the data in the Base Operation Count. The Base-to-Current reload operation facilitates repetitive DMA operations without the multiple memory accesses required by chaining. Although the channel must have bus control to perform Base-to-Current reloading, the complete reloading operation occurs in four clock cycles (TAU_1 through TAU_4). If the channel has to relinquish the bus because two unacknowledged interrupts are queued, it has to regain bus control to perform any Base-to-Current reloading (or chaining). In this case it acquires the system bus once an interrupt acknowledge is received, even if it immediately afterward relinquishes the bus because no hardware or software request is present.

Chaining. If the channel is programmed to chain at the end of a DMA operation, it uses the Chain Address register to point to a Chain Control Table in memory. The first word in the table is a Reload word, specifying the register(s) to be loaded. Following the Reload word are the data values to be transferred into the register(s). Chaining is described in detail in the Channel Initialization section.

Because chaining occurs after Base-to-Current reloading, it is possible to reset the Current Address registers A and B and the Current Operation Count register to the values used for previous DMA operations and then chain reload one or two of these registers to some special value. If the Base values are not reloaded during chaining, the channel can revert back to the Base values at a later cycle.

If an all zero Reload word is fetched during chaining, the chain operation does not reload any registers but performs like any other chaining operation. Thus, the Chain Address is incremented by 2 to point to the next word in memory and, at the end of the all Zero-Reload word chain operation, the channel is ready to perform a DMA operation. All zero Reload words are useful as "Stubs" to start or terminate linked lists of DMA operations traversed by chaining. Care must be taken in their use since the channel may perform an erroneous operation if it is unintentionally started after the chaining operation.

COMMANDS

Table 5 shows a list of DTC commands. The commands are executed immediately after the host CPU writes them into the DTC's Command register (Figure 26). A description of each command follows.

Table 5. DTC Command Summary

Command	Opcode Bits		Example Code HEX
	7654	3210	
Reset	000X	XXXX	00
Start Chain Channel 1	101X	XXX0	A0
Start Chain Channel 2	101X	XXX1	A1
Set Software Request Channel 1	010X	XX10	42
Set Software Request Channel 2	010X	XX11	43
Clear Software Request Channel 1	010X	XX00	40
Clear Software Request Channel 2	010X	XX01	41
Set Hardware Mask Channel 1	100X	XX10	82
Set Hardware Mask Channel 2	100X	XX11	83
Clear Hardware Mask Channel 1	100X	XX00	80
Clear Hardware Mask Channel 2	100X	XX01	81
Set CIE, or IP Channel 1	001E	XP10	32
Set CIE, or IP Channel 2	001E	XP11	33
Clear CIE, or IP Channel 1	001E	XP00	30
Clear CIE, or IP Channel 2	001E	XP01	31
Set Flip Bit Channel 1	011X	XX10	62
Set Flip Bit Channel 2	011X	XX11	63
Clear Flip Bit Channel 1	011X	XX00	60
Clear Flip Bit Channel 2	011X	XX01	61

NOTES: E = Set to 1 to perform set/clear on CIE; clear to 0 for no effect on CIE
P = Set to 1 to perform set/clear on IP; clear to 0 for no effect on IP
X = "dont't care" bit. This bit is not decoded and may be 0 or 1

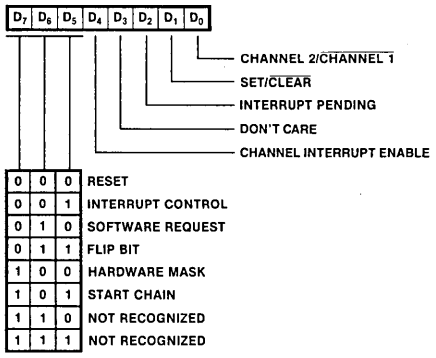


Figure 26. Command Register

Reset (00)

This command causes the DTC to be set to the same state as a hardware reset. The Master Mode register is cleared to all 0s, the CIE, IP, and SIP bits are cleared to 0, the NAC and CA bits in each channel's Status register are set to 1, and the channel activity is forbidden. The Chain Address must be programmed since its state may be indeterminate after a Reset. The lockout preventing channel activity is cleared by issuing a Start Chain command.

Start Chain Channel 1/Channel 2 (A_0/A_1)

This command causes the selected channel to clear the No Auto-Reload or Chain (NAC) bit in the channel's Status register, and to start a chain reload operation of the channel's registers, as described in the Channel Initialization section. These effects take place even if the fetched Reload word is all zeros. This command is only honored if the Chain Abort (CA) bit and the Second Interrupt Pending (SIP) bit in the Channel's Status register are clear. If either the CA or SIP bit is set, this command is disregarded.

When the Waiting For Bus (WFB) bit of the Status Register is set, if the "Start Chain" command is issued, the channel honors the command after one DMA iteration. It is nearly impossible for the CPU to issue a command when WFB = 1 and the DTC is enabled.

Software Request Channel 1/Channel 2 (Set: 42/43, Clear: 40/41)

This command sets or clears the Software Request bit in the selected channel's Mode register. If the Second Interrupt Pending (SIP) bit and No Auto-Reload or Chain (NAC) bit in the channel's Status register are both cleared, the channel begins executing the programmed DMA operation. If either the SIP or NAC bit is set, the channel does not start executing a DMA operation. The SIP bit clears when the channel receives an interrupt acknowledge. One way to clear the NAC bit is to issue a Start Chain command to the channel. If the fetched Reload Word is all zeros, the channel's registers remain unchanged and the software request bit, if set earlier by command, causes the programmed DMA operation to start immediately. If during chaining, new information is loaded into the Channel Mode register, this new information overwrites the software request bit.

Set/Clear Hardware Mask 1/Mask 2 (Set: 82/83; Clear: 80/81)

This command sets or clears the Hardware Mask in the selected channel's Mode register. This command always takes effect. The Hardware Mask bit inhibits recognition of an active signal on the channel's $\overline{DREQ}$ input; this bit does not affect recognition of a software request. If the channel is in single transfer mode, it performs DMA operations upon receipt of a transition on $\overline{DREQ}$ rather than in response to a $\overline{DREQ}$ level. Transitions, occurring while the Hardware mask bit is set, are stored and serviced when the Hardware Mask is cleared, assuming the Channel has not chained. The DTC requests the system bus one and one half to two clock cycles after the receipt of any $\overline{DREQ}$, after which a minimum of one DMA iteration is unavoidable. $\overline{DREQ}$ transitions are stored only for the current DMA operation. If the channel performs a chain operation of single transfer mode, any $\overline{DREQ}$ transition stored for later service is cleared.

Figures 15 and 16 show the minimum times when a new $\overline{DREQ}$ can be applied if it is to be serviced by the new DMA operation. First iteration and Last iteration in Figure 15 mean, for example, that $\overline{DREQ}$ may be asserted during the write cycle T_1 of a Flowthrough transaction, but may never be asserted during T_1 of a Flyby transaction since Flyby is done in one iteration.

Set/Clear CIE, and IP Channel 1/Channel 2 (Table 5)

This command allows the user to set or clear any combination of the CIE and IP bits in the selected channel's Status register. These bits control the operation of the channel's interrupt structure and are described in the Interrupts section. Setting the IP bit causes the Interrupt Save register to be loaded with the current vector and status. The IP bit is cleared to facilitate an efficient conclusion to the processing of an interrupt.

Set/Clear Flip Bit Channel 1/Channel 2 (Set: 62/63; Clear 60/61)

The Flip Bit in the selected channel's Mode register can be cleared and set by this command. This allows the user to reverse the source and destination and thereby reverse the data transfer direction without reprogramming the channel. This command is useful when repetitive DMA operations are performed by the channel, using this command to control the direction of transfer. Chaining new information into the Channel Mode register overwrites the Flip bit.

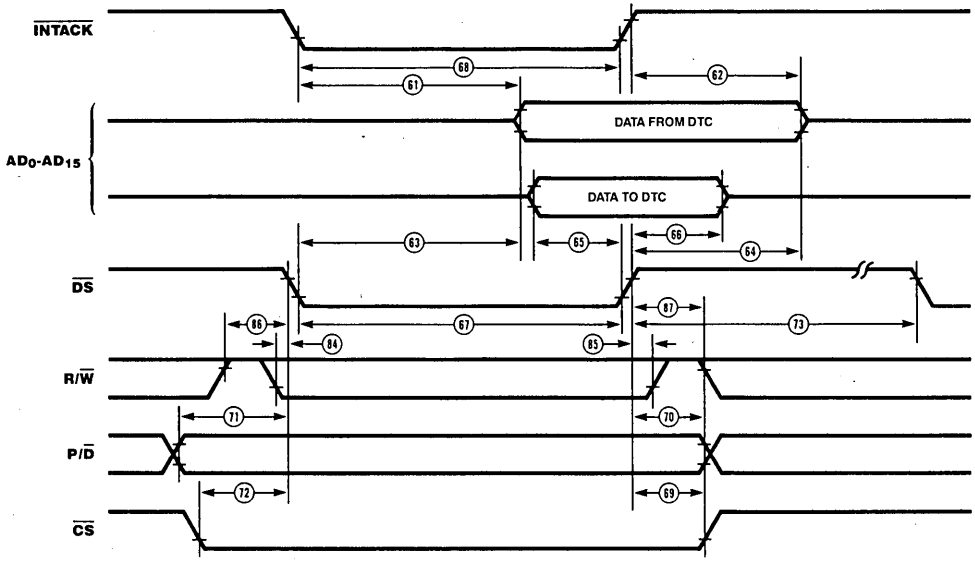


Figure 27. AC Timing when DTC is a Bus Slave

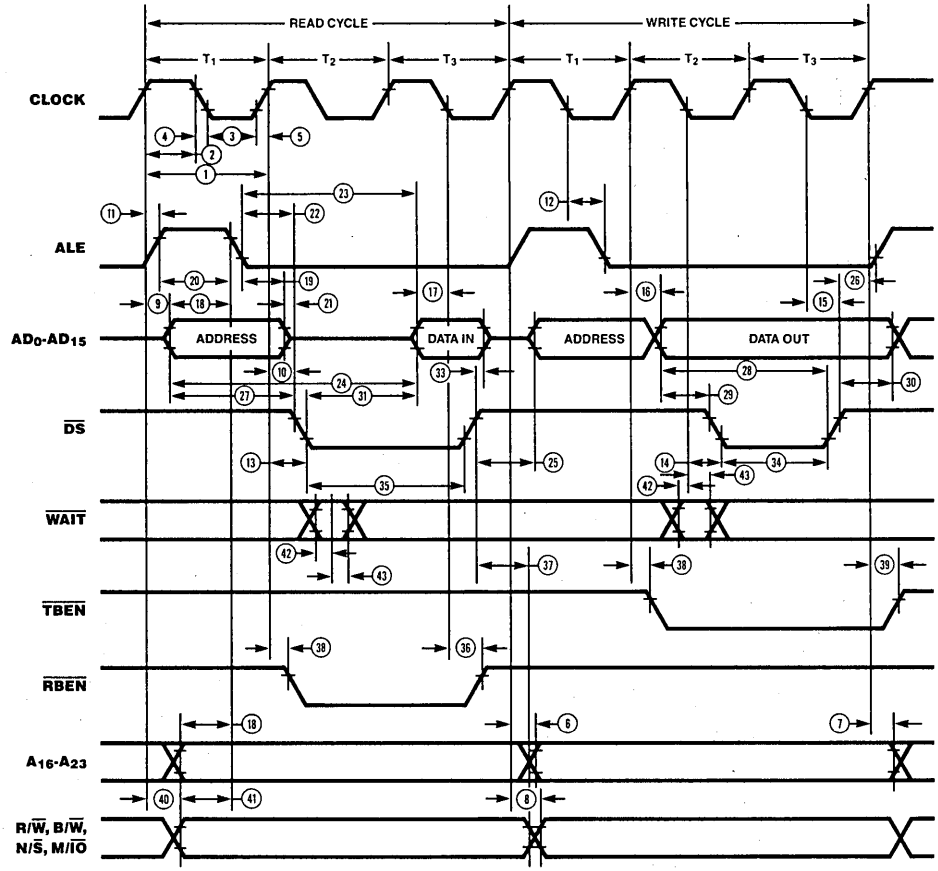


Figure 28. AC Timing when DTC is a Bus Master

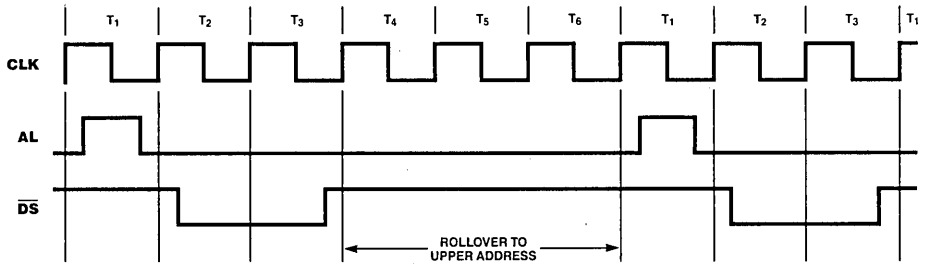


Figure 29. Upper Address Rollover Timing

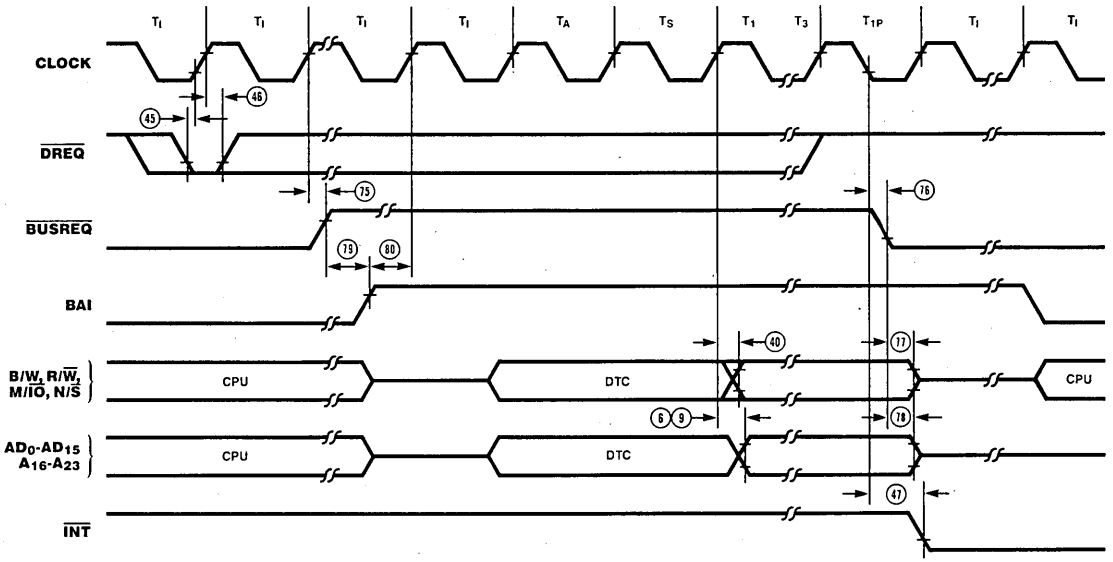


Figure 30. Bus Exchange Timing

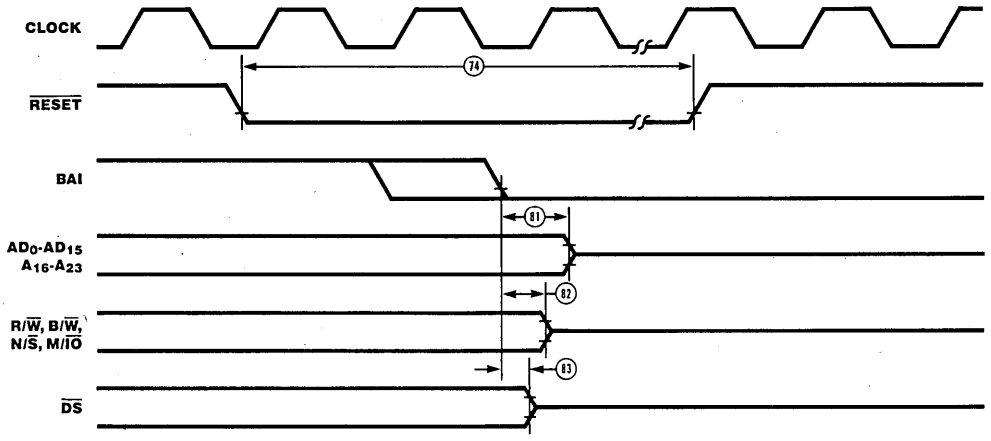


Figure 31. Reset Timing

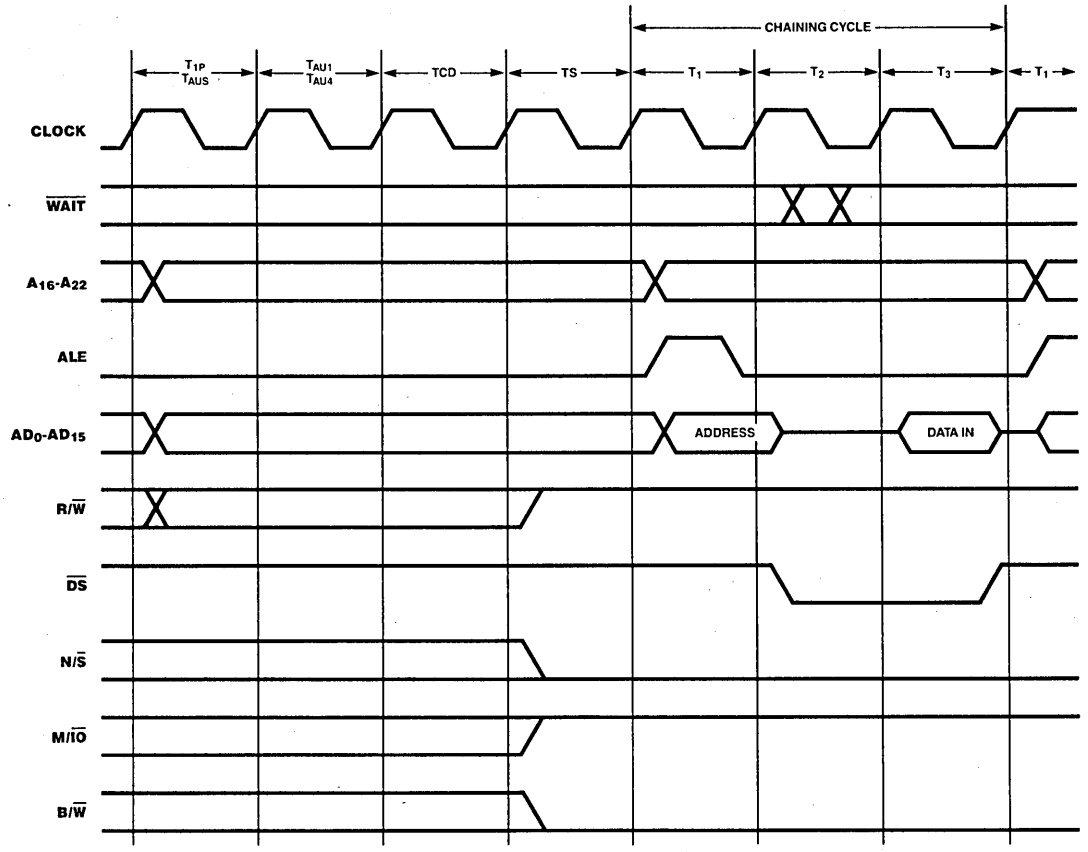
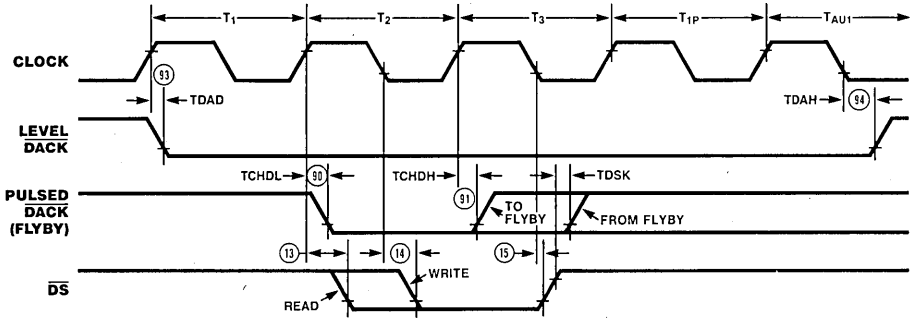
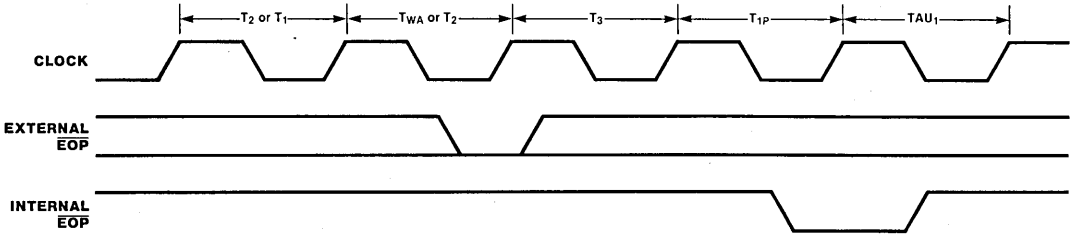


Figure 32. Timing During Chaining

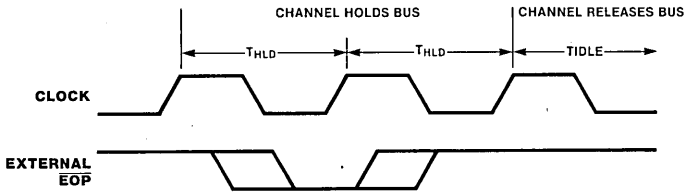


NOTES: 1. State T_{1P} is a pseudo-T_1 state, without active $\overline{AS}$ generated following termination of any DMA operation.
2. State T_{AU1}, is an auto-initialization state generated following the TC, MC, or EOP termination.
3. Level DACK Rising Edge occurs as shown if auto-reloading is not programmed, otherwise it stays Low for three additional clock cycles.

Figure 33. DACK Timing



a) $\overline{EOP}$ SAMPLING AND GENERATION DURING DMA OPERATIONS



b) SAMPLING OF $\overline{EOP}$ DURING BUS HOLD

Notes:
1. The diagram lists state names for both I/O and memory accesses. Sampling of $\overline{EOP}$ will occur on the falling edge of state T_3.
2. State T_{1P} is a pseudo-T_1 state, without active $\overline{AS}$ generated following termination of any DMA operation.
3. State TAU_1 is an auto-initialization state generated following the TC, MC, or $\overline{EOP}$ termination.

Figure 34. EOP Timing

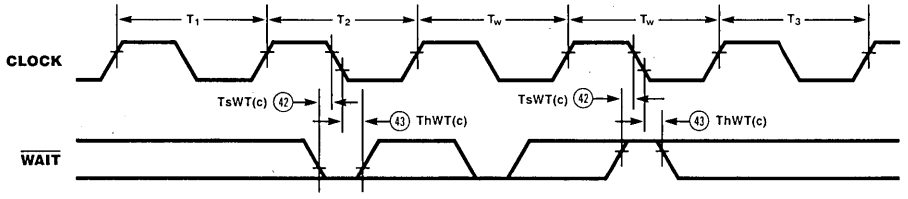


Figure 35. WAIT Timing

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . −0.3 V to +7.0 V
Operating Ambient
Temperature . . . . . .See ordering information
Storage Temperature . . . . . −65 °C to +150 °C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

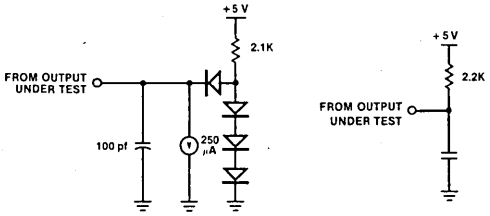
STANDARD TEST CONDITIONS

The DC Characteristics and Capacitance sections listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard condtions are as follows:

- $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- GND = 0V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pf maximum.



DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{CH}	Clock Input High Voltage	$V_{CC}-0.4$	$V_{CC}+0.3$	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10	µA	$0.4 \leqslant V_{IN} \leqslant V_{CC}$
I_{OL}	Output Leakage		±10	µA	$0.4 \leqslant V_{IN} \leqslant +V_{CC}$
I_{CC}	V_{CC} Supply Current		350	mA	$T_A = 0°C$

NOTE: V_{CC} = 5V ± 5% unless otherwise specified.

CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C_{CLOCK}	Clock		40	pf
C_{IN}	Input		5	pf
C_{OUT}	Output		10	pf

NOTES:
T_A = 25°C, f = 1 MHz.
Unmeasured pins returned to ground.

AC CHARACTERISTICS†

DTC AS BUS MASTER

Number	Symbol	Parameter	4MHz Min(ns)	4MHz Max(ns)	6MHz Min(ns)	6MHz Max(ns)
1	TcC	Clock Cycle Time	250	2000	165	2000
2	TwCh	Clock Width (High)	105		70*	
3	TwCl	Clock Width (Low)	105		70*	
4	TfC	Clock Time		20		10
5	TrC	Clock Rise Time		20		15
6	TdC(AUv)	Clock ↑ to Upper Address (A_{16}-A_{23}) Valid Delay		90		80
7	ThC(AUv)	Clock ↑ to Upper Address Valid Hold Time	20		10	
8	TdC(ST)	Clock ↑ to R/$\overline{W}$ and B/$\overline{W}$ Valid Delay		110		90
9	TdC(A)	Clock ↑ to Lower Address (A_0-A_{15}) Valid Delay		90		90
10	TdC(Az)	Clock ↑ to Lower Address (A_0-A_{15}) Float Delay		60		60
11	TdC(Alr)	Clock ↑ to ALE ↑ Delay		70		60
12	TdC(AL)	Clock ↓ to ALE ↓ Delay		70		60
13	TdC(DS)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		60		60
14	TdC(DSf)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		60		60
15	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		60		60
16	TdC(DO)	Clock ↑ to Data Out Valid Delay		90		90
17	TsDI(C)	Data In to Clock ↓ Setup Time	20		15	
18	TdA(AL)	Address Valid to ALE ↓ Delay	50		35	
19	ThAL(A)	ALE ↓ to Lower Address Valid Hold Time	60		40	
20	TwAL	ALE Width (High)	80		60	
21	TdAz(DS)	Lower Address Float to $\overline{DS}$ ↓ Delay	0		0	
22	TdAL(DS)	ALE ↓ to $\overline{DS}$ (Read) ↓ Delay	75		35	
23	TdAL(DI)	ALE ↓ to Data In Required Valid Delay		300		215
24	TdA(DI)	Address Valid to Data In Required Valid Delay		410		305
25	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	80		45	
26	TdDS(AL)	$\overline{DS}$ ↑ to ALE ↑ Delay	75		40	
27	TdA(DS)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	160		110	

*These must not occur simultaneously.
†Units in nanoseconds.

AC CHARACTERISTICS†

DTC AS BUS MASTER (Continued)

Number	Symbol	Parameter	4MHz Min(ns)	4MHz Max(ns)	6MHz Min(ns)	6MHz Max(ns)
28	TdDO(DSr)	Data Out Valid to $\overline{DS}$ ↑ Delay	230		200	
29	TdDO(DSf)	Data Out Valid to $\overline{DS}$ ↓ Delay	55		35	
30	ThDS(DO)	$\overline{DS}$ ↑ to Data Out Valid Hold Time	85		45	
31	TdDS(DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid Delay		205		155
33	ThDI(DS)	$\overline{DS}$ ↑ to Data In Hold Time	0		0	
34	TwDSmw	$\overline{DS}$ (Write) Width (Low)	185		110	
35	TwDSmr	$\overline{DS}$ (Read) Width (Low)	275		220	
36	TdC(RBr)	Clock ↓ to $\overline{RBEN}$ ↑ Delay‡		70		65
37	ThDS(ST)	$\overline{DS}$ ↑ to B/$\overline{W}$, N/S, R/$\overline{W}$ and M/$\overline{IO}$ Valid Hold Time	75		45	
38	†TdC(TRf)	Clock ↑ to $\overline{TBEN}$ or $\overline{RBEN}$ ↓ Delay		60		60
39	TdC(TRr)	Clock ↑ to $\overline{TBEN}$ ↑ Delay		60		60
40	TdC(ST)	Clock ↑ to M/$\overline{IO}$ and N/$\overline{S}$ Valid Delay		90		75
41	TdS(AL)	R/$\overline{W}$, M/$\overline{IO}$, B/$\overline{W}$ and N/$\overline{S}$ Valid to ALE ↓ Delay	60		35	
42	TsWT(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	20		20	
43	ThWT(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	20		20	
44	TwDRQ	$\overline{DREQ}$ Pulse Width (Single Transfer Mode)	20		20	
45	TsDRQ(C)	$\overline{DREQ}$ Valid to Clock ↑ Setup Time	60		50	
46	ThDRQ(C)	Clock ↑ to $\overline{DREQ}$ Valid Hold Time	20		20	
47	TdC(INTf)	Clock ↓ to $\overline{INT}$ ↓ Delay		150		150

†Units in nanoseconds.
‡Parameter 36 is slower than parameter 15.

AC CHARACTERISTICS†

DTC AS BUS SLAVE BUS EXCHANGE

Number	Symbol	Parameter	4MHz Min(ns)	4MHz Max(ns)	6MHz Min(ns)	6MHz Max(ns)
61	TdIN(DO)	$\overline{INTACK}$ ↓ to Data Output Valid Delay		135		135
62	TdIN(DOz)	$\overline{INTACK}$ ↑ to Data Output Float Delay		80		80
63	TdDS(DO)	$\overline{DS}$ ↓ (IOR) to Data Output Driven Delay		135*		135*
64	TdDS(DOz)	$\overline{DS}$ ↑ (IOR) to Data Output Float Delay		80		80
65	TsDI(DS)	Data Valid to $\overline{DS}$ ↑ (IOW) Setup Time	40		40	
66	ThDS(DI)	$\overline{DS}$ ↑ (IOW) to Data Valid Hold Time	40		30	
67	TwDS	$\overline{DS}$ Low Width	150*		150*	
68	TwIN	$\overline{INTACK}$ Low Width	150		150	
69	ThDS(CS)	$\overline{DS}$ ↑ to $\overline{CS}$ Valid Hold Time	20		20	
70	ThDS(PD)	$\overline{DS}$ ↑ to $\overline{P/D}$ Valid Hold Time	20		20	
71	TsPD(DS)	$\overline{P/D}$ Valid to $\overline{DS}$ ↓ Setup Time (IOR) $\overline{P/D}$ Valid to $\overline{DS}$ ↓ Setup Time (IOW)	10 50		10 50	

*2000ns for slow readable registers (worst case)
†Units in nanoseconds.

AC CHARACTERISTICS†

DTC AS BUS SLAVE BUS EXCHANGE (Continued)

Number	Symbol	Parameter	4MHz Min(ns)	4MHz Max(ns)	6MHz Min(ns)	6MHz Max(ns)
72	TsCS(DS)	$\overline{CS}$ Valid to $\overline{DS}$ ↓ Setup Time	30		30	
73	TrDS	$\overline{DS}$ ↑ to $\overline{DS}$ ↓ Recovery Time (for Commands Only)	4TcC		4TcC	
74	TwRST	$\overline{RESET}$ Low Width	3TcC		3TcC	
75	TdC(BRQf)	Clock ↑ to BREQ ↑ Delay		150		150
76	TdC(BRQr)	Clock ↓ to BREQ ↓ Delay		165		150
77	TdBRQ(CTRz)	BUSREQ ↓ to Control Bus Float Delay		140		140
78	TdBRQ(ADz)	BUSREQ ↓ to AD Bus Float Delay		140		140
79	TdBRQ(BAI)	BUSREQ ↑ to BAI ↑ Required Delay	0		0	
80	TsBAI(C)	BAI Valid to Clock ↑ Setup Time	50		45	
81	TdBAI(ADz)	BAI ↓ to A and AD Buses Float Delay (Reset)		135		135
82	TdBAI(CTRz)	BAI ↓ to Control Bus Float Delay (Reset)		100		100
83	TdBAI(DSz)	BAI ↓ to $\overline{DS}$ Float Delay (Reset)		90		90
84	TsRW(DS)	R/$\overline{W}$ Valid to $\overline{DS}$ ↓ Setup Time (IOW)	2		2	
85	ThDS(RW)	$\overline{DS}$ ↑ to R/$\overline{W}$ Valid Hold Time (IOW)	-10		-10	
86	TsRW(DS)	R/$\overline{W}$ Valid to $\overline{DS}$ ↓ Setup Time (IOR)	20		20	
87	ThDS(RW)	$\overline{DS}$ ↑ to R/$\overline{W}$ Valid Hold Time (IOR)	20		20	

*2000ns for slow readable registers (worst case)
†Units in nanoseconds.

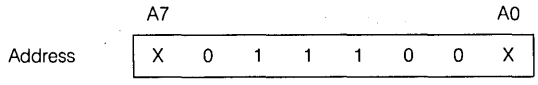
AC CHARACTERISTICS†

DTC-PERIPHERAL INTERFACE

Number	Symbol	Parameter	4MHz Min(ns)	4MHz Max(ns)	6MHz Min(ns)	6MHz Max(ns)
90	TCHDL	Clock ↑ to Pulsed $\overline{DACK}$ ↓ Delay (Flyby Transactions Only)		100		85
91	TCHDH	Clock ↑ to Pulsed $\overline{DACK}$ ↑ Delay (To Flyby Transactions Only)		100		85
92	TDSK	DS ↑ to Pulsed $\overline{DACK}$ ↑ Delay (From Flyby Transactions Only)	10		10	
93	TDAD	Clock ↑ To Level $\overline{DACK}$ Valid Delay		100		85
94	TDAH	Clock ↓ to Level $\overline{DACK}$ Valid Hold Time		100		85
95	TEIDL	Clock ↓ to Internal $\overline{EOP}$ Low Delay		110		90
96	TEIDH	Clock ↓ to Internal $\overline{EOP}$ ↑ Delay		110		90
97	TES	External $\overline{EOP}$ Valid to Clock ↓ Setup Time During Operation	10		10	
98	TEW	External $\overline{EOP}$ Pulse Width Required During Operation	20		20	
99	TES(BH)	External $\overline{EOP}$ Valid to Clock ↓ Setup Time During Bus Hold	10		10	
100	TEW(BH)	External $\overline{EOP}$ Pulse Width Required During Bus Hold	20		20	

†Units in nanoseconds.

Appendix A



Fast Readable
Writeable

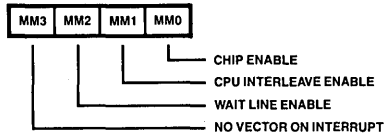


Figure A1. Master Mode Register

	A7							A0		
Address	X	0	1	1	0	0	1	X	Current Operation Count	CH1
	X	0	1	1	0	0	0	X	Current Operation Count	CH2
	X	0	1	1	0	1	1	X	Base Operation Count	CH1
	X	0	1	1	0	1	0	X	Base Operation Count	CH2
	X	1	0	0	1	0	1	X	Pattern	CH1
	X	1	0	0	1	0	0	X	Pattern	CH2
	X	1	0	0	1	1	1	X	Mask	CH1
	X	1	0	0	1	1	0	X	Mask	CH2

Chain Loadable
Writeable
Pattern and Mask—Slow Readable
Operation Count—Fast Readable

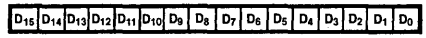


Figure A2. Miscellaneous Registers

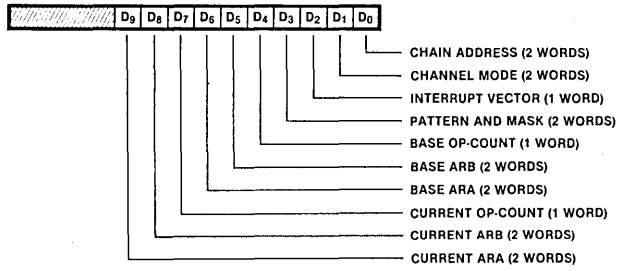


Figure A3. Chain Control Register

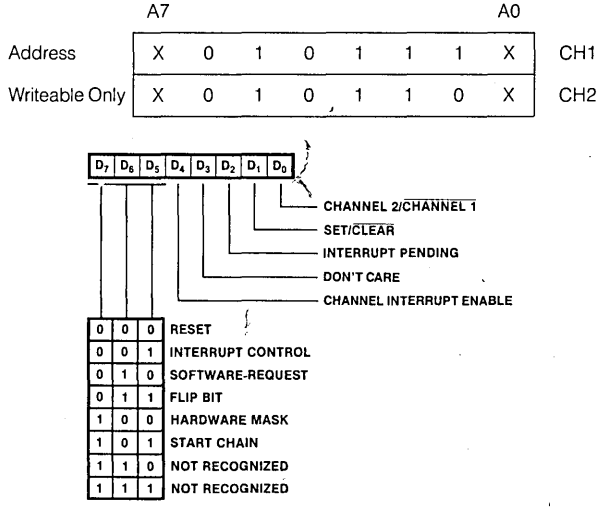


Figure A4. Command Register

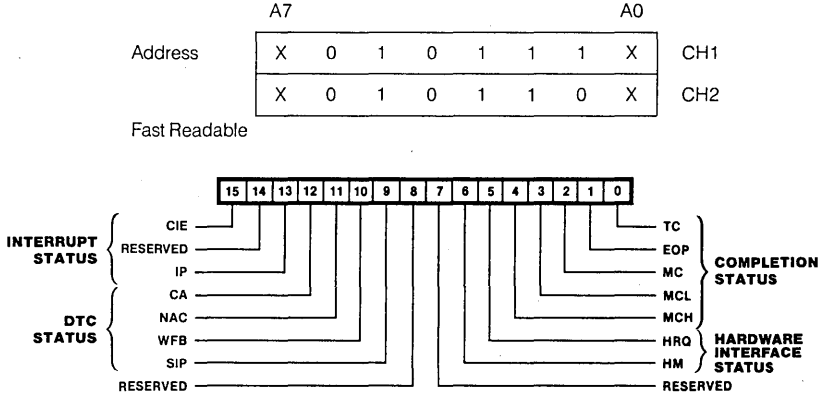


Figure A5. Status Registers

	A7							A0	
Address	X	0	1	0	1	0	1	X	CH1
	X	0	1	0	1	0	0	X	CH2

Fast Readable

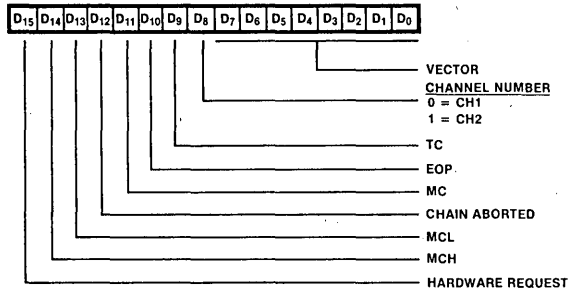


Figure A6. Interrupt Save Registers

	A7							A0	
	X	1	0	1	0	1	1	X	High CH1
	X	1	0	1	0	1	0	X	High CH2
	X	1	0	1	0	0	1	X	Low CH1
	X	1	0	1	0	0	0	X	Low CH2

Chain Loadable
Writeable (Lower 16 bits)
Slow Readable

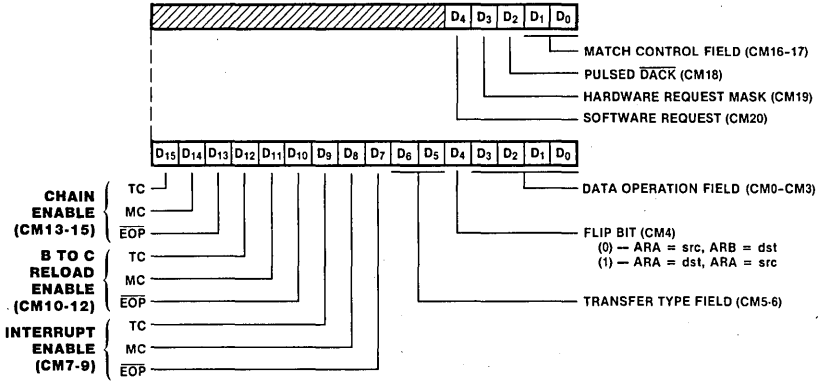


Figure A7. Channel Mode Register

Address	A7							A0	
	X	0	1	0	0	1	1	X	Up-Addr CH1
	X	0	1	0	0	1	0	X	Up-Addr CH2
	X	0	1	0	0	0	1	X	Low-Addr CH1
	X	0	1	0	0	0	0	X	Low-Addr CH2

Fast Readable

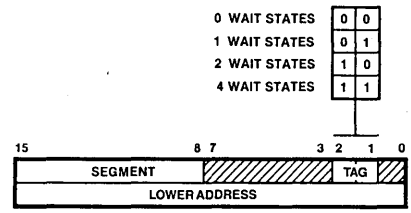


Figure A8. Chain Address Register

Address	A7							A0		
	X	0	0	1	1	0	1	X	Current Address Register A Up-Addr/Tab	CH1
	X	0	0	1	1	0	0	X	Current Address Register A Up-Addr/Tag	CH2
	X	0	0	0	1	0	1	X	Current Address Register A Low-Addr	CH1
	X	0	0	0	1	0	0	X	Current Address Register A Low-Addr	CH2
	X	0	0	1	0	0	1	X	Current Address Register B Up-Addr/Tag	CH1
	X	0	0	1	0	0	0	X	Current Address Register B Up-Addr/Tab	CH2
	X	0	0	0	0	0	1	X	Current Address Register B Low-Addr	CH1
	X	0	0	0	0	0	0	X	Current Address Register B Low-Addr	CH2
	X	0	0	1	1	1	1	X	Base Address Register A Up-Addr/Tag	CH1
	X	0	0	1	1	1	0	X	Base Address Register A Up-Addr/Tag	CH2
	X	0	0	0	1	1	1	X	Base Address Register A Low-Addr	CH1
	X	0	0	0	1	1	0	X	Base Address Register A Low-Addr	CH2
	X	0	0	1	0	1	1	X	Base Address Register B Up-Addr/Tag	CH1
	X	0	0	1	0	1	0	X	Base Address Register B Up-Addr/Tag	CH2
	X	0	0	0	0	1	1	X	Base Address Register B Low-Addr	CH1
	X	0	0	0	0	1	0	X	Base Address Register B Low-Addr	CH2

Chain Loadable
Fast Readable and Writeable

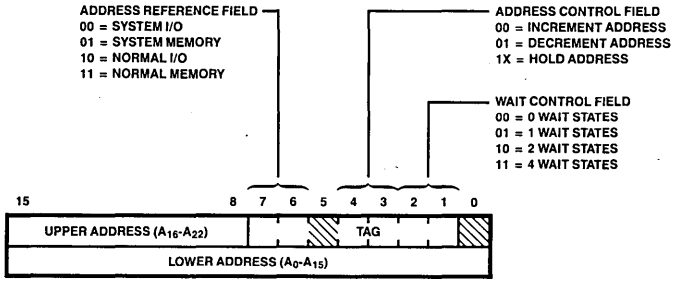


Figure A-9. Address Registers

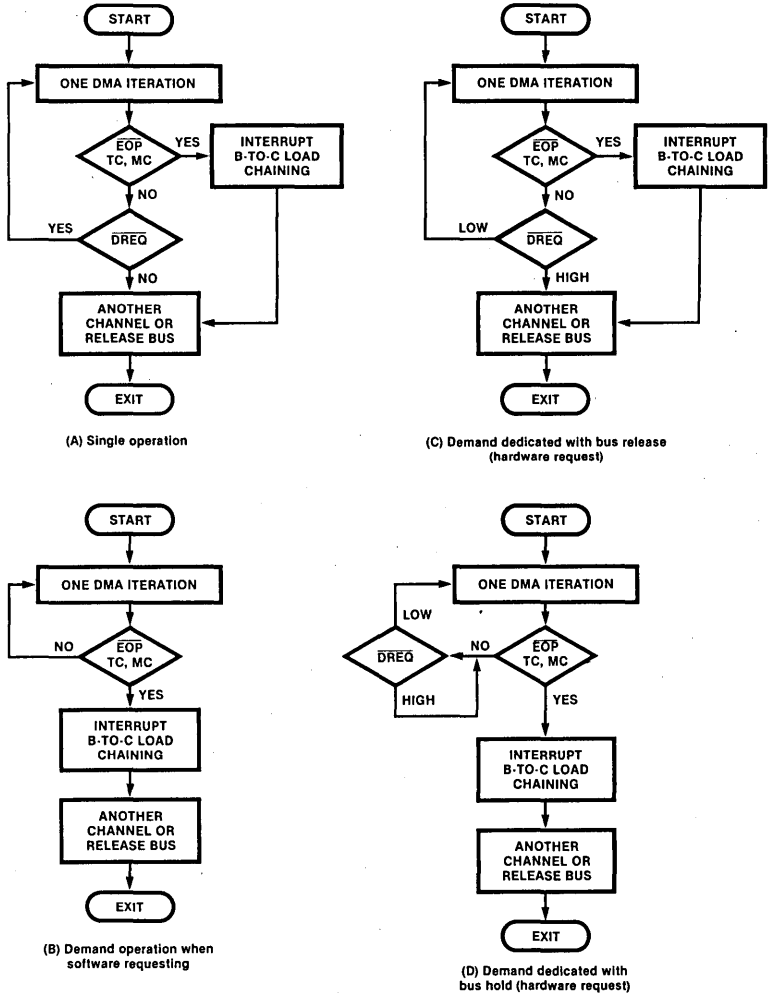


Figure A-10. Basic DMA Operations of Z8516/Z9516 DTC

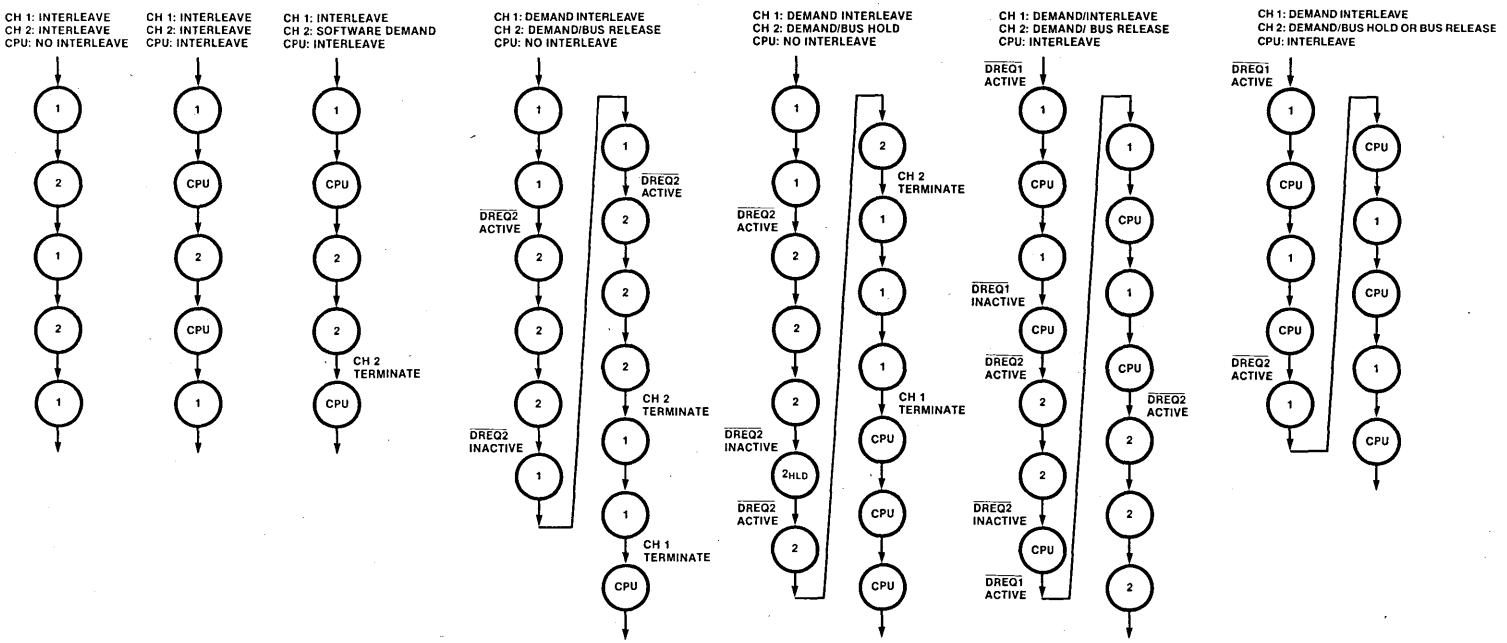


Figure A-11. Demand Interleave Operations of Z8516/Z9516 DTC

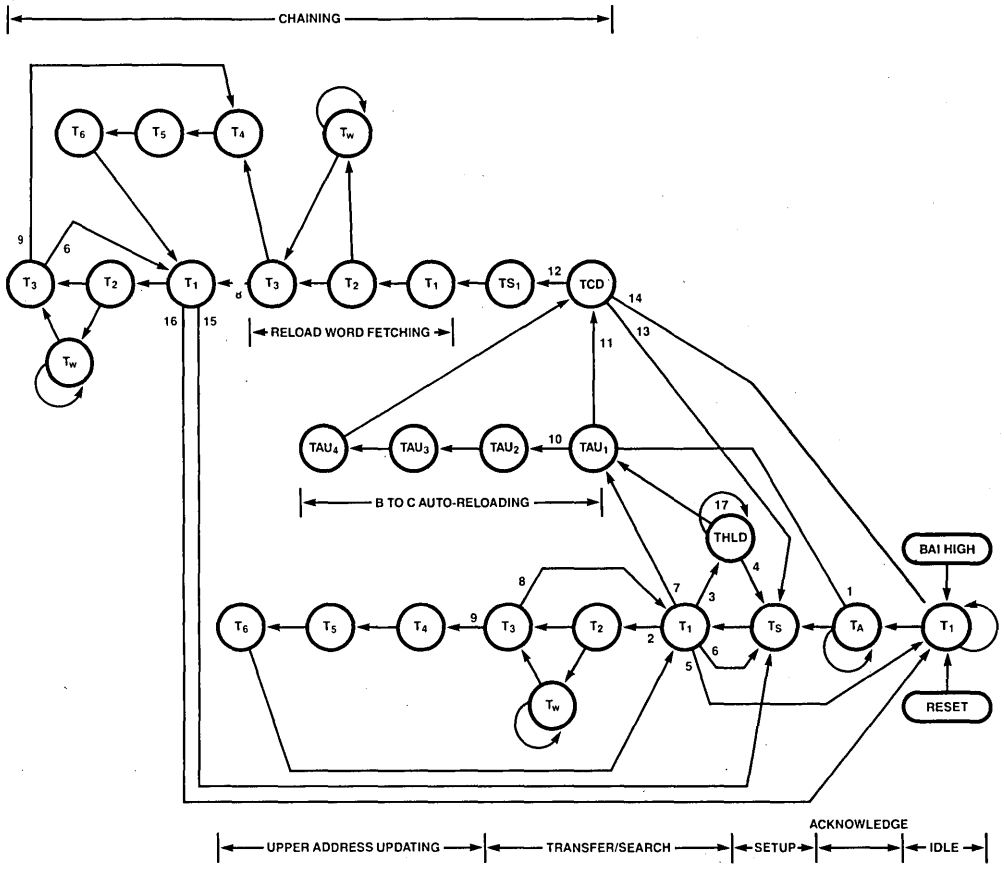


Figure A-12. Z8516/Z9516 State Diagram

Z8516/Z9516 INTERNAL OPERATION ROUTINES

1. "Start Chain" command issued or start updating routine, including base-to-current auto-reloading and chaining, after an interrupt has been served.
2. Normal DMA operation.
3. Demand with Bus hold while DREQ is inactive.
4. DREQ is active while bus held.
5. Single transfer, CPU interleave enabled, or demand with bus release while current DREQ is inactive and no DMA request is pending.
6. Single Transfer or Demand/Bus release while current DREQ is inactive, but the other DMA request is pending.
7. TC, MC, or EOP termination occurs.
8. On DMA or chain transaction is done and the upper address is not changed.
9. One DMA or chain transaction is done and the upper address is changed.
10. Base-to-current auto-reloading is enabled.
11. Base-to-current auto-reloading is disabled.
12. Chaining is enabled.
13. Chaining is disabled and another DMA request is pending.
14. Chaining is disabled and no DMA request is pending.
15. Chaining ends and another DMA request is pending.
16. Chaining ends and no DMA request is pending.
17. EOP termination of Bus Hold.

NOTE: When a second interrupt is issued before the first interrupt is acknowledged, the Status register's SIP bit is set and the channel relinquishes the bus until the first interrupt is serviced. If the channel performs the updating routine, once the SIP bit is cleared, the DTC reacquires the bus and performs the operation.

Z8581 Clock Generator and Controller

October 1988

FEATURES

- Two independent 20 MHz oscillators generate two 10 MHz clock outputs and one 20 MHz clock output.
- Oscillator input frequency sources can be either crystals or external oscillators.
- Outputs directly drive the Z80, Z8000, 8086, 8088, and 68000 microprocessor clock inputs.
- Can be used as a general-purpose clock generator.
- 18-pin slimline package used; single +5V dc power required.
- Provides ability to stretch High and/or Low phase of clock signal under external control.
  - On-chip 2-bit counter can be used to selectively stretch clock cycles.
- On-chip reset logic
  - Reset output is synchronized with System Clock output.
  - Power-up reset period is maintained for a minimum of 30 ms.
  - External input initiates system reset.

GENERAL DESCRIPTION

The Z8581 Clock Generator and Controller is a versatile addition to Zilog's family of Universal microprocessor components. The selective clock-stretching capabilities and variety of timing outputs produced by this device allow it to easily meet the timing design requirements of systems with microprocessors and LSI peripherals. The clock output drivers of the Z8581 also meet the non-TTL voltage requirements for driving NMOS clock inputs with no additional external components. The Z8581 provides an elegant, single-chip solution to the design of system clocks for microprocessor-based products.

The Z8581 oscillators are referenced as the system clock oscillator and the general-purpose clock oscillator. Both oscillators are driven by external crystals or other frequency sources.

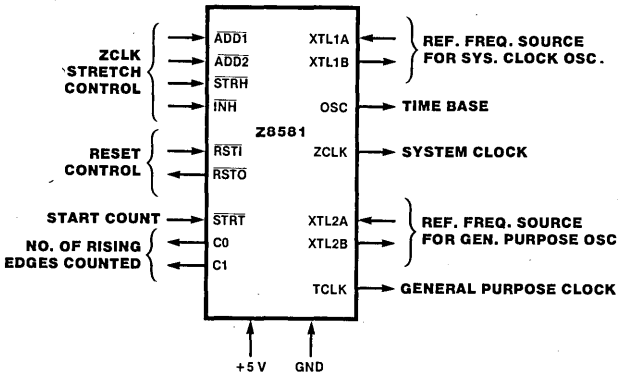


Figure1. Pin Functions

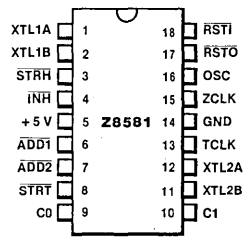


Figure 2. Pin Assignments

PIN DESCRIPTIONS

Figures 1 and 2, respectively, show the pin functions and assignments of the Z8581. Tie unused inputs High through a resistor.

ADD1, ADD2. *Add Delay 1* (input, active Low) and *Add Delay 2* (input, active Low). These signals control the addition of one, two, or three delay periods to a selected half-cycle of the ZCLK output.

C0, C1. *ZCLK Count 0* (output, active High) and *ZCLK Count 1* (output, active High). These signals indicate, in binary, the number or rising edges of ZCLK that have occurred after the assertion of the STRT input.

INH. *Inhibit Delay* (input, active Low). When asserted, this signal inhibits the functions of inputs ADD1 and ADD2.

OSC. *Time Base Clock* (output, active High). This signal provides a TTL-compatible clock output at the same frequency as the system clock frequency source.

RSTI. *Reset In* (input, active Low). When asserted, this signal indicates a reset condition and initiates the assertion of RSTO synchronized with ZCLK.

RSTO. *Reset Out* (output, active Low). When asserted, this signal indicates that a system reset condition is required, either by RSTI going Low or by a system powerup condition.

STRT. *Start Count* (input, negative edge-triggered). When asserted, this signal resets a two-bit binary counter and then enables the counter to count the rising edges of the ZCLK output.

STRH. *Delay ZCLK* (input, active Low). When asserted, this signal causes the current half-cycle of the ZCLK output to be delayed (stretched) for as long as STRH is held Low. This control input overrides the ADD1, ADD2, and INH functions.

TCLK. *General-Purpose Clock* (output, MOS-compatible, active High). This signal is the timing output of the general-purpose oscillator. TCLK's frequency is half that of the external oscillator used to drive the general purpose oscillator.

XTAL1A, XTAL1B. *System Clock Frequency Source A* (input, active High) and System Clock Frequency Source B (output, active High). These signals are used by the external oscillator to drive the internal system clock oscillator and the OSC output.

XTAL2A, XTAL2B. *General-Purpose Clock Frequency Source A* (input, active High) and *General-Purpose Clock Frequency Source B* (output, active High). These signals are used by the external oscillator to drive the internal general-purpose clock oscillator.

ZCLK. *System Clock* (output, MOS-compatible, active High). This signal is the timing output of the system clock oscillator. This clock can be modified by the delay (stretch) control inputs. Its frequency, when unmodified, is half that of the external system clock frequency source.

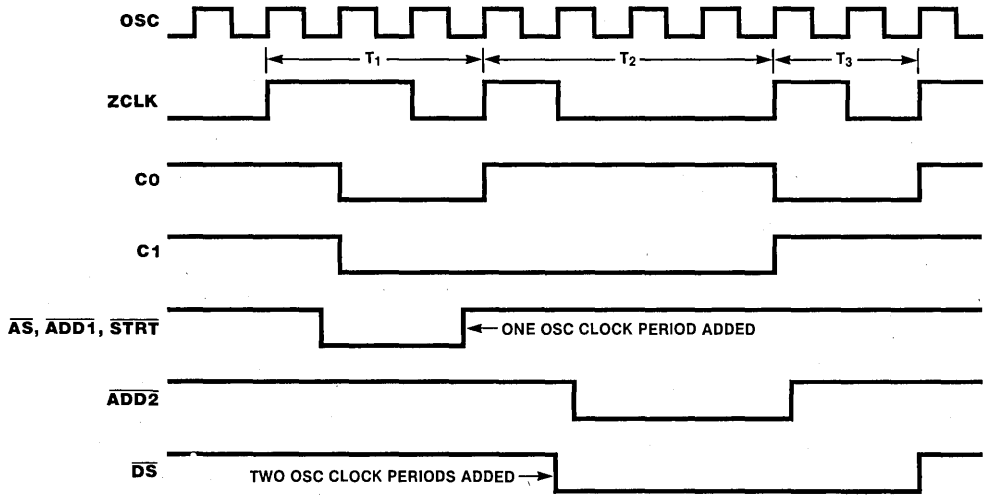


Figure 3. Timing Diagram Stretching Z8000 AS and DS

System Clock Oscillator

The timing outputs provided by this oscillator consist of a Time Base output (OSC), at the frequency of the reference source, and a stretchable System Clock output (ZCLK), at a frequency determined by the stretch control inputs. An on-chip TTL driver at OSC and an NMOS driver at ZCLK eliminate the need for external buffers or drivers. The NMOS drivers can drive 200 pf loads with output rise and fall times of 10 ns maximum.

ZCLK can be stretched under program or hardwired control by selectively adding periods equivalent to a full OSC cycle to either the High or Low portion of a clock cycle. One, two, or three periods can be added to double, triple, or quadruple the duration of the selected ZCLK half-cycle. Adding periods to ZCLK is a function of the ADD1 and ADD2 inputs. These active Low inputs are sampled prior to the rising edge of signal OSC; their sampled status represents the number of periods to be added to ZCLK.

Two additional control inputs, $\overline{INH}$ and $\overline{STRH}$, affect the stretch function. Input $\overline{INH}$, when asserted, inhibits the function of $\overline{ADD1}$ and $\overline{ADD2}$. Input $\overline{STRH}$ stretches the ZCLK output for as long as it is asserted (Low); it overrides all other stretch control inputs.

Table 1 summarizes the functions performed by the stretch control inputs.

The system clock oscillator also contains a 2-bit ZCLK counter. This counter, when initialized by the assertion of $\overline{STRT}$, counts the next four rising edges of the ZCLK output. The current count is presented on outputs C0 and C1. This counter and its outputs enable the user to determine the occurrence (rising edge) of each of four clocks after a specific event ($\overline{STRT}$ is asserted). This facility can, for example, be used to determine when a delay is to be inserted into a CPU machine cycle when $\overline{STRT}$ is triggered by either an $\overline{M1}$ (Z80) or an $\overline{AS}$ (Z8000) input signal.

The clock stretch capability allows systems to run at the nominal high speed of ZCLK, except during cycles that require more time than usual to complete a transaction. For example, extended access time may be required in accessing certain areas of memory, in accessing I/O devices, or in other CPU/Peripheral transactions. Figures 3 and 4 illustrate, respectively, the circuit configuration and timing required to stretch the Z8000 Address Strobe ($\overline{AS}$) and Data Strobe ($\overline{DS}$) to allow more time for address functions and to enable the CPU to operate with memories that have a relatively long access time.

In addition, the ZCLK stretch control logic can be hardwired to meet various duty cycle requirements. For example, a simple hardwired connection can cause every other ZCLK cycle to be stretched to produce a ZCLK output with a 33% duty cycle.

The system clock oscillator also provides a system reset output ($\overline{RSTO}$) that is synchronized with ZCLK. This output is controlled by a system reset input ($\overline{RSTI}$) during normal system reset operations and by delay circuitry in the system clock oscillator during power-up operations. During a normal system reset operation, a Low on $\overline{RSTI}$ causes $\overline{RSTO}$ to be asserted (Low) on the next rising edge of ZCLK. Output $\overline{RSTO}$ is held Low for a period of 16 ZCLK clock cycles (the required reset time for both the Z80 and Z8000 CPU system reset functions). During a power-up operation, $\overline{RSTO}$ is asserted for a minimum of 30 ms after power is turned on (the time required for both the Z80 and Z8000 power-up functions).

Table 1. Stretch Control Functions

STRH	INH	ADD2	ADD1	Periods Added
0	X	X	X	Unlimited
1	0	X	X	0
1	1	0	0	3
1	1	0	1	2
1	1	1	0	1
1	1	1	1	0

NOTES: X = Don't Care, 1 = High, 0 = Low

General-Purpose Oscillator

This oscillator provides a fixed frequency General-Purpose Clock output (TCLK) at half its source frequency. This output is useful for system timing functions such as controlling a baud rate generator. Output TCLK can also be used as the frequency reference source for the system clock oscillator.

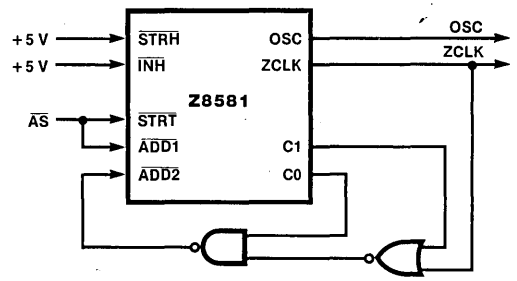


Figure 4. Configuration for Stretching Z8000 Address (AS) and Data (DS) Strobes

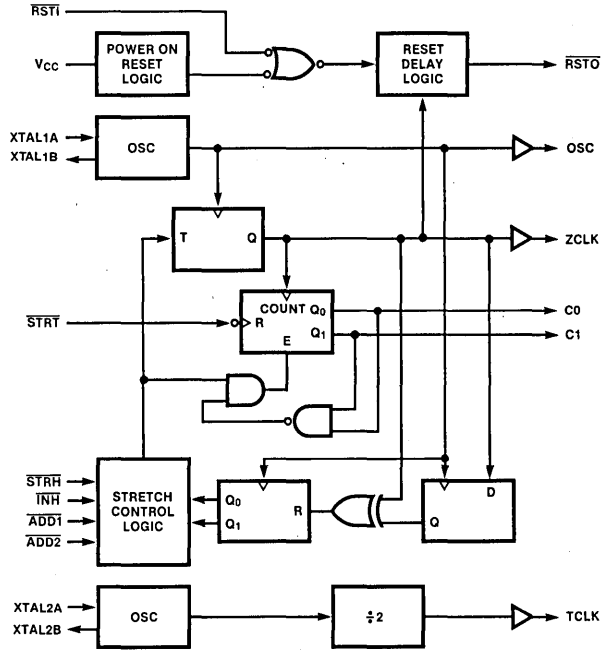
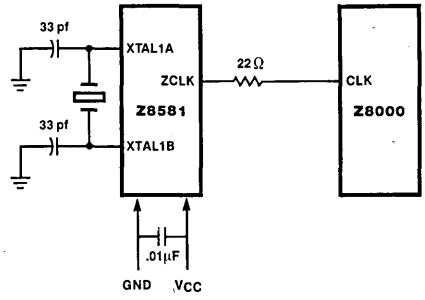


Figure 5. Z8581 Functional Block Diagram

SYSTEM INTERFACE CONSIDERATIONS

Due to the fast rise and fall times produced by the Z8581, transmission line concepts must be applied in order to avoid ringing and reflections on the clock outputs. More specifically, the interconnections between the clock outputs and the loads they are driving must be treated as transmission lines, and it is necessary to match the source impedance of the clock outputs to the characteristic impedances of these transmission lines. In most cases the impedances can be matched by placing termination resistors in series with the clock outputs. These resistors range in value from 22 to 220 ohms, with the value chosen to optimize the clock risetime at the load. **(See Figure 6.)** It is important to control the impedance seen by the clock output by keeping leads short and avoiding stray inductances wherever possible.

Another important consideration is the bypass capacitor. To avoid distortion of the power supply, the Z8581 requires a high frequency 0.01 μF ceramic capacitor between V_{CC} and ground, and the leads connecting this capacitor to the pins should be kept as short as possible.



NOTE: The Z8581 requires a parallel-resonant fundamental type crystal. The capacitor may be varied to fine tune the frequency.

Figure 6. Z8581/Z8000 Interface

SERIES RESONANCE AND PARALLEL RESONANCE

Manufacturers of crystals specify crystals as either series-resonant or parallel-resonant **(Figure 7.)**.

At the frequency of series resonance:

$$fs = \frac{1}{2\pi\sqrt{L*C}}$$

When the magnitude of X_C (reactance of C) and X_L (reactance of L) are equal, one effectively cancels the other, resulting in an equivalent of R shunted by Cs. If R is very small compared to X_{Cs}, series resonance is indicated by minimum impedance and zero phase shift. The series-resonant crystals are resistive and produce an output in phase with the input.

At frequencies slightly higher than series resonance, X_L increases and X_C decreases, resulting in a net inductive reactance, X_L. When $X_L = X_{Cs}$, the result is parallel resonance with the frequency of parallel resonance:

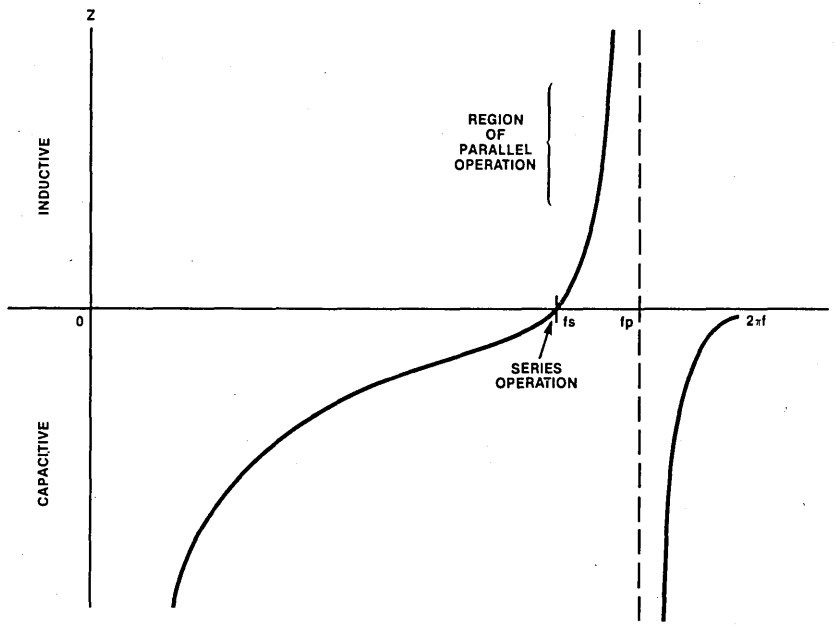
$$fp = \frac{1}{2\pi\sqrt{L*\frac{C*C_s}{C+C_s}}}$$

Parallel resonance is indicated by maximum impedance across the crystal terminals. Parallel-resonant crystals are inductive and produce the output shifted in phase from the input.

If a series crystal is used with a parallel-resonant oscillator, the crystal is forced to operate in the parallel region (and vice versa). The clock frequency produced is shifted a small percentage (about 350 ppm) from the specified series crystal frequency **(Figure 7.)**. Any attempts to fine tune the frequency by changing the value of external capacitance may cause the crystal to stop oscillating.

From the equivalent circuit of the crystal and the expression of fs, it can be seen that the series-resonant frequency of the crystal cannot be changed by reactance across the crystal terminals because there is no connection to the junction of L and C. In the case of parallel resonant frequency, Cs appears in the expression and its effective value can be changed by reactance across the terminals.

The load capacitance (C_L) is the capacitance that the crystal sees at its terminals. In parallel-resonant mode, load capacitance is very important, because C_L in combination with crystal inductance determines the frequency.



fp-fs IS VERY SMALL (APPROXIMATELY 350 PARTS PER MILLION)

Figure 7. Series vs. Parallel Resonance

Crystal selection is based on the oscillator design used to provide the clock to the system. Series-resonant crystals should be used with non-inverting oscillators because series-resonant crystals have no phase shift. Parallel-resonant crystals should be used with inverting oscillators because parallel-resonant crystals have some phase shift due to their inductive nature.

DRIVE LEVEL

Drive level is critical because the crystal can dissipate only limited power (10 mW typical) and still meet all specifications. Overdrive may cause the temperature to rise in the crystal. Further overdrive may cause the quartz to operate in a non-linear region producing a frequency shift and possibly causing permanent damage to the crystal.

SERIES CONFIGURATION

When the internal oscillator is non-inverting, the recommended crystal is series-resonant and the circuit **diagram shown in Figure 8. is used**

Many supposedly series-resonant IC oscillators actually operate below series resonance. C1 is used to compensate for this inductive nature of the series-type oscillator. C1 provides enough capacitive reactance to cancel the inductive shift caused by the IC. This brings the overall frequency back to series resonance.

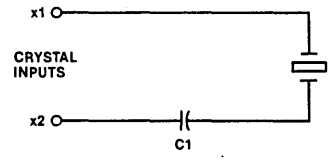


Figure 8. Circuit for Series-Resonant Crystals

PARALLEL CONFIGURATION

When the internal oscillator is inverting, the recommended crystal is parallel-resonant and the circuit diagram **shown in Figure 9. is used.**

The load capacitance, C_L, is determined by the following equation:

$$C_L = \frac{C_1 * C_2}{C_1 + C_2} + C_C$$

where C_C is the parasitic circuit capacitance.

The parallel resonance frequency is determined by the following equation:

$$fp = \frac{1}{2\pi \sqrt{L^* \frac{C^* C_t}{C + C_t}}}$$

where $C_t = C_L + C_S$

Typically, the value of C_L ranges between 20pf and 30pf, and the value of the ratio C1/C2 ranges between 1 and 2. The required frequency can be fine tuned by varying the value of C2. The value of C_L, derived from C1 and C2, depends on the required frequency, characteristics of the oscillator, and the crystal used.

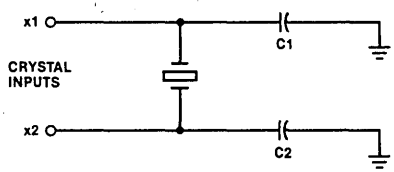


Figure 9. Circuit for Parallel Resonant Crystals

RECOMMENDED CIRCUIT FOR Z8581

Since the internal oscillator of the Zilog Z8581 is inverting, a parallel-resonant type crystal is recommended. The circuit configuration is shown in figure 10. The preferred value of the C_L is about 22pf. The preferred value of C1 and C2 is 33pf, which produces a ratio (C1/C2) of one. C2 can be made variable to fine tune the required frequency.

The output of one oscillator can also be connected to the crystal inputs of the second oscillator, using a 4700 ohms pull-up resistor at the output. (Figure 10.).

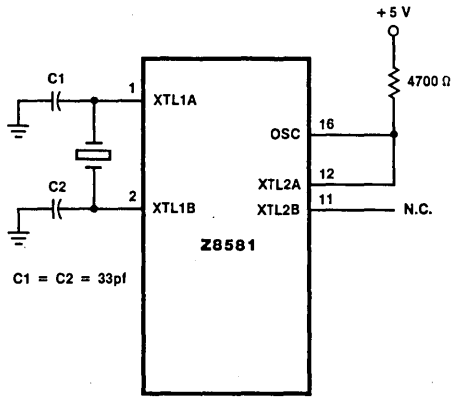


Figure 10. Circuit Configuration for Z8581

SPECIFICATIONS

For proper operation when using Zilog's Z8581, use a fundamental, parallel-type crystal. The following crystal specifications are suggested:

Frequency tolerance: Application dependent
C_L, load capacitance: Approximately 22pf (acceptable range is from 20-30pf)
Rs, equivalent-series resistance: © 150 ohms
Drive level
(for © 10 MHz crystal): 10 milliwatts
(for □ 10 MHz crystal): 5 milliwatts

Holder specifications are user-determined.

For frequencies lower than 4 MHz, the recommended holder type is HC-33/U (0.75"W × 0.765"H, 1.5" lead length with specing of 0.486").

For frequencies higher than 4 MHz, the recommended holder type is HC-18/U (0.435"W × 0.530"H, 1.5" lead length with specing of 0.192").

SUGGESTED VENDORS

The wide variety of applications and frequencies in which crystals are used makes it necessary to custom manufacture all but a very few crystal types. With the crystal specifications given above, most vendors can make the desired crystal even though it may not be a standard off-the-shelf item.

The following two vendors supply crystals to the specifications given above. The user is not limited to these vendors because these suppliers are among many suppliers who might meet your specifications.

Midland-Ross Corporation
NEL UNIT
357 Beloit Street
Burlington, WI 53105
Telephone: (414) 763-3591

CTS KNIGHTS, INC.
400 E. Reimann Ave.
Sandwich, Illinois 60548
Telephone: (815) 786-8411

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature. . . . . .See ordering information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only: operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

The Ordering Information section lists package temperature ranges and product numbers. Refer to the Literature List for additional documentation. Package drawings are in the Package Information section.

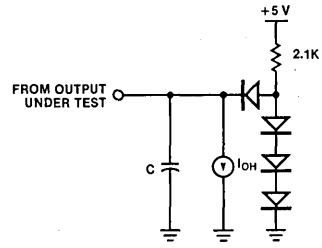
STANDARD TEST CONDITIONS

The DC characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

- S = 0°C to +70°C, $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- E = −40°C to +85°C, $+4.5V \leqslant V_{CC} \leqslant +5.25V$
- M = −55°C to +125°C, $+4.5V \leqslant V_{CC} \leqslant +5.5V$

All ac parameters assume a total load capacitance (C), including parasitic capacitances, of 100 pf max, except for parameters 8, 9, 21, and 22 which are 200 pf max. Timing references between two output signals assume a load difference of 50 pf max.



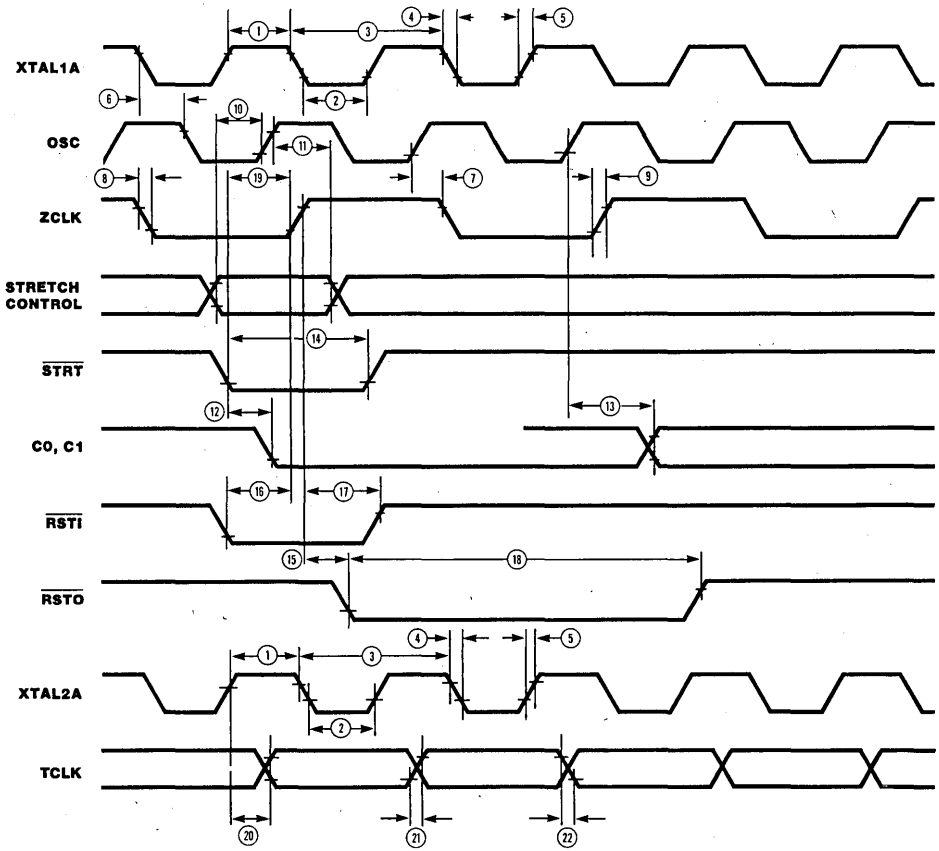
DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC} −3.7	V_{CC} + 0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 μA
V_{OH} (ZCLK, TCLK)	Output High Voltage	V_{CC} − 0.3		V	I_{OH} = −250 μA tested at 5 μs after ZCLK or TCLK rises High
		2.4		V	I_{OH} = −250 μA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = = +2.0 mA
I_{IL}	Input Leakage		±10	μA	$0.4 \leqslant V_{IN} \leqslant +2.4V$
I_{CC}	V_{CC} Supply Current		150	mA	

AC CHARACTERISTICS

Number	Symbol	Parameter	Z8581 6 MHz Min.	Z8581 6 MHz Max.	Z8581-10 10 MHz Min	Z8581-10 10 MHz Max	Units	Notes[1]
1	TwCH	Clock Input High Width	31		18		ns	2
2	TwCL	Clock Input Low Width	31		18		ns	2
3	TpC	Clock Input Cycle Time	82		50		ns	2
4	TfC	Clock Input Fall Time		10		7	ns	2
5	TrC	Clock Input Rise Time		10		7	ns	2
6	TdOSC	Clock Input to OSC Delay		30		20	ns	
7	TdZC	OSC to ZCLK Delay		20		15	ns	
8	TfZC	ZCLK Fall Time		10		10	ns	
9	TrZC	ZCLK Rise Time		10		10	ns	
10	TsSC	Stretch Controls to OSC ↑ Setup	35		20		ns	
11	ThSC	Stretch Controls to OSC ↑ Hold	20		20		ns	
12	Td(ST/CR)	STRT ↓ to 2-bit Counter Reset Delay		35		25	ns	
13	Td(OSC/CC)	OSC ↑ to 2-bit Counter-Change		20		17	ns	3
14	Tw(STRT)	STRT Low Width	50		30		ns	
15	Td(RSTO)	ZCLK ↑ to RSTO ↓ Delay		30		20	ns	
16	Ts(RSTI)	RSTI ↓ to ZCLK ↑ Setup	30		20		ns	
17	Th(RSTI)	RSTI ↓ to ZCLK ↑ Hold	30		20		ns	
18	Tw(RSTO)	RSTO Low Width	16		16		cycles	
19	Ts(ST/ZC)	STRT ↓ to ZCLK ↑ Setup to include ZCLK edge	40		30		ns	
20	TdTC	Clock Input to TCLK Delay		40		30	ns	
21	TrTC	TCLK Rise Time		10		10	ns	
22	TfTC	TCLK Fall Time		10		10	ns	

NOTES: 1. All timings are preliminary and subject to change.
2. Clock input other than a crystal oscillator.
3. Assuming ZCLK rising.



Timing measurements are made at the following voltages:

	High	Low
ZCLK, TCLK	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V

October 1988

Z80,000™ CPU

FEATURES

- Full 32-bit architecture and implementation
- 4G (billion) bytes of directly addressable memory in each of four address spaces
- Linear or segmented address space
- Virtual memory management integrated with CPU
- On-chip cache memory
- General-purpose register file with sixteen 32-bit registers
- Nine general addressing modes
- Numerous data types include bit, bit field, logical value, signed integer, and string
- Regular use of operations, addressing modes, and data types in instruction set
- System and normal modes of operation with separate stacks
- Sophisticated interrupt and trap handling
- Software is a binary-compatible extension of Z8000® software
- Hardware is compatible with other Z-BUS® bus components
- Mainframe performance
- **Hermetic 84 pin grid array package**

GENERAL DESCRIPTION

The Z80,000 CPU is an advanced, high-end 32-bit microprocessor that integrates the architecture of a mainframe computer into a single chip. While maintaining full compatibility with Z8000 family software and hardware, the Z80,000 CPU offers greater power and flexibility in both its architecture and interface capability. Operating systems and compilers are easily developed in the Z80,000 CPU's high-quality environment, and the hardware interface provides for connection to a wide variety of system configurations.

Addresses in the Z80,000 CPU are 32 bits. This allows direct addressing of 4G bytes in each of four address spaces: system-mode data, system-mode instruction, normal-mode data, and normal-mode instruction. The CPU supports three modes of address representation. The 16-bit compact addresses are compatible with Z8000 nonsegmented mode. The 32-bit segmented addresses include both 16-bit offset, which is compatible with Z8000 segmented mode, and 24-bit offset. In addition a full 32-bit linear address space is provided.

The CPU features a general-purpose register file with sixteen 32-bit registers and nine operand addressing modes. The various addressing modes allow encoding choices for compact representation or for full 32-bit addressing. The instruction set can operate on bit, bit field, logical value, signed integer, unsigned integer, address, string, stack, and packed decimal byte data types. Logical and arithmetic instructions operate on bytes (8 bits), words (16 bits) and longwords (32 bits). The Extended Processing Architecture (EPA) supports floating-point operations. In addition, the instruction set is highly regular in combining operations, data types, and addressing modes. High-level language compilation is supported with instructions for procedure linkage, array index calculation, and bounds checking. Other instructions provide operating system functions such as system call and control of memory management.

There are two main operating modes, system and normal, supported by separate stacks. User programs operate in normal mode, while sensitive operating

system functions are performed in system mode. This protects critical parts of the operating system from user access. In addition, some instructions are privileged, and execute only in system mode. Memory management functions protect both system memory from user programs, and user memory from other users. Vectored, nonvectored, and nonmaskable interrupts support real-time operating systems.

Memory management is fully integrated with the CPU; no external support circuitry is necessary. A paging address translation mechanism is implemented. Registers in the CPU point to address translation tables located in memory; the most recently used table entries are kept in a Translation Lookaside Buffer (TLB) in the CPU. The CPU performs logical to physical address translation and access protection for each memory reference. When a logical memory reference causes a translation or protection violation, the state of the CPU is automatically restored to restart the instruction. I/O ports can be referenced either by dedicated instructions or by the memory management mechanism mapping logical memory addresses to I/O port addresses.

Extensive trapping facilities, such as integer overflow, subrange out of bounds, and subscript out of bounds, catch common run-time errors. Software debuggers can use trace and breakpoint traps. Privileged instruction traps and memory protection violation traps secure the operating system from user programming errors or mischief. The overflow stack allows recovery from otherwise fatal errors.

The CPU has full 32-bit internal address and data paths. Externally, 32 pins time-multiplex the address and data. The interface is compatible with the complete line of Z-BUS peripherals. The hardware interface features 16-bit or 32-bit memory data path and programmable wait states. Burst transfers and an on-chip cache for instructions and data help develop high-performance systems. The interface supports multiprocessing configurations with interlocked memory references and two types of bus request protocols. The system designer can tailor the Z80,000-based system to cost and performance needs.

In summary, the Z80,000 CPU meets and surpasses the requirements of medium and high-end microprocessor systems for the 1980s. Software program development is easily accomplished with the CPU's sophisticated architecture. The highly pipelined design, on-chip cache, and external interface support systems ranging from dedicated controllers to mainframe computers. While Zilog continues to develop support for the Z80,000 CPU, Z8000 peripherals and development software are fully compatible with this latest in Zilog's line of high-performance microprocessors.

REGISTERS

The Z80,000 CPU is a register-oriented processor offering sixteen 32-bit general-purpose registers, a 32-bit Program Counter (PC), a 16-bit Flag and Control Word (FCW), and nine other special-purpose registers.

The general-purpose register file (Figure 1) contains 64 bytes of storage. The first 16 bytes (RL0,RH0,...,RL7,RH7) can be used as accumulators for byte data. The first 16 words (R0,R1,...,R15) can be used as accumulators for word data, as index registers (except R0), or for memory addresses in compact mode (except R0). Any longword register (RR0,RR2,...,RR30) can be used as an accumulator for longword data, an index register in linear or segmented mode (except RR0), or for memory addresses in linear or segmented mode (except RR0). Quadword registers (RQ0,RQ4,...,RQ28) can be used as accumulators for Multiply, Divide, and Extend Sign instructions. This unique register organization allows bytes and words of data to be manipulated conveniently while leaving most of the register file free to hold addresses, counters, and any other data.

Two registers are dedicated to the Stack Pointer (SP) and Frame Pointer (FP) used by Call, Enter, Exit, and Return and R14 the Frame Pointer. In linear or segmented mode, RR14 is the Stack Pointer and RR12 is the Frame Pointer.

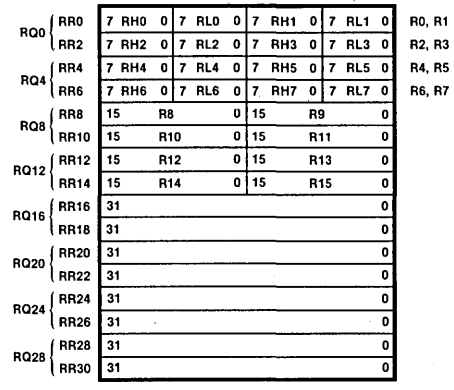


Figure 1. General-Purpose Register File

The PC and FCW form the Program Status (Figure 2), which is automatically saved for traps and interrupts. The bits in FCW indicate operating modes, masks for traps and interrupts, and flags set according to the result of instructions. The remaining special registers are used for memory management, system configuration, and other CPU control (Figure 3).

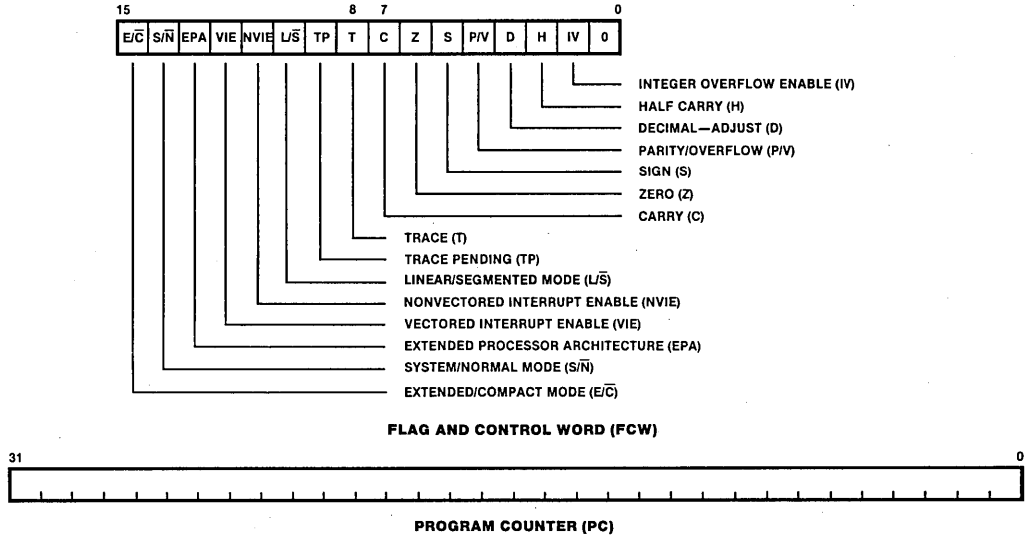


Figure 2. Program Status Registers

As shown in Figure 4, the CPU has three modes of address representation: compact, segmented, and linear. The mode is selected by two control bits in the Flag and Control Word register (Table 1). The Extended/Compact (E/$\overline{C}$) bit selects whether compact addresses (16 bits) or extended addresses (32 bits) are used. For extended addresses the Linear/Segmented (L/$\overline{S}$) bit selects whether linear or segmented addresses are used.

The Load Address instruction can be used to manipulate addresses in any mode of representation.

In compact mode, addresses are 16 bits. Address calculations using compact addresses involve all 16 bits. Compact mode is more efficient and less program-consuming for applications requiring less than 64K bytes of program and less than 64K bytes of data. This efficiency is due to shorter instructions in compact mode, and the fact that addresses in the register file use word rather than longword registers. Applications requiring more than 64K bytes of either program or data should use segmented or linear modes.

Table 1. Address Representation

Control Bits in FCW E/$\overline{C}$	L/$\overline{S}$	Representation
0	0	Compact
0	1	Reserved
1	0	Segmented
1	1	Linear

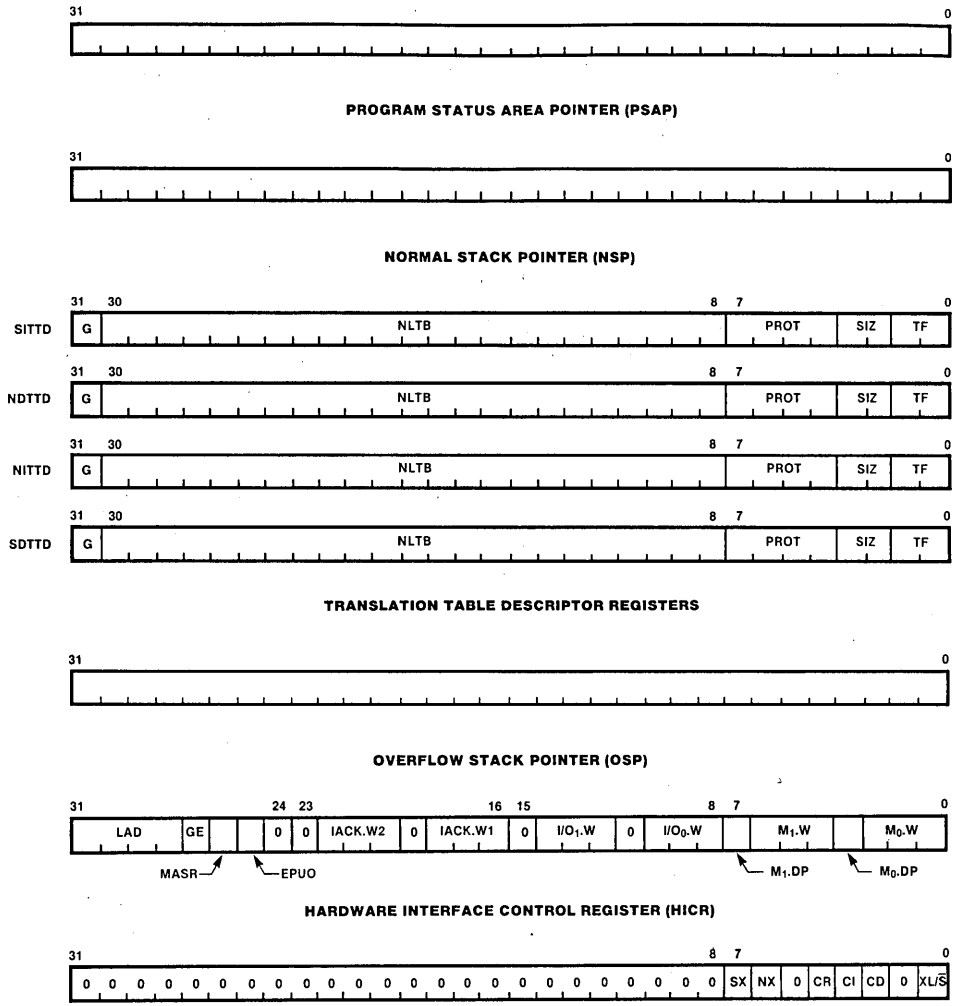


Figure 3. Special-Purpose Control Registers

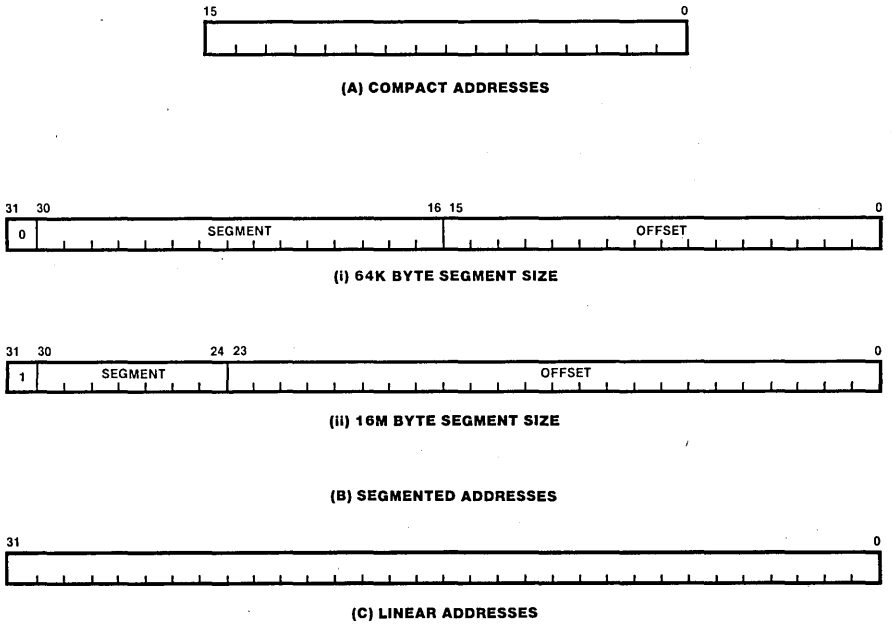


Figure 4. Address Representations

In segmented mode, addresses are 32 bits. Segmented addresses are composed of either a 15-bit segment number and a 16-bit segment offset or a 7-bit segment number and a 24-bit segment offset. Bit 31 of the address selects either of the two types of segmented addresses. Address calculations using segmented addresses involve only the segment offset; the segment number is unaffected. In segmented mode, the address space allows up to 32,768 segments of 64K-byte maximum size and up to 128 segments of 16M-byte maximum size. Many applications benefit from the logical structure of segmentation by allocating individual objects, such as a program module, stack, or large data structure, to separate segments.

In linear mode, addresses are 32 bits. Address calculations using linear addresses involve all 32 bits. In linear mode, the address space of 4G bytes is uniform and unstructured. Many applications benefit from the flexibility of linear addressing by allocating objects at arbitrary positions in the address space.

Memory is byte addressable by the CPU. The address used for multiple-byte data is the address of the most-significant byte. Multiple-byte data can be located at any byte address with no alignment restrictions.

I/O ports can be addressed by either dedicated instructions or by the memory management mechanism mapping logical memory addresses to I/O ports. I/O ports can be byte, word, or longword in size.

NORMAL AND SYSTEM MODES

The CPU has two modes of operation, normal and system, selected by the S/$\overline{N}$ bit in the Flag and Control Word register. These modes impact on CPU operation in three areas: privileged instructions, stack pointers, and memory management.

Since the most sensitive portions of the operating system usually execute in system mode, separate stack pointers are used to isolate the two operating modes.

Some instructions, such as those performing I/O operations or accessing control registers, can only be executed in system mode; in addition, the memory management mechanism allows access to some memory locations in system mode only. Programs executing in normal mode can request services from the operating system using the System Call instruction and trap.

THEORY OF OPERATION

Figure 5 shows a block diagram of the Z80,000 CPU's internal organization, including the following major functional units and data paths:

- The external interface logic controls transactions on the bus. Addresses and data from the internal memory bus are transmitted through the interface to the Z-BUS. The Z-BUS is a time-multiplexed, address/data bus that connects the components of a microprocessor system.
- The cache stores copies of instruction and data memory locations. Instructions are read from the cache on the instruction bus. Data is read from or written to the cache on the memory bus. The cache also includes a copy of the physical Program Counter, so that the logical addresses of instructions are translated only for branches and when incrementing the Program Counter across a page boundary.
- The Translation Lookaside Buffer (TLB) translates logical addresses calculated by the address arithmetic unit to physical addresses used to access the cache.
- The address arithmetic unit performs all address calculations. This unit has a path to the register file for reading base and index registers and another path to the instruction bus for reading displacements and direct addresses. The result of the address calculation is transmitted to the TLB.
- The register file contains the sixteen general-purpose longword registers, Program Status registers, special-purpose control registers, and several registers used to store values temporarily during instruction execution. The register file has one path to the address arithmetic unit and two paths to the execution arithmetic and logic unit.
- The execution arithmetic and logic unit calculates the results of instruction execution, such as add, exclusive-OR, and simple load. This unit has two paths to the register file on which two operands can be read simultaneously or one can be written. One of the paths to the register file is multiplexed with a path from the memory bus.
- The instruction decoding and control unit decodes instructions and controls the operation of the other functional units. This unit has a path from the instruction bus and two programmable logic arrays for separate microcoded control of the two arithmetic units. This unit also controls exception handling and TLB loading.

All of the functional units and data paths listed above are 32 bits wide.

The operation of the CPU is highly pipelined so that several instructions are simultaneously in different stages of execution. Thus, the functional units effectively operate in parallel with one instruction being fetched while an address is calculated for another instruction and results are stored for a third instruction.

Figure 6 shows the six-stage, synchronous pipeline. Instructions flow through each stage of the pipeline in sequence. The various pipeline stages can be working simultaneously on separate instructions or on separate portions of a single complex instruction. Each pipeline stage operates in one processor cycle, which is composed of two clock cycles, called $\phi 1$ and $\phi 2$. Thus, a processor cycle is 200 ns with a 10 MHz clock.

The instruction-fetch stage increments the Program Counter and initiates instructions fetched from the cache. The instruction-decoding stage receives and decodes instructions to set up control of the address-calculation stage.

The address-calculation stage can generally calculate a memory address in one processor cycle, except for Base Index, Relative, and Relative Index addressing modes, which require multiple cycles. After the logical effective address has been calculated, the corresponding physical address is provided by the TLB. The operand-fetch stage fetches the data from the cache and latches it into a holding register.

The execution stage performs data manipulations. Byte, word, and longword results are generally calculated in one processor cycle, but certain instructions, such as multiply and block-move operations, require multiple cycles. During the execution stage, results are stored to registers. Results are stored to the cache and external memory during the operand-store stage. The flags are also set during the operand-store stage.

The cache can handle two references during a processor cycle. Instruction fetches use the $\phi 2$ clock cycle for tag comparison and $\phi 1$ for data access. Either an operand fetch or store can use $\phi 1$ for tag comparison and $\phi 2$ for data access.

The pipeline allows single instructions, like register-to-register load and memory-to-register add, to execute at a rate of one per processor cycle. Thus, the peak performance of the CPU is 5 million instructions per second (MIPS) with a 10 MHz clock. In practice, the actual performance is reduced to approximately one-third of the peak because of delays due to the execution of multiple-cycle instructions, interference between instructions in the pipeline, and main memory accesses for cache and TLB misses.

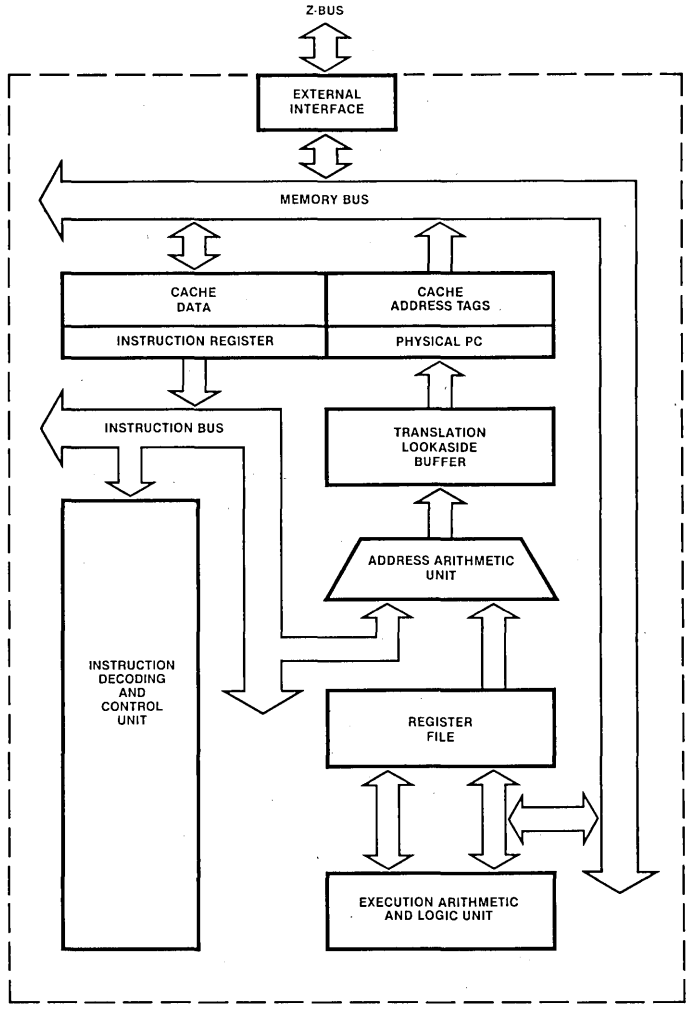


Figure 5. Z80,000 CPU Functional Block Diagram

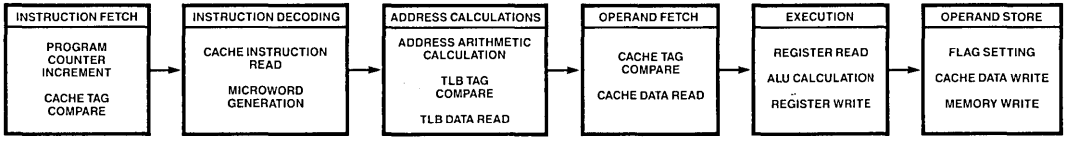


Figure 6. Instruction Pipeline

The CPU and the operating system cooperate in translating logical to physical addresses and protecting memory for execute, read, and write accesses. The CPU implements a paging translation mechanism similar to that in most mainframe and super-minicomputers. The operating system creates translation tables in memory, then initializes pointers to the tables in control registers. The CPU automatically references the tables to perform address translation and access protection. The CPU enables the operating system to implement efficient virtual memory by marking pages that have been referenced or modified and by automatically recovering from address translation faults to allow instruction restart.

The paging translation scheme implemented by the CPU divides the logical address spaces into pages and the physical address space into frames. The logical pages and physical frames are each 1K bytes. A logical page, which is specified by the 22 most-significant bits of the logical address, can be mapped into any physical frame, which is specified by the 22 most-significant bits of the physical address. The 10 least-significant bits, which specify the byte within a page or frame, are not translated. For each memory reference, the CPU translates the logical address to the corresponding physical address and also tests whether access to the memory location is permitted. For most references the information needed to perform the translation is stored in the CPU Translation Lookaside Buffer (TLB). The TLB (Figure 7) stores the translation information for the 16 most recently referenced pages in a fully associative memory. Only when information to translate the page is missing from the TLB does the CPU reference translation tables in memory. In the case of a TLB miss, the CPU translates the logical address using the procedure described below and the translation information is loaded into the TLB entry of the least recently referenced page.

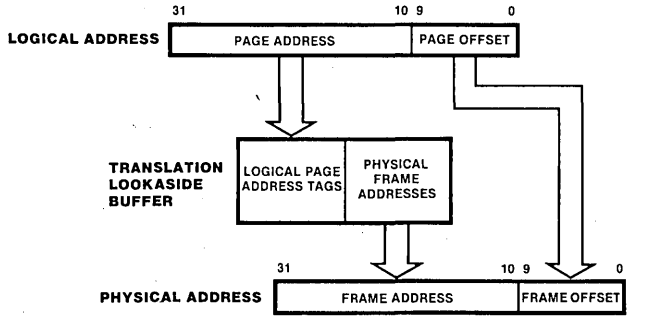


Figure 7. Address Translation Using the TLB

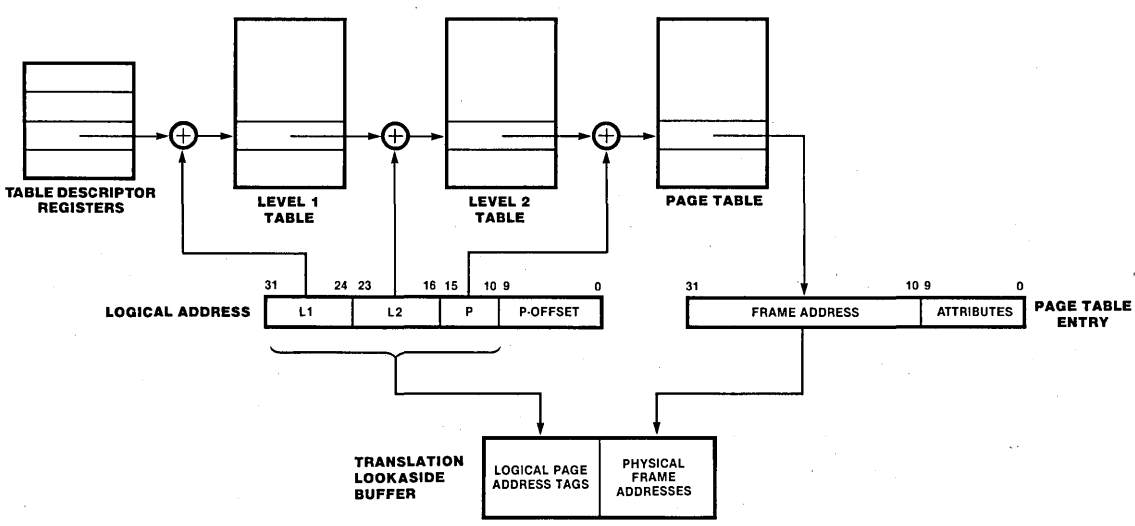


Figure 8. Automatic Loading of the TLB Using Tables in Memory

The address translation mechanism is a three-level paging scheme. A logical address is partitioned into an 8-bit level-1 field (L1), an 8-bit level-2 field (L2), a 6-bit page number field (P), and a 10-bit page offset field (P-OFFSET). During translation, the L1, L2, and P fields are used as indexes into tables in physical memory. The TF field of the Translation Table Descriptor register can be programmed to selectively skip the first and second level tables to reduce both the storage space needed for tables and the number of references necessary to perform translation when the information to translate a page is missing from the on-chip TLB.

To load the TLB (Figure 8), the CPU selects one of four table descriptor registers according to the address space for the reference: system instruction, system data, normal instruction, or normal data. The table descriptor register points to the beginning of the level-1 table in memory; the L1 field of the logical address is used as an index into this table to select the level-1 table entry. Next, the level-1 table entry points to the beginning of the level-2 table; the L2 field of the logical address is used as an index into this table to select the level-2 table entry. After this, the level-2 table entry points to the beginning of the page table in memory; the P field of the logical address is used as an index into this table to select the page table entry. The page table entry contains the physical address of the frame corresponding to the logical address. The CPU then loads the logical page address and physical frame address into the TLB.

When bit 31 in the page table entry is 1, the frame is in physical I/O space. The CPU uses I/O status and timing for the reference. Thus, the address translation process allows protected access to memory-mapped I/O devices.

Figures 9 and 10 show the translation and table entry formats.

Access protection information (Table 2) is encoded in the 4-bit PROT field contained in the Translation Table Descriptor, level-1 table entry, level-2 table entry, or page table entry. During the translation process, a PROT field is encountered at each level. The first PROT field with value other than 1000 is selected; the other PROT fields are ignored. The protection code specifies the types of access (execute, read, and write) permitted in normal and system modes. A value of 1000 in the page table entry indicates no access.

There are several optional features that allow the number of levels and the size of tables to be reduced. When memory address spaces are not separated, two or more of the translation table descriptor registers can be loaded with the same value so that tables are held in common. The table descriptor register can specify that either or both of the level-1 and level-2 tables should be skipped during the translation process. Level-1 tables can be skipped when a 24-bit logical address space is sufficient, both level-1 and level-2 segment tables can be skipped for compact addresses, and level-2 tables can be skipped for compatibility with Z8000 segmented addresses. The table size can be reduced by allocating only 256, 512, or 768 bytes for the tables; the remaining table entries are assumed invalid. The tables can be allocated efficiently for downward growing stacks by setting the G bit of the translation table descriptor or level-1 table entry.

During execution of an instruction, if an invalid translation table entry is encountered or a protection violation is detected, the CPU traps to the operating system. The CPU automatically saves the state of registers and memory so the instruction can simply be restarted.

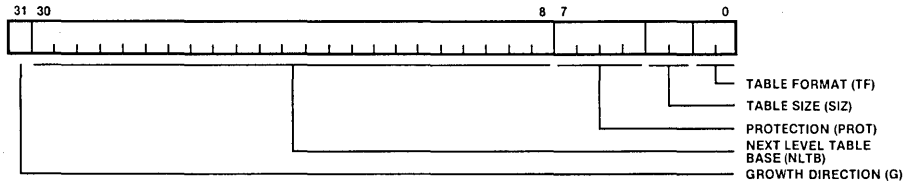


Table Format (TF)

00	THREE LEVELS
01	SKIP LEVEL 2 TABLES
10	SKIP LEVEL 1 TABLES
11	SKIP LEVEL 1 AND LEVEL 2 TABLES

TABLE SIZE (SIZ)	VALID TABLE ENTRIES G = 0	G = 1
00	0 TO 63	0 TO 255
01	0 TO 127	64 TO 255
10	0 TO 191	128 TO 255
11	0 TO 255	192 TO 255

Figure 9. Translation Table Descriptor

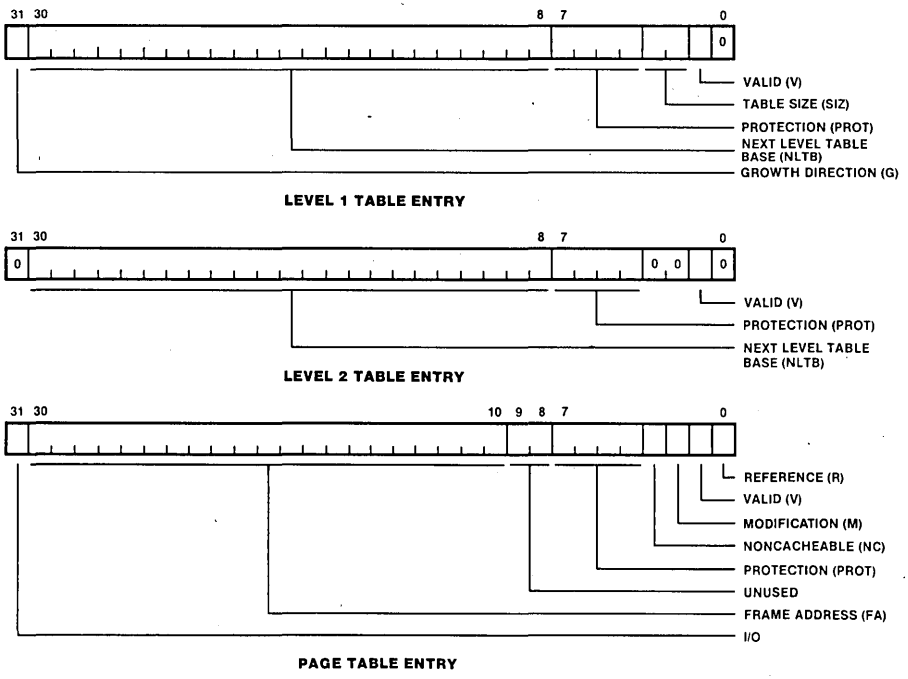


Figure 10. Table Entry Formats

Table 2. Protection Field Encoding

Encoding	System	Normal
0000	NA	NA
0001	RE	NA
0010	RE	E
0011	RE	RE
0100	E	NA
0101	E	E
0110	R	NA
0111	R	R
1000	Next	Next
1001	RW	NA
1010	RW	R
1011	RW	RW
1100	RWE	NA
1101	RWE	E
1110	RWE	RE
1111	RWE	RWE

NA—no access is permitted
R—read access is permitted
W—write access is permitted
E—execute access is permitted
Next—use the protection field of the next level translation table; for page table entries, a PROT field of 1000 indicates no access is permitted.

Several instructions are provided to help the operating system control memory management. The Purge TLB instructions are used to purge the TLB of a single entry, normal mode entries, or all entries. The Load Physical Address instructions translate logical addresses into physical addresses, and set the flags to verify access permission for system call parameters. The Load Normal Data and Load Normal Instruction instructions allow system mode programs to reference normal memory spaces.

The memory management mechanism can be selectively enabled for normal and system space references by using the SX and NX bits of the System Configuration Control Longword register.

EXCEPTIONS

The CPU supports four types of exceptions: reset, bus error, interrupts, and traps. A reset exception occurs when the RESET line is activated; this causes the CPU to be reset to an initialized state. A bus error exception occurs when external hardware indicates an error on a bus transaction. An interrupt is an asynchronous event that typically occurs when a peripheral device needs attention. A trap is a synchronous event that occurs when a particular condition is detected during execution of an instruction.

In responding to a reset exception, the CPU fetches the program status (FCW and PC) from physical address 2. In responding to other exceptions, the CPU pushes the old program status onto the system stack along with information specific to the type of exception. The CPU then fetches a new program status from the table designated by the Program Status Area Pointer control register.

During exception processing, the mode of address representation for the system Stack Pointer and Program Status Area Pointer is either linear or segmented, selected by the XL/$\overline{S}$ bit in the System Configuration Control Longword register. Three types of interrupts are supported: vectored, nonvectored, and nonmaskable. The vectored and nonvectored interrupts have mask bits in the FCW. All interrupts read an identifier word from the bus during an interrupt acknowledge transaction and save the word on the system stack. Vectored interrupts use the lower byte of this word to select a unique PC value from the Program Status Area.

The CPU recognizes twelve trap conditions.

- Extended Instruction trap occurs when an Extended Processing Architecture instruction is executed and the EPA bit in the FCW is clear.
- Privileged Instruction trap occurs when an attempt is made to execute a privileged instruction in normal mode.
- System Call trap occurs when the System Call instruction is executed to request service from the operating system.
- Address Translation trap occurs when an address translation or access protection violation is detected.
- Unimplemented Instruction trap occurs when an attempt is made to execute an instruction with a reserved bit pattern.
- Odd PC trap occurs when an odd-byte address is loaded into the PC.
- Trace trap occurs after execution of an instruction when tracing is enabled by setting the T bit in the FCW.
- Breakpoint trap occurs when the Breakpoint instruction is executed, usually to invoke a debugging or monitoring program.
- Conditional trap occurs when the Conditional trap instruction is executed and the specified condition code is satisfied. This trap can allow detection of user-defined exceptions.
- Integer Arithmetic trap occurs when any of the following three error conditions is detected:
  - Integer Overflow error occurs when overflow is detected during execution of an integer arithmetic instruction and the IV bit in the FCW is set.
  - Bounds Check error occurs when the Check instruction is executed and the source operand is out of bounds.
  - Index error occurs when the Index instruction is executed and the subscript operand is out of bounds.

In descending order, the priority of exceptions is: reset, bus error, trap (other than trace), nonmaskable interrupt, vectored interrupt, and nonvectored interrupt. Trace Trap uses two control bits, T and TP, so that when tracing is enabled, exactly one trace trap occurs after each instruction is executed.

When an Address Translation trap occurs for the system stack, the CPU cannot save the program status and other exception information on the system stack. The system can still recover from this otherwise fatal error because the CPU saves the information on the overflow stack designated in physical memory by the Overflow Stack Pointer control register.

(The current NMOS versions of the Z80,000 CPU do not support the Trace trap and the Integer Overflow portion of the Integer Arithmetic trap. The IV bit of the Flag and Control Word register should always be cleared to zero in these parts. However, these traps will be implemented in future CMOS versions.)

ADDRESSING MODES

The CPU locates operands (the data manipulated by instructions) in registers, memory, I/O ports, or in the instruction. The location of an operand is specified by one of nine addressing modes (Figure 11): Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Index (X), Base Address (BA), Base Index (BX), Relative Address (RA), and Relative Index (RX). Most operations can be used with any addressing mode; however, some operations are restricted. Instruction encoding provides compact representation for the most frequently used addressing modes.

The term Extended Addressing Modes (EAM) refers to the following addressing modes that require one or more extra words to be added to the opcode.

- In compact mode: DA and X (X is equivalent to BA)
- In linear or segmented modes: DA, X, BA, BX, RA, and RX

Addressing Mode	Operand Addressing			Operand Value
	In the Instruction	In a Register	In Memory	
R Register	REGISTER NUMBER →	OPERAND		The contents of the register
IM Immediate	OPERAND			In the instruction
***IR** Indirect Register	REGISTER NUMBER →	ADDRESS →	OPERAND	The contents of the location whose address is in the register
DA Direct Address	ADDRESS →		OPERAND	The contents of the location whose address is in the instruction
***X** Index	REGISTER NUMBER BASE ADDRESS	INDEX + →	OPERAND	The contents of the location whose address is the address in the instruction, plus the contents of the Index Register
***BA** Base Address	REGISTER NUMBER DISPLACEMENT	BASE ADDRESS + →	OPERAND	The contents of the location whose address is the contents of the Base register, plus the displacement in the instruction
***BX** Base Index	REGISTER NUMBER REGISTER NUMBER DISPLACEMENT	BASE ADDRESS INDEX + →	OPERAND	The contents of the location whose address is the contents of the Base register, plus the contents of the Index register, plus the displacement in the instruction
RA Relative Address	DISPLACEMENT →	PC ADDRESS ± →	OPERAND	The contents of the location whose address is the contents of the Program Counter, plus the displacement in the instruction
***RX** Relative Index	REGISTER NUMBER DISPLACEMENT	PC ADDRESS INDEX + →	OPERAND	The contents of the location whose address is the contents of the Program Counter, plus the contents of the Index register, plus the displacement in the instruction

*R0 and RR0 cannot be used for Indirect, Base, or Index registers

Figure 11. Addressing Modes

DATA TYPES

The CPU supports operations on the following data types.

- Bit
- Bit field—1 to 32 contiguous bits within a longword
- Signed integer—byte, word, longword, and quadword
- Unsigned integer—byte, word, longword, and quadword
- Logical value—byte, word, and longword
- Address—word or longword
- Packed BCD integer—byte
- Stack—word and longword
- String—dynamic length byte, word, and longword

INSTRUCTION SET SUMMARY

The CPU provides 11 types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Bit Field
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control
- Extended Instructions

Instructions are encoded in one or more words, located in memory at even addresses. The generic instruction mnemonic indicates the instruction operates on words; addition of a "B" or "L" suffix to the mnemonic indicates operation on bytes or longwords, respectively. For example: CLR operates on words, CLRB on bytes, and CLRL on longwords.

EXTENDED INSTRUCTIONS

The Z80,000 CPU supports extended instructions through the Zilog Extended Processing Architecture (EPA). The EPA facility allows the operations defined in the Z80,000 architecture to be extended by software or hardware.

Up to four Extended Processing Units (EPUs) can be included in a Z80,000 CPU system. The CPU and EPU cooperate in execution of EPA instructions. When the CPU encounters an EPA instruction, the instruction is transmitted across the external bus to the appropriate EPU. The CPU then performs transactions on the external bus to transfer data between the EPU and memory or the EPU and CPU. Transfers between the EPU and CPU can involve the CPU general-purpose registers or FCW flag byte. EPU internal operations do not require any data transfers. After the data transfers for the EPU instruction are completed, the CPU can continue processing while the EPU performs the operation. While the EPU is processing an instruction, it can drive the $\overline{\text{EPUBSY}}$ signal to stop the CPU.

The data processing operations performed by the EPU are transparent to the CPU. The EPU can execute floating point operations, decimal arithmetic, specialized operating system functions, signal processing operations, or any other that the system designer chooses. For this reason, no mnemonic is listed for the extended instructions, as the mnemonic will depend on the type of EPU. EPUs designed to speed execution of special purpose operations can provide significant performance improvements. The operation of the EPU can be overlapped with operation of the CPU and other EPUs.

The EPA bit in the Flag and Control Word register indicates whether an EPU is present. If no EPU is present, the CPU traps EPA instructions for software simulation. Thus, the EPA facility can be used even with no external support circuitry. This allows software compatibility between systems, whether or not an EPU is present. The system designer can choose to include an EPU in high-performance systems but not in low-cost systems, and software can be developed using the EPA instructions before an EPU is available.

CACHE

The CPU implements a cache mechanism to keep on-chip copies of the most recently referenced memory locations (Figure 12). The CPU examines the cache on memory fetches to determine if the addressed data are located in the cache. If the information is in the cache (a hit), then the CPU fetches from the cache, and no transaction is necessary on the external interface. If the information is not in the cache (a miss), then the CPU performs a memory read transaction to fetch the missing information.

The cache stores data in blocks of 16 bytes. Each data word in the cache has an associated validity bit to indicate whether or not the word is a valid copy of the corresponding main memory location. The cache contains 16 blocks, providing 256 bytes of storage.

The cache is fully associative, so that a block currently needed and missing in the cache can replace any block in the cache. Moreover, when a block miss occurs, the least recently used (LRU) block in the cache is replaced. When a cache miss occurs on an instruction fetch, the CPU fetches the missing instruction from memory and prefetches the following words in the block using a burst transaction. When a cache miss occurs on an operand fetch, the CPU fetches the missing data from memory. (The CPU uses burst transactions only for fetching operands when more than one data transfer is necessary: longword operands on a 16-bit bus, unaligned operands, string instructions, Load Multiple instructions, and loading Program Status.)

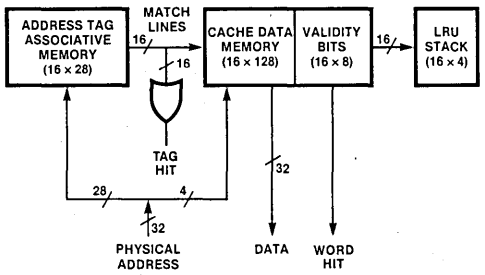


Figure 12. Cache Organization

On store references, the data is written to memory (store through), and if the reference hits in the cache, the data is also written to the cache. If the store reference misses in the cache, the cache is unaffected.

Software has some control over the cache. The cache can be selectively enabled for instruction and data references by bits CI and CD in the SCCL control register. The memory management mechanism allows cacheing to be inhibited for individual pages. The Pcache instruction can be used to invalidate all information in the cache.

The cache has an option, controlled by bit CR in SCCL, to inhibit block replacement on a miss. This option can be used to lock fixed locations into the cache for fast, onchip access. To do this, the cache is first enabled for block replacement of data references only. Selected blocks are read into the cache. The block replacement algorithm is then disabled, while the cache is enabled for instruction and data references.

PIN DESCRIPTIONS

The CPU has 58 signal lines and additional power supply connections. Pin functions are shown in Figure 13a; pin assignments are shown in Figure 13b, 13c, and Table 4.

AD_0-AD_{31}. *Address/Data (Bidirectional, active High, 3-state).* These 32 lines are time-multiplexed to transfer address and data. At the beginning of each transaction the lines are driven with the 32-bit address. After the address has been driven, the lines are used to transfer one or more bytes, words, or longwords of data.

$\overline{AS}$. *Address Strobe (Output, active Low, 3-state).* The falling edge of $\overline{AS}$ indicates the beginning of a transaction and shows that the address and ST_0-ST_3, are valid. R/$\overline{W}$, BL/$\overline{W}$, BW/$\overline{L}$, and $\overline{BRST}$ are valid on the rising edge of $\overline{AS}$.

BL/$\overline{W}$; BW/$\overline{L}$. *Byte, Longword/Word; Byte, Word/Longword (Output, 3-state).* These two lines specify the data transfer size.

BL/$\overline{W}$	BW/$\overline{L}$	Size
High	High	Byte
Low	High	Word
High	Low	Longword
Low	Low	Reserved

$\overline{BRST}$. *Burst (Output, active Low, 3-state).* A Low on this line indicates that the CPU is performing a burst transfer; i.e, multiple Data Strobes following a single Address Strobe.

$\overline{BRSTA}$. *Burst Acknowledge (Input, active Low).* A Low on this line indicates that the responding device can support burst transfers.

$\overline{BUSACK}$. *Bus Acknowledge (Output, active Low).* A Low on this line indicates that the CPU has relinquished control of the local bus in response to a bus request.

$\overline{BUSREQ}$. *Bus Request (Input, active Low).* A Low on this line indicates that a bus requestor has obtained or is trying to obtain control of the local bus.

CLK. *Clock (Input).* This is the clock used to generate all CPU timing.

$\overline{DS}$. *Data Strobe (Output, active Low, 3-state).* DS is used for timing data transfers.

EPUABORT. ***EPU Abort (Output, active High).*** **A High on this line indicates that the CPU is aborting execution of an EPA instruction, typically because an Address Translation trap has occurred.**

$\overline{EPUBSY}$. *EPU Busy (Input, active Low).* A Low on this line indicates that an EPU is busy. This line is used to synchronize the operation of the CPU with an EPU during execution of an EPA instruction.

$\overline{GACK}$. *Global Acknowledge (Input, active Low).* A Low on this line indicates the CPU has been granted control of a global bus.

$\overline{GREQ}$. *Global Request (Output, active Low, 3-state).* A Low on this line indicates the CPU has obtained or is trying to obtain control of a global bus.

$\overline{IE}$. *Input Enable (Output, active Low, 3-state).* A Low on this line can be used to enable buffers on the AD lines to drive toward the CPU.

$\overline{NMI}$. *Non-Maskable Interrupt (Input, Edge activated).* A High-to-Low transition on this line requests a nonmaskable interrupt.

$\overline{NVI}$. *Non-Vectored Interrupt (Input, active Low).* A Low on this line requests a non-vectored interrupt.

$\overline{OE}$. *Output Enable (Output, active Low, 3-state).* A Low on this line can be used to enable buffers on the AD lines to drive away from the CPU.

$\overline{RESET}$. *Reset (Input, active Low).* A Low on this line resets the CPU.

RSP_0-RSP_1. *Response (Input).* These lines encode the response to transactions initiated by the CPU. Note that RSP_0 and RSP_1 can be connected together for Z-BUS $\overline{WAIT}$ timing.

RSP_0	RSP_1	Response
High	High	Ready
Low	High	Bus Error
High	Low	Bus Retry
Low	Low	Wait

R/$\overline{W}$. *Read/Write (Output, Low = Write, 3-state).* This signal indicates the direction of data transfer.

ST_0-ST_3. *Status (Output, active High, 3-state).* These lines specify the kind of transaction occurring on the bus. (See Table 5.)

VBB. *Substrate Bias Generator* (Output, for internal biasing only).

$\overline{VI}$. *Vectored Interrupt (Input, active Low).* A Low on this line requests a vectored interrupt.

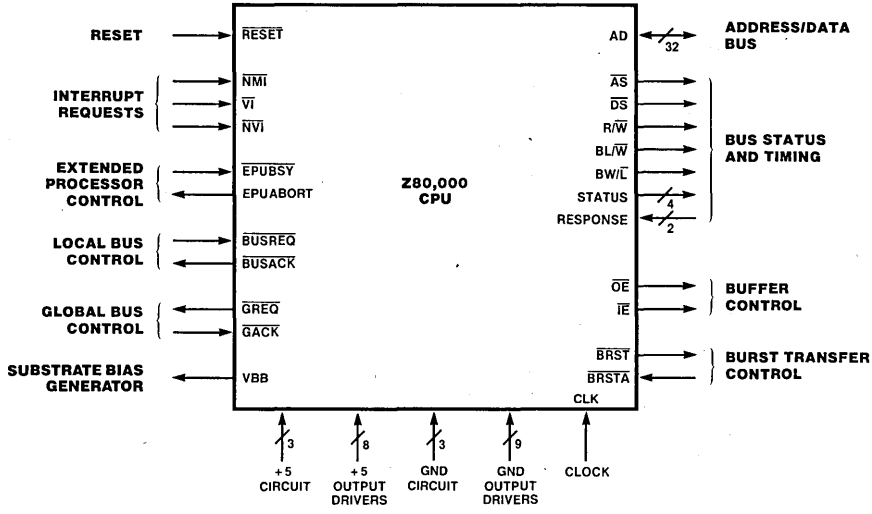
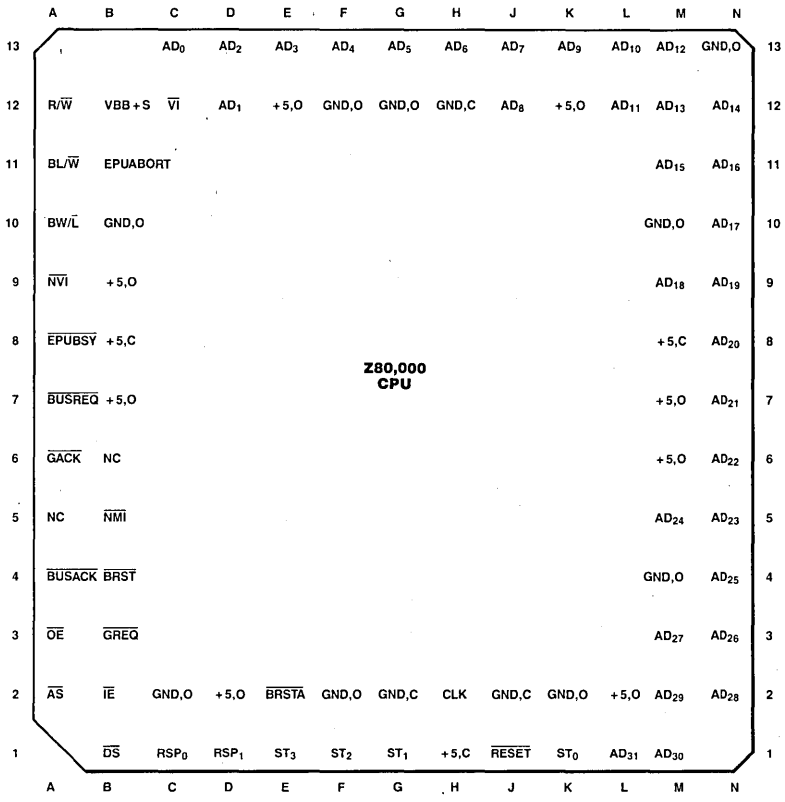
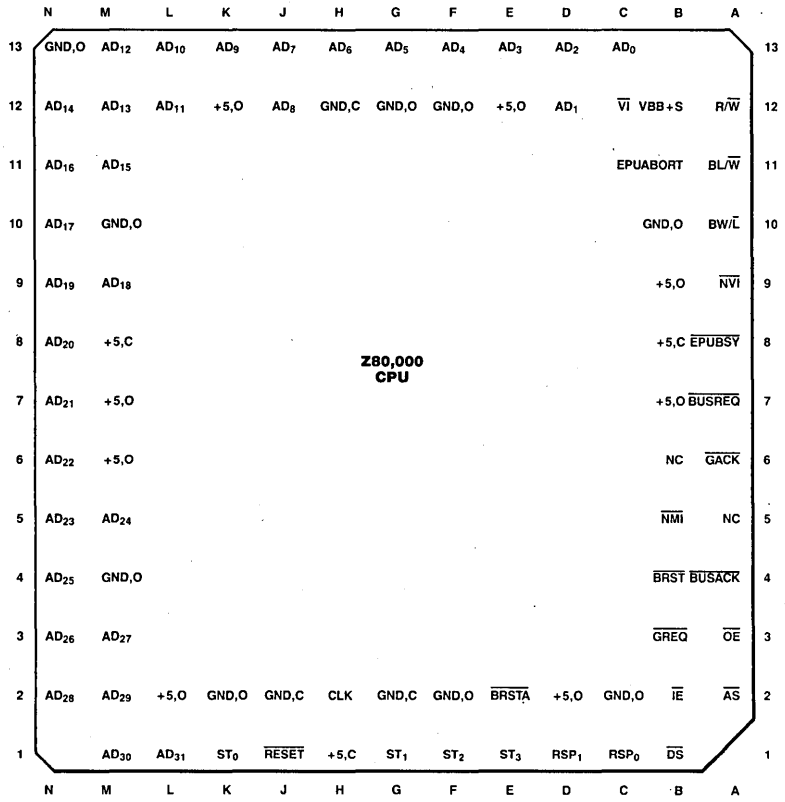


Figure 13a. Pin Functions



ABBREVIATIONS
O = OUTPUT DRIVER
C = CIRCUIT
S = SUBSTRATE

Figure 13b. 84-pin Pin Grid Array (PGA), Pin Assignments, Preliminary
View toward PC Board



ABBREVIATIONS
O = OUTPUT DRIVER
C = CIRCUIT
S = SUBSTRATE

Figure 13c. 84-pin Pin Grid Array (PGA), Pin Assignments, Preliminary View of Pin Side

Table 4. Z80,000 Pin Assignments, Functional Grouping

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
AD_0	C13	AD_{25}	N4	GND Output Driver	B10	$\overline{GREQ}$	B3
AD_1	D12	AD_{26}	N3	GND Output Driver	C2	$\overline{GACK}$	A6
AD_2	D13	AD_{27}	M3	GND Output Driver	F2		
AD_3	E13	AD_{28}	N2	GND Output Driver	F12	$\overline{NVI}$	A9
AD_4	F13	AD_{29}	M2	GND Output Driver	G12	$\overline{NMI}$	B5
AD_5	G13	AD_{30}	M1	GND Output Driver	K2	$\overline{VI}$	C12
AD_6	H13	AD_{31}	L1	GND Output Driver	M4		
AD_7	J13			GND Output Driver	M10	VBB and substrate	B12
AD_8	J12	+5 Circuit	B8	GND Output Driver	N13		
AD_9	K13	+5 Circuit	H1			RSP_0	C1
AD_{10}	L13	+5 Circuit	M8	$\overline{DS}$	B1	RSP_1	D1
AD_{11}	L12			$\overline{AS}$	A2		
AD_{12}	M13	+5 Output Driver	B7			$\overline{BRST}$	B4
AD_{13}	M12	+5 Output Driver	B9	$\overline{IE}$	B2	$\overline{BRSTA}$	E2
AD_{14}	N12	+5 Output Driver	D2	$\overline{OE}$	A3		
AD_{15}	M11	+5 Output Driver	E12			ST_3	E1
AD_{16}	N11	+5 Output Driver	K12	$\overline{BUSACK}$	A4	ST_2	F1
AD_{17}	N10	+5 Output Driver	L2	$\overline{BUSREQ}$	A7	ST_1	G1
AD_{18}	M9	+5 Output Driver	M6			ST_0	K1
AD_{19}	N9	+5 Output Driver	M7	$\overline{EPUBSY}$	A8		
AD_{20}	N8			EPUABORT	B11	CLK	H2
AD_{21}	N7	GND Circuit	G2				
AD_{22}	N6	GND Circuit	H12	BW/$\overline{L}$	A10	$\overline{RESET}$	J1
AD_{23}	N5	GND Cricuit	J2	BL/$\overline{W}$	A11		
AD_{24}	M5			R/$\overline{W}$	A12	NC	A5
						NC	B6

The +5 and GND pins are separated into groups to provide isolated power supply connections.

Group	+5	GND
Output Driver	B7, B9, D2, E12, K12, L2, M6, M7	B10, C2, F2, F12, G12, K2, M4, M10, N13
Circuit	B8, H1, M8	G2, H12, J2

MULTIPROCESSOR CONFIGURATIONS

The CPU provides support for interconnection in four types of multiprocessor configurations (Figure 14): coprocessor, slave processor, tightly-coupled multiple CPUs, and loosely-coupled multiple CPUs.

Coprocessors work synchronously with the CPU to execute a single instruction stream using the Extended Processing Architecture facility. The $\overline{\text{EPUBSY}}$ and EPUABORT signals are dedicated for connection with coprocessors.

Slave processors, such as the Z9516 DMA Transfer Controller, perform dedicated functions asynchronously to the CPU. The CPU and slave processor share a local bus, of which the CPU is the default master, using the CPU's $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ lines.

Tightly-coupled, multiple CPUs execute independent instruction streams and generally communicate through shared memory located on a common (global) bus using the CPU's $\overline{\text{GREQ}}$ and $\overline{\text{GACK}}$ lines. Each CPU is default master of its local bus, but the global bus master is chosen by an external arbiter. The CPU also provides status information about interlocked memory references (for Test and Set, Increment Interlocked, and Decrement Interlocked instructions), which can be used with multiported memories.

Loosely-coupled, multiple CPUs generally communicate through a multiple-ported peripheral, such as the Z8038 FIO. The Z80,000 CPU's I/O and interrupt facilities can support loosely-coupled multiprocessing.

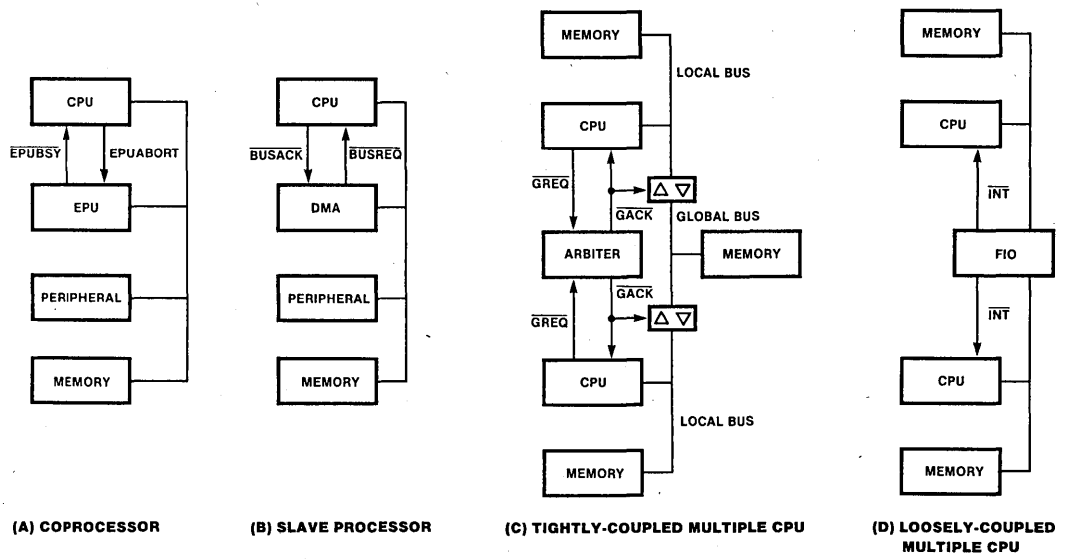


Figure 14. Multiprocessor Configurations

HARDWARE INTERFACE CONTROL REGISTER

The Hardware Interface Control register (HICR) specifies certain characteristics of the hardware configuration surrounding the CPU, including memory data path width, and number of automatic wait states. The physical memory address space is divided into two sections, M_0 and M_1, selected by bit 30 of the address. A typical system would locate slow, 16-bit wide bootstrap ROM in M_0 and faster 32-bit wide dynamic RAM in M_1. The physical I/O address space is similarly divided into two sections, I/O_0 and I/O_1, selected by bit 30 of the address.

Fields in HICR specify the following interface characteristics (see Figure 3):

Memory data path (M_0.DP, M_1.DP)—The data path width for M_0 and M_1 are each specified as 16 or 32 bits.

Automatic wait states (M_0.W, M_1.W, I/O_0.W, I/O_1.W, IACK.W1, IACK.W2)—The number of Wait states automatically inserted by the CPU for references to M_0, M_1, I/O_0, I/O_1, and interrupt acknowledge, are separately specified.

Global bus protocol control (LAD, GE)—The CPU can access a global bus (a bus shared with other CPUs). On references to the global bus, the CPU must use a request/acknowledge handshake with an external arbiter. The GE field enables the use of the global bus; the LAD field selects

the portion of the address space used for references to the global bus.

Minimum Address Strobe Rate (MASR)—This optional feature ensures that an Address Strobe is generated at least once every 16 bus clock cycles. This is useful for refreshing dynamic and pseudostatic RAMS.

EPU overlap (EPUO)—This bit, along with another bit in an EPU control register, controls the degree of overlap for CPU and EPU operations. The degree of overlap can be limited to simplify debugging and recovery from exceptions, although to do so reduces overall execution speed.

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs
with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient
Temperature. . . . . .See ordering information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only: operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

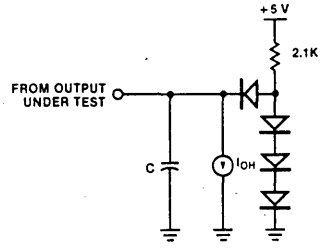
STANDARD TEST CONDITIONS

The DC characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

- S = 0°C to +70°C, $+4.75V \leq V_{CC} \leq +5.25V$

All ac parameters assume a total load capacitance (C), including parasitic capacitances, of 100 pf max.



DC CHARACTERISTICS

Symbol	Parameter	Min	Typ	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.0		V_{CC} + 0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3		0.6	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0		V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	−0.3		0.8	V	
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −500 μA
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = +4.0 mA
I_{IL}	Input Leakage			±10	μA	$0.4 \leq V_{IN} \leq +2.4V$
I_{CC}	V_{CC} Supply Current		900		mA	

AC CHARACTERISTICS*

Number	Symbol	Parameter	Min	Max
1	TcC	PClock Cycle Time	100	500
2	TwCh	Width (High)	40	
3	TwCl	Width (Low)	40	
4	TfC	PClock Fall Time		10
5	TrC	PClock Rise Time		10
6	TdC(Bz)	PClock ↑ to Bus Float		47
7	TsA(C)	Address Valid to PClock ↑ Setup Time	0	
8	TdC(Az)	PClock ↑ to Address Float		40
9	TdA(DR)	Address Valid to Read Data Required Valid (single memory read timing with one wait, without wait is 280)	480†	
10	TsDR(C)	Read Data to PClock ↑ Setup Time	20	
11	TdDS(A)	$\overline{DS}$ ↑ to Address Active	70†	
12	TdC(DW)	PClock ↑ to Write Data Valid		65
13	ThDR(C)	Read Data Valid to PClock ↑ Hold Time	5	
14	ThDR(DS)	Read Data Valid to $\overline{DS}$ ↑ Hold Time	0	
15	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	255†	
16	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	55†	
17	TdC(ASf)	PClock ↑ to $\overline{AS}$ ↓ Delay		50
18	TsS(C)	Status Valid to PClock ↑ Setup Time	0	
19	TdC(ASr)	PClock ↑ to $\overline{AS}$ ↑ Delay		50
20	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid (single memory read timing with one wait; without wait is 120)	320†	
21	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	260†	
22	TwAS	$\overline{AS}$ Width (Low)	85†	
23	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	70†	
24	TdAz(DSR)	Address Float to $\overline{DS}$ ↓ (Read) Delay	0	
25	TdBz(BUS)	Bus Float to $\overline{BUSACK}$ ↓ Delay	9	
26	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Read) Delay	70†	
27	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid (single memory read timing with one wait, without wait is 20)	220†	
28	TdC(DS)	PClock ↑ to $\overline{DS}$ ↑ Delay		50
29	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid	70†	
31	TdC(DSR)	PClock ↑ to $\overline{DS}$ (Read) ↓ Delay		50
32	TwDSR	$\overline{DS}$ (Read) Width (Low) (single memory read timing with one wait, without wait is 85)	285†	
33	TdC(DSW)	PClock ↑ to $\overline{DS}$ (Write) ↓ Delay		50
34	TwDSW	$\overline{DS}$ (Write) Width (Low)	185†	
35	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid	120†	
36	TdC(DSI)	PClock ↑ to $\overline{DS}$ (I/O) ↓ Delay		50

*Units in nanoseconds. See Footnotes to AC Characteristics.
†Clock-cycle time-dependent characteristics.

AC CHARACTERISTICS* (Continued)

Number	Symbol	Parameter	Min	Max
37	TwDSI	$\overline{DS}$ (I/O) Width (Low)	185†	
38	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay	70†	
39	TdC(DSA)	PClock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		50
40	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Required Valid	120†	
41	TdC(S)	PClock ↑ to Status Valid Delay		60
42	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	60†	
43	TsR(C)	$\overline{RESET}$ to PClock ↓ Setup Time	20	
44	ThR(C)	$\overline{RESET}$ to PClock ↓ Hold Time	25	
45	TwNMIl	$\overline{NMI}$ Width (Low)	290†	
46	TwNMIh	$\overline{NMI}$ Width (High)	190†	
47	TsNMI(C)	$\overline{NMI}$ ↓ to PClock ↑ Setup Time	40	
48	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to PClock ↑ Setup Time	40	
49	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to PClock ↑ Hold Time	20	
50	TwVI	$\overline{VI}$, $\overline{NVI}$ Width (Low)	290†	
51	TsBREQ(C)	$\overline{BUSREQ}$ Change to PClock ↑ Setup Time	40	
52	TwBREQ	$\overline{BUSREQ}$ Width (Low)	290†	
53	ThBREQ(C)	$\overline{BUSREQ}$ to PClock ↑ Hold Time	20	
54	TdC(BACKr)	PClock ↑ to $\overline{BUSACK}$ ↑ Delay		65
55	TdC(BACKf)	PClock ↑ to $\overline{BUSACK}$ ↓ Delay		65
57	TdC(IEr)	PClock ↑ to $\overline{IE}$ ↑ Delay		65
58	TdC(IEf)	PClock ↑ to $\overline{IE}$ ↓ Delay		65
59	TsBRSTA(C)	$\overline{BRSTA}$ to PClock ↑ Setup Time	25	
60	TsEPUBSY(C)	$\overline{EPUBSY}$ to PClock ↑ Setup Time	20	
61	ThBRSTA(C)	$\overline{BRSTA}$ to PClock ↑ Hold Time	5	
62	ThEPUBSY(C)	$\overline{EPUBSY}$ to PClock ↑ Hold Time	5	
63	TsRSP(C)	RSP Change to PClock ↑ Setup Time	20	
64	ThRSP(C)	RSP to PClock ↑ Hold Time	5	
65	TdIE(OE)	$\overline{OE}$ Change to $\overline{IE}$ Change Delay	270†	
66	TwGACK	$\overline{GACK}$ Width (Low)	290†	
67	TsGACK(C)	$\overline{GACK}$ Change to PClock ↑ Setup time	40	
68	ThGACK(C)	$\overline{GACK}$ to PClock ↑ Hold Time	20	
69	TdC(OEf)	PClock ↑ to $\overline{OE}$ ↓ Delay		50
70	TdC(OEr)	PClock ↑ to $\overline{OE}$ ↑ Delay		50
71	TdC(BRSTf)	PClock ↑ to $\overline{BRST}$ ↓ Delay		65
72	TdC(BRSTr)	PClock ↑ to $\overline{BRST}$ ↑ Delay		65
73	TdC(GREQf)	PClock ↑ to $\overline{GREQ}$ ↓ Delay		50
74	TdC(GREQr)	PClock ↑ to $\overline{GREQ}$ ↑ Delay		50

*Units in nanoseconds. See Footnotes to AC Characteristics.
†Clock-cycle time-dependent characteristics.

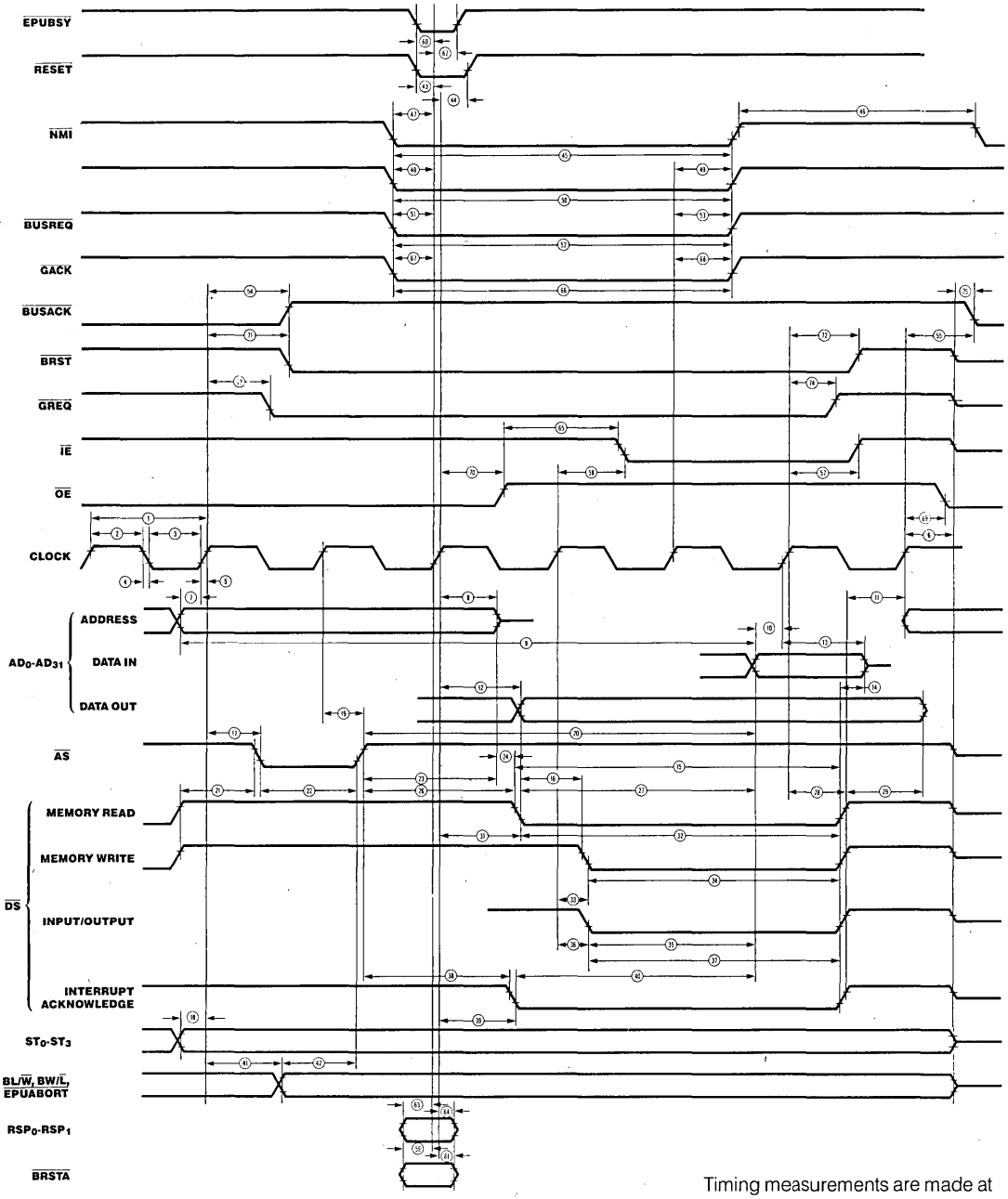
FOOTNOTES TO AC CHARACTERISTICS

No.	Symbol	Equation
9	TdA(DR)	5TcC – TsDR(C)
11	TdDS(A)	TcC – TdC(DS) + TdC(Az) – 20ns
15	TdDW(DS)	3TcC – TdC(DW) + TdC(DS) – 30ns
16	TdDW(DSW)	TcC – TdC(DW) + TdC(DSW) – 30ns
20	TdAS(DR)	4TcC – TdC(ASr) – TsDR(C) – TrC
21	TdDS(AS)	3TcC – TdC(DS) + TdC(ASf) – 40ns
22	TwAS	TcC – 15ns
23	TdAS(A)	TcC – TdC(ASr) + TdC(Az) – 20ns
26	TdAS(DSR)	TcC – TdC(ASr) + TdC(DSR) – 30ns
27	TdDSR(DR)	3TcC – TdC(DSR) – TsDR(C) – TrC
29	TdDS(DW)	TcC – TdC(DS) + TdC(Az) – 20ns
32	TwDSR	3TcC – 15ns
34	TwDSW	2TcC – 15ns
35	TdDSI(DR)	2TcC – TdC(DSI) – TsDR(C) – TrC
37	TwDSI	2TcC – 15ns
38	TdAS(DSA)	TcC – TdC(ASr) + TdC(DSA) – 30ns
40	TdDSA(DR)	2TcC – TdC(DSA) – TsDR(C) – TrC
42	TdS(AS)	TcC – TdC(S) + TdC(ASr) – 30ns
45	TwNMIl	3TcC – 10ns
46	TwNMIh	2TcC – 10ns
50	TwVI	3TcC – 10ns
52	TwBREQ	3TcC – 10ns
65	TdIE(OE)	TcC – TdC(OEr) + TdC(IEf) – 45ns
66	TwGACK	3TcC – 10ns

AC Timing Test Conditions

V_{OL} = 0.8V
V_{OH} = 2.0V
V_{IL} = 0.8V
V_{IH} = 2.4V
V_{ILC} = 0.6V
V_{IHC} = 3.0V

AC TIMING



Timing measurements are made at the following voltages.

	High	Low
Clock	3.0V	0.6V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	ΔV	±0.5V

Interfacing Z80 CPUs to the Z8500 Peripheral Family

October 1988

INTRODUCTION

The Z8500 Family consists of universal peripherals that can interface to a variety of microprocessor systems that use a non-multiplexed address and data bus. Though similar to Z80 peripherals, the Z8500 peripherals differ in the way they respond to I/O and Interrupt Acknowledge cycles. In addition, the advanced features of the Z8500 peripherals enhance system performance and reduce processor overhead.

To design an effective interface, the user needs an understanding of how the Z80 Family interrupt structure works, and how the Z8500 peripherals interact with this structure. This application note provides basic information on the interrupt structures, as well as a discussion of the hardware and software considerations involved in interfacing the Z8500 peripherals to the Z80 CPUs. Discussions center around each of the following situations:

- Z80A 4 MHz CPU to Z8500 4 MHz peripherals
- Z80B 6 MHz CPU to Z8500A 6 MHz peripherals
- Z80H 8 MHz CPU to Z8500 4 MHz peripherals
- Z80H 8 MHz CPU to Z8500A 6 MHz peripherals

This application note assumes the reader has a strong working knowledge of the Z8500 peripherals; it is not intended as a tutorial.

CPU HARDWARE INTERFACING

The hardware interface consists of three basic groups of signals: data bus, system control, and interrupt control, described below. For more detailed signal information, refer to Zilog's Data Book, Universal Peripherals.

Data Bus Signals

D_7-D_0 Data Bus (bidirectional, 3-state). This bus transfers data between the CPU and the peripherals.

System Control Signals

A_n-A_0 Address Select Lines (optional). These lines select the port and/or control registers.

$\overline{CE}$ Chip Enable (input, active Low). $\overline{CE}$ is used to select the proper peripheral for programming. $\overline{CE}$ should be gated with $\overline{IORQ}$ or $\overline{MREQ}$ to prevent spurious chip selects during other machine cycles.

$\overline{RD}$* Read (input, active Low). $\overline{RD}$ activates the chip-read circuitry and gates data from the chip onto the data bus.

$\overline{WR}$* Write (input, active Low). $\overline{WR}$ strobes data from the data bus into the peripheral.

*Chip reset occurs when $\overline{RD}$ and $\overline{WR}$ are active simultaneously.

Interrupt Control

$\overline{INTACK}$ Interrupt Acknowledge (input, active Low). This signal indicates an Interrupt Acknowledge cycle and is used with $\overline{RD}$ to gate the interrupt vector onto the data bus.

$\overline{INT}$ Interrupt Request (output, open-drain, active Low).

IEI Interrupt Enable In (input, active High).

IEO Interrupt Enable Out (output, active High).

These lines control the interrupt daisy chain for the peripheral interrupt response.

Z8500 I/O OPERATION

The Z8500 peripherals generate internal control signals from $\overline{RD}$ and $\overline{WR}$. Since PCLK has no required phase relationship to $\overline{RD}$ or $\overline{WR}$, the circuitry generating these signals provides time for metastable conditions to disappear.

The Z8500 peripherals are initialized for different operating modes by programming the internal registers. These internal registers are accessed during I/O Read and Write cycles, which are described below.

Read Cycle Timing

Figure 1 illustrates the Z8500 Read cycle timing. All register addresses and $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ goes active after $\overline{RD}$ goes active, or if $\overline{CE}$ goes inactive before $\overline{RD}$ goes inactive, then the effective Read cycle is shortened.

Write Cycle Timing

Figure 2 illustrates the Z8500 Write cycle timing. All register addresses and $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ goes active after $\overline{WR}$ goes active, or if $\overline{CE}$ goes inactive before $\overline{WR}$ goes inactive, then the effective Write cycle is shortened. Data must be available to the peripheral prior to the falling edge of $\overline{WR}$.

PERIPHERAL INTERRUPT OPERATION

Understanding peripheral interrupt operation requires a basic knowledge of the Interrupt Pending (IP) and Interrupt Under Service (IUS) bits in relation to the daisy chain. Both Z80 and Z8500 peripherals are designed in such a way that no additional interrupts can be requested during an Interrupt Acknowledge cycle. This allows the interrupt daisy chain to settle, and ensures proper response of the interrupting device.

The IP bit is set in the peripheral when CPU intervention is required (such conditions as buffer empty, character available, error detection, or status changes). The Interrupt Acknowledge cycle does not necessarily reset the IP bit. This bit is cleared by a software command to the peripheral, or when the action that generated the interrupt is completed (i.e., reading a character, writing data, resetting errors, or changing the status). When the interrupt has been serviced, other interrupts can occur.

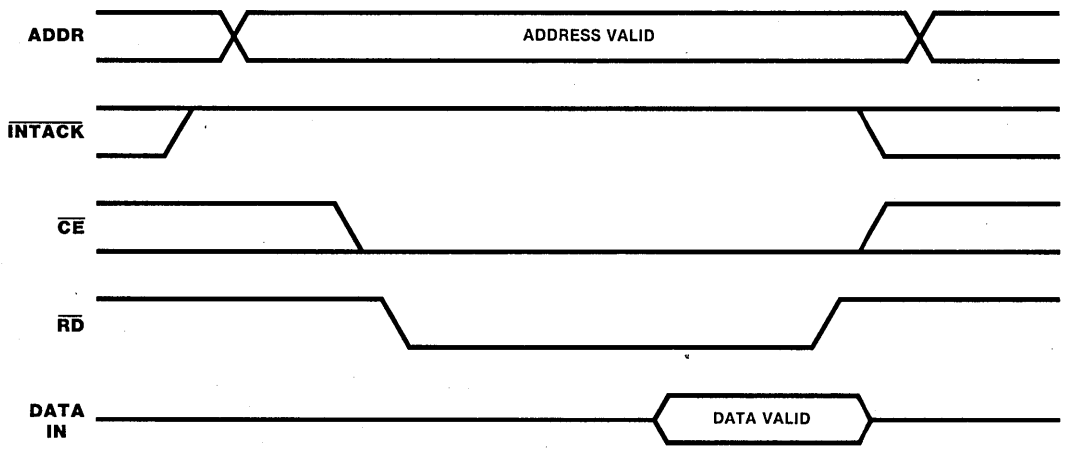


Figure 1. Z8500 Peripheral I/O Read Cycle Timing

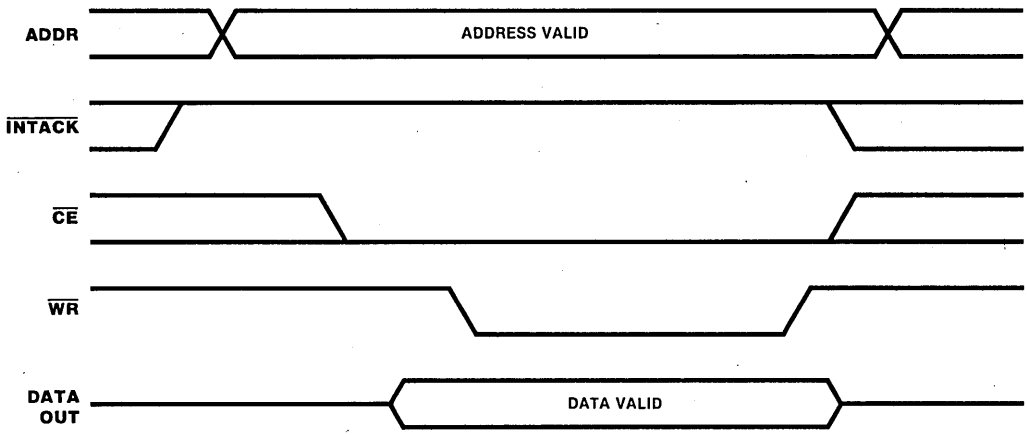


Figure 2. Z8500 Peripheral I/O Write Cycle Timing

The IUS bit indicates that an interrupt is currently being serviced by the CPU. The IUS bit is set during an Interrupt Acknowledge cycle if the IP bit is set and the IEI line is High. If the IEI line is Low, the IUS bit is not set, and the device is inhibited from placing its vector onto the data bus. In the Z80 peripherals, the IUS bit is normally cleared by decoding the RETI instruction, but can also be cleared by a software command (S10). In the Z8500 peripherals, the IUS bit is cleared only by software commands.

Z80 Interrupt Daisy-Chain Operation

In the Z80 peripherals, both the IP and IUS bits control the IEO line and the lower portion of the daisy chain.

When a peripheral's IP bit is set, its IEO line is forced Low. This is true regardless of the state of the IEI line. Additionally, if the peripheral's IUS bit is clear and its IEI line High, the $\overline{INT}$ line is also forced Low.

The Z80 peripherals sample for both $\overline{M1}$ and $\overline{IORQ}$ active, and $\overline{RD}$ inactive to identify an Interrupt Acknowledge cycle. When $\overline{M1}$ goes active and $\overline{RD}$ is inactive, the peripheral detects an Interrupt Acknowledge cycle and allows its interrupt daisy chain to settle. When the $\overline{IORQ}$ line goes active with $\overline{M1}$ active, the highest priority interrupting peripheral places its interrupt vector onto the data bus. The IUS bit is also set to indicate that the peripheral is currently under service. As long as the IUS bit is set, the IEO line is forced Low. This inhibits any lower priority devices from requesting an interrupt.

When the Z80 CPU executes the RETI instruction, the peripherals monitor the data bus and the highest priority device under service resets its IUS bit.

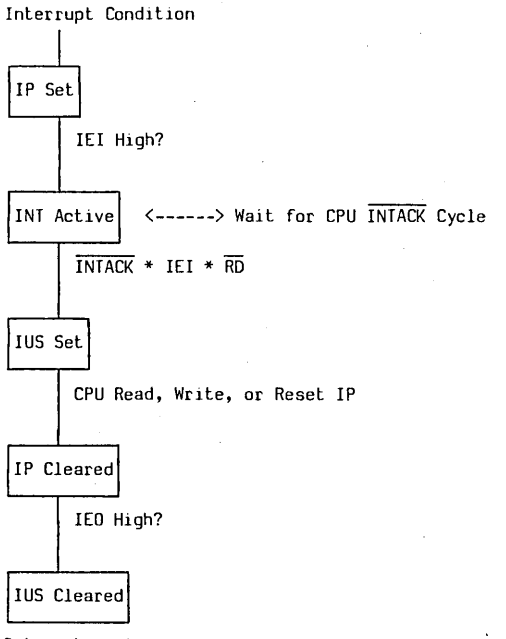
Z8500 Interrupt Daisy-Chain Operation

In the Z8500 peripherals, the IUS bit normally controls the state of the IEO line. The IP bit affects the daisy chain only during an Interrupt Acknowledge cycle. Since the IP bit is normally not part of the Z8500 peripheral interrupt daisy chain, there is no need to decode the RETI instruction. To allow for control over the daisy chain, Z8500 peripherals have a Disable Lower Chain (DLC) software command that pulls IEO Low. This can be used to selectively deactivate parts of the daisy chain regardless of the interrupt status. Table 1 shows the truth tables for the Z8500 interrupt daisy-chain control signals during certain cycles. Table 2 shows the interrupt state diagram for the Z8500 peripherals.

Table 1. Z8500 Daisy-Chain Control Signals

Truth Table for Daisy Chain Signals During Idle State				Truth Table for Daisy Chain Signals During $\overline{INTACK}$ Cycle			
IEI	IP	IUS	IEO	IEI	IP	IUS	IEO
0	X	X	0	0	X	X	0
1	X	0	1	1	1	X	0
1	X	1	0	1	X	1	0
				1	0	0	1

Table 2. Z8500 Interrupt State Diagram



The Z8500 peripherals use $\overline{\text{INTACK}}$ (Interrupt Acknowledge) for recognition of an Interrupt Acknowledge cycle. This pin, used in conjunction with $\overline{\text{RD}}$, allows the Z8500 peripheral to gate its interrupt vector onto the data bus. An active $\overline{\text{RD}}$ signal during an Interrupt Acknowledge cycle performs two functions. First, it allows the highest priority device requesting an interrupt to place its interrupt vector on the data bus. Secondly, it sets the IUS bit in the highest priority device to indicate that the device is currently under service.

INPUT/OUTPUT CYCLES

Although Z8500 peripherals are designed to be as universal as possible, certain timing parameters differ from the standard Z80 timing. The following sections discuss the I/O interface for each of the Z80 CPUs and the Z8500 peripherals. Figure 5 depicts logic for the Z80A CPU to Z8500 peripherals (and Z80B CPU to Z8500A peripherals) I/O interface as well as the Interrupt Acknowledge interface. Figures 4 and 7 depict some of the logic used to interface the Z80H CPU to the Z8500 and Z8500A peripherals for the I/O and Interrupt Acknowledge interfaces. The logic required for adding additional Wait states into the timing flow is not discussed in the folowing sections.

Z80A CPU to Z8500 Peripherals

No additional Wait states are necessary during the I/O cycles, although additional Wait states can be inserted to compensate for timing delays that are inherent in a system. Although the Z80A timing parameters indicate a negative value for data valid prior to $\overline{\text{WR}}$, this is a worse than "worst case" value. This parameter is based upon the longest (worst case) delay for data available from the falling edge of the CPU clock minus the shortest (best case) delay for CPU clock High to $\overline{\text{WR}}$ Low. The negative value resulting from these two parameters does not occur because the worst case of one parameter and the best case of the other do not occur within the same device. This indicates that the value for data available prior to $\overline{\text{WR}}$ will always be greater than zero.

All setup and pulse width times for the Z8500 peripherals are met by the standard Z80A timing. In determining the interface necessary, the $\overline{\text{CE}}$ signal to the Z8500 peripherals is assumed to be the decoded address qualified with the $\overline{\text{IORQ}}$ signal.

Figure 3a shows the minimum Z80A CPU to Z8500 peripheral interface timing for I/O cycles. If additional Wait states are needed, the same number of Wait states can be inserted for both I/O Read and Write cycles to simplify interface logic. There are several ways to place the Z80A CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500 I/O cycles. Tables 3 and 4 list the Z8500 peripheral and the Z80A CPU timing parameters (respectively) of concern during the I/O cycles. Tables 5 and 6 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 3 and 4 refer to the timing diagram in Figure 3a.

Table 3. Z8500 Timing Parameters I/O Cycles

	Worst Case		Min	Max	Units
6.	TsA(WR)	Address to $\overline{WR}$ Low Setup	80		ns
1.	TsA(RD)	Address to $\overline{RD}$ Low Setup	80		ns
2.	TdA(DR)	Address to Read Data Valid		590	ns
	TsCE1(WR)	$\overline{CE}$ Low to $\overline{WR}$ Low Setup	0		ns
	TsCE1(RD)	$\overline{CE}$ Low to $\overline{RD}$ Low Setup	0		ns
4.	TwRD1	$\overline{RD}$ Low Width	390		ns
8.	TwWR1	$\overline{WR}$ Low Width	390		ns
3.	TdRDf(DR)	$\overline{RD}$ Low to Read Data Valid		255	ns
7.	TsDW(WR)	Write Data to $\overline{WR}$ Low Setup	0		ns

Table 4. Z80A Timing Parameters I/O Cycles

	Worst Case		Min	Max	Units
	TcC	Clock Cycle Period	250		ns
	TwCh	Clock Cycle High Width	110		ns
	TfC	Clock Cycle Fall Time		30	ns
	TdCr(A)	Clock High to Address Valid		110	ns
	TdCr(RDf)	Clock High to $\overline{RD}$ Low		85	ns
	TdCr(IORQf)	Clock High to $\overline{IORQ}$ Low		75	ns
	TdCr(WRf)	Clock High to $\overline{WR}$ Low		65	ns
5.	TsD(Cf)	Data to Clock Low Setup	50		ns

Table 5. Parameter Equations

Z8500 Parameter	Z80A Equation	Value	Units
TsA(RD)	TcC-TdCr(A)	140 min	ns
TdA(DR)	3TcC+TwCh-TdCr(A)-TsD(Cf)	800 min	ns
TdRDf(DR)	2TcC+TwCh-TsD(Cf)	460 min	ns
TwRD1	2TcC+TwCh+TfC-TdCr(RDf)	525 min	ns
TsA(WR)	TcC-TdCr(A)	140 min	ns
TsDW(WR)		> 0 min	ns
TwWR1	2TcC+TwCh+TfC-TdCr(WRf)	560 min	ns

Table 6. Parameter Equations

Z80A Parameter	Z8500 Equation	Value	Units
TsD(Cf)	Address 3TcC+TwCh-TdCr(A)-TdA(DR) $\overline{RD}$ 2TcC+TwCh-TdCr(RDf)-TdRD(DR)	160 min 135 min	ns ns

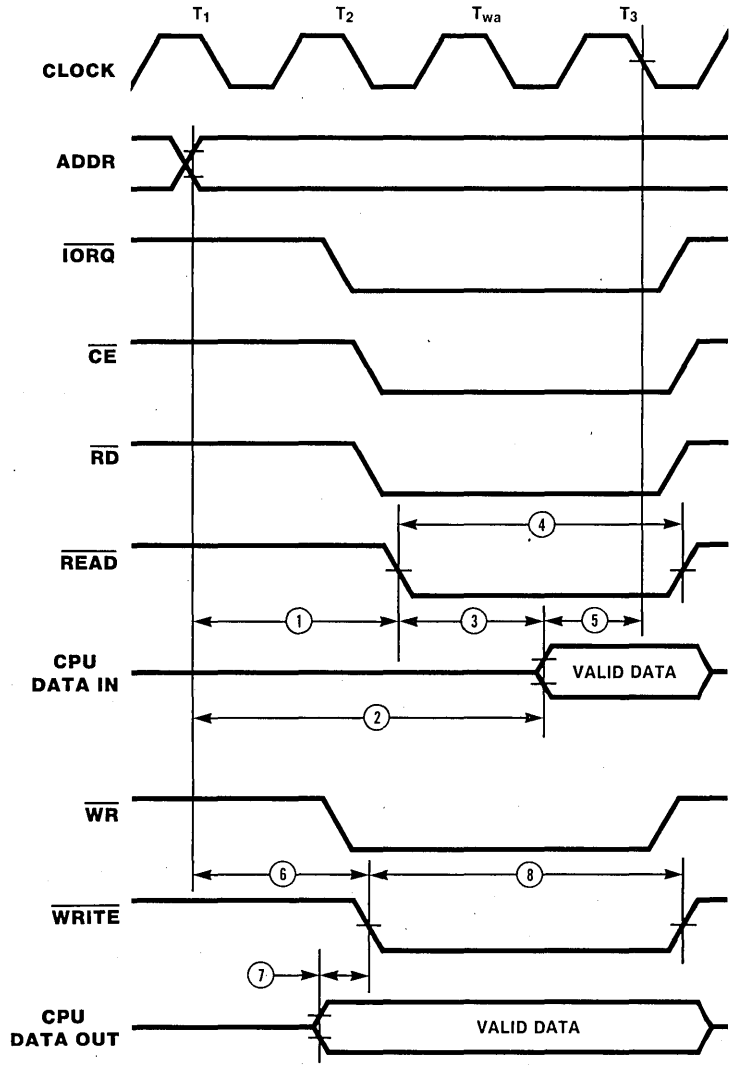


Figure 3a. Z80A CPU to Z8500 Peripheral Minimum I/O Cycle Timing

Z80B CPU to Z8500A Peripherals

No additional Wait states are necessary during I/O cycles, although Wait states can be inserted to compensate for any system delays. Although the Z80B timing parameters indicate a negative value for data valid prior to $\overline{WR}$, this is a worse than "worst case" value. This parameter is based upon the longest (worst case) delay for data available from the falling edge of the CPU clock minus the shortest (best case) delay for CPU clock High to $\overline{WR}$ Low. The negative value resulting from these two parameters does not occur because the worst case of one parameter and the best case of the other do not occur within the same device. This indicates that the value for data available prior to $\overline{WR}$ will always be greater than zero.

All setup and pulse width times for the Z8500A peripherals are met by the standard Z80B timing. In determining the interface necessary, the $\overline{CE}$ signal to the Z8500A peripherals is assumed to be the decoded address qualified with the $\overline{IORQ}$ signal.

Figure 3b shows the minimum Z80B CPU to Z8500A peripheral interface timing for I/O cycles. If additional Wait states are needed, the same number of Wait states can be inserted for both I/O Read and I/O Write cycles in order to simplify interface logic. There are several ways to place the Z80B CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500A I/O cycles. Tables 7 and 8 list the Z8500A peripheral and the Z80B CPU timing parameters (respectively) of concern during the I/O cycles. Tables 9 and 10 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 7 and 8 refer to the timing diagram of Figure 3b.

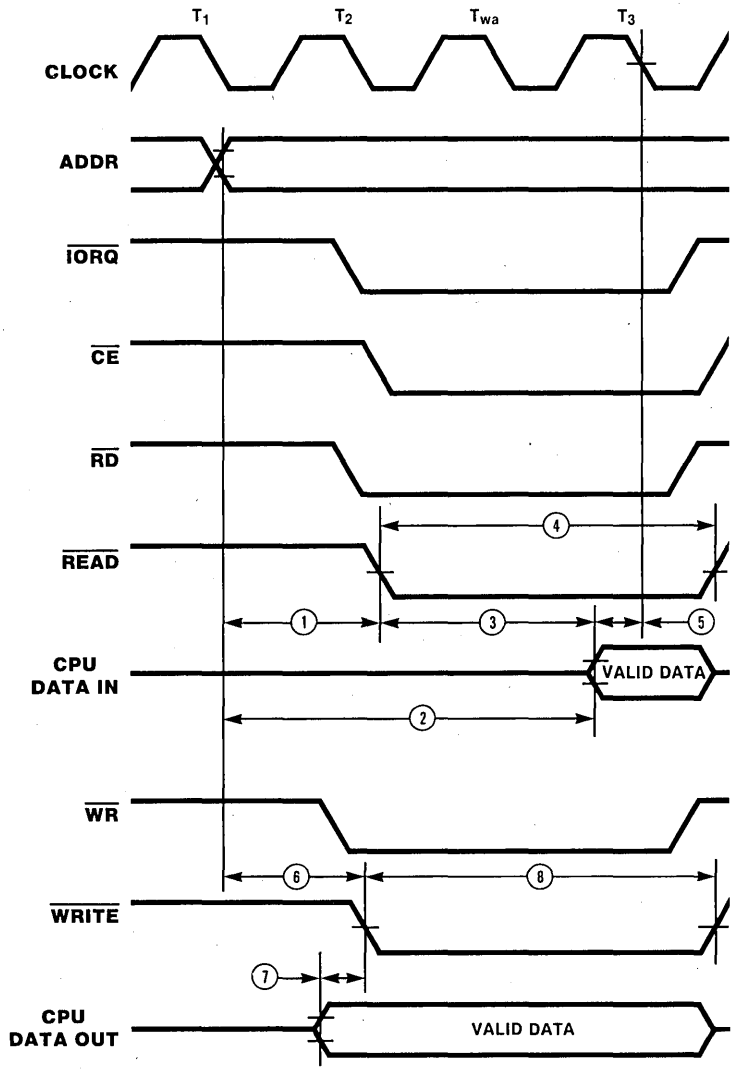


Figure 3b. Z80B CPU to Z8500A Peripheral Minimum I/O Cycle Timing

Table 7. Z8500A Timing Parameters I/O Cycles

	Worst Case		Min	Max	Units
6.	TsA(WR)	Address to $\overline{WR}$ Low Setup	80		ns
1.	TsA(RD)	Address to $\overline{RD}$ Low Setup	80		ns
2.	TdA(DR)	Address to Read Data Valid		420	ns
	TsCE1(WR)	$\overline{CE}$ Low to $\overline{WR}$ Low Setup	0		ns
	TsCE1(RD)	$\overline{CE}$ Low to $\overline{RD}$ Low Setup	0		ns
4.	TwRDl	$\overline{RD}$ Low Width	250		ns
8.	TwWRl	$\overline{WR}$ Low Width	250		ns
3.	TdRDf(DR)	$\overline{RD}$ Low to Read Data Valid		180	ns
7.	TsDW(WR)	Write Data to $\overline{WR}$ Low Setup	0		ns

Table 8. Z80B Timing Parameters I/O Cycles

	Worst Case		Min	Max	Units
	TcC	Clock Cycle Period	165		ns
	TwCh	Clock Cycle High Width	65		ns
	TfC	Clock Cycle Fall Time		20	ns
	TdCr(A)	Clock High to Address Valid		90	ns
	TdCr(RDf)	Clock High to $\overline{RD}$ Low		70	ns
	TdCr(IORQf)	Clock High to $\overline{IORQ}$ Low		65	ns
	TdCr(WRf)	Clock High to $\overline{WR}$ Low		60	ns
5.	TsD(Cf)	Data to Clock Low Setup	40		ns

Table 9. Parameter Equations

Z8500A Parameter	Z80B Equation	Value	Units
TsA(RD)	TcC-TdCr(A)	>75 min	ns
TdA(DR)	3TcC+TwCh-TdCr(A)-TsD(Cf)	430 min	ns
TdRDf(DR)	2TcC+TwCh-TsD(Cf)	345 min	ns
TwRDl	2TcC+TwCh+TfC-TdCr(RDf)	325 min	ns
TsA(WR)	TcC-TdCr(A)	75 min	ns
TsDW(WR)		> 0 min	ns
TwWRl	2TcC+TwCh+TfC-TdCr(WRf)	352 min	ns

Table 10. Parameter Equations

Z80B Parameter	Z8500A Equation	Value	Units
TsD(Cf)	Address		
	3TcC+TwCh-TdCr(A)-TdA(DR)	50 min	ns
	$\overline{RD}$		
	2TcC+TwCh-TdCr(RDf)-TdRD(DR)	75 min	ns

Z80H CPU to Z8500 Peripherals

During an I/O Read cycle, there are three Z8500 parameters that must be satisfied. Depending upon the loading characteristics of the $\overline{RD}$ signal, the designer may need to delay the leading (falling) edge of $\overline{RD}$ to satisfy the Z8500 timing parameter TsA(RD) (Address Valid to $\overline{RD}$ Setup). Since Z80H timing parameters indicate that the $\overline{RD}$ signal may go Low after the falling edge of T_2, it is recommended that the rising edge of the system clock be used to delay $\overline{RD}$ (if necessary). The CPU must also be placed into a Wait condition long enough to satisfy TdA(DR) (Address Valid to Read Data Valid Delay) and TdRDf(DR) ($\overline{RD}$ Low to Read Data Valid Delay).

During an I/O Write cycle, there are three other Z8500 parameters that must be satisfied. Depending upon the loading characteristics of the $\overline{WR}$ signal and the data bus, the designer may need to delay the leading (falling) edge of $\overline{WR}$ to satisfy the Z8500 timing parameters TsA(WR) (Address Valid to $\overline{WR}$ Setup) and TsDW(WR) (Data Valid Prior to $\overline{WR}$ setup). Since Z80H timing parameters indicate that the $\overline{WR}$ signal may go Low after the falling edge of T_2, it is recommended that the rising edge of the system clock be used to delay $\overline{WR}$ (if necessary). This delay will ensure that both parameters are satisfied. The CPU must also be placed into a Wait condition long enough to satisfy TwWRl ($\overline{WR}$ Low Pulse Width). Assuming that the $\overline{WR}$ signal is delayed, only two additional Wait states are needed during an I/O Write cycle when interfacing the Z80H CPU to the Z8500 peripherals.

To simplify the I/O interface, the designer can use the same number of Wait states for both I/O Read and I/O Write cycles. Figure 3c shows the minimum Z80H CPU to Z8500 peripheral interface timing for the I/O cycles (assuming that the same number of Wait states are used for both cycles and that both $\overline{RD}$ and $\overline{WR}$ need to be delayed). Figure 4 shows two circuits that can be used to delay the leading (falling) edge of either the $\overline{RD}$ or the $\overline{WR}$ signals. There are several ways to place the Z80A CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500 I/O cycles. Tables 4 and 11 list the Z8500 peripheral and the Z80H CPU timing parameters (respectively) of concern during the I/O cycles. Tables 14 and 15 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 4 and 11 refer to the timing diagram of Figure 3c.

Table 11. Z80H Timing Parameter I/O Cycles

		Equation	Min	Max	Units
	TcC	Clock Cycle Period	125		ns
	TwCh	Clock Cycle High Width	55		ns
	TfC	Clock Cycle Fall Time		10	ns
	TdCr(A)	Clock High to Address Valid		80	ns
	TdCr(RDf)	Clock High to $\overline{RD}$ Low		60	ns
	TdCr(IORQf)	Clock High to $\overline{IORQ}$ Low		55	ns
	TdCr(WRf)	Clock High to $\overline{WR}$ Low		55	ns
5.	TsD(Cf)	Data to Clock Low Setup	30		ns

Table 12. Parameter Equations

Z8500 Parameter	Z80H Equation	Value	Units
TsA(RD)	2TcC-TdCr(A)	170 min	ns
TdA(DR)	6TcC+TwCh-TdCr(A)-TsD(Cf)	695 min	ns
TdRDf(DR)	4TcC+TwCh-TsD(Cf)	523 min	ns
TwRDl	4TcC+TwCh+TfC-TdCr(RDf)	503 min	ns
TsA(WR)	$\overline{WR}$ - delayed		
	2TcC-TdCr(A)	170 min	ns
TsDW(WR)		> 0 min	ns
TwWRl	4TcC+TwCh+TfC	563 min	ns

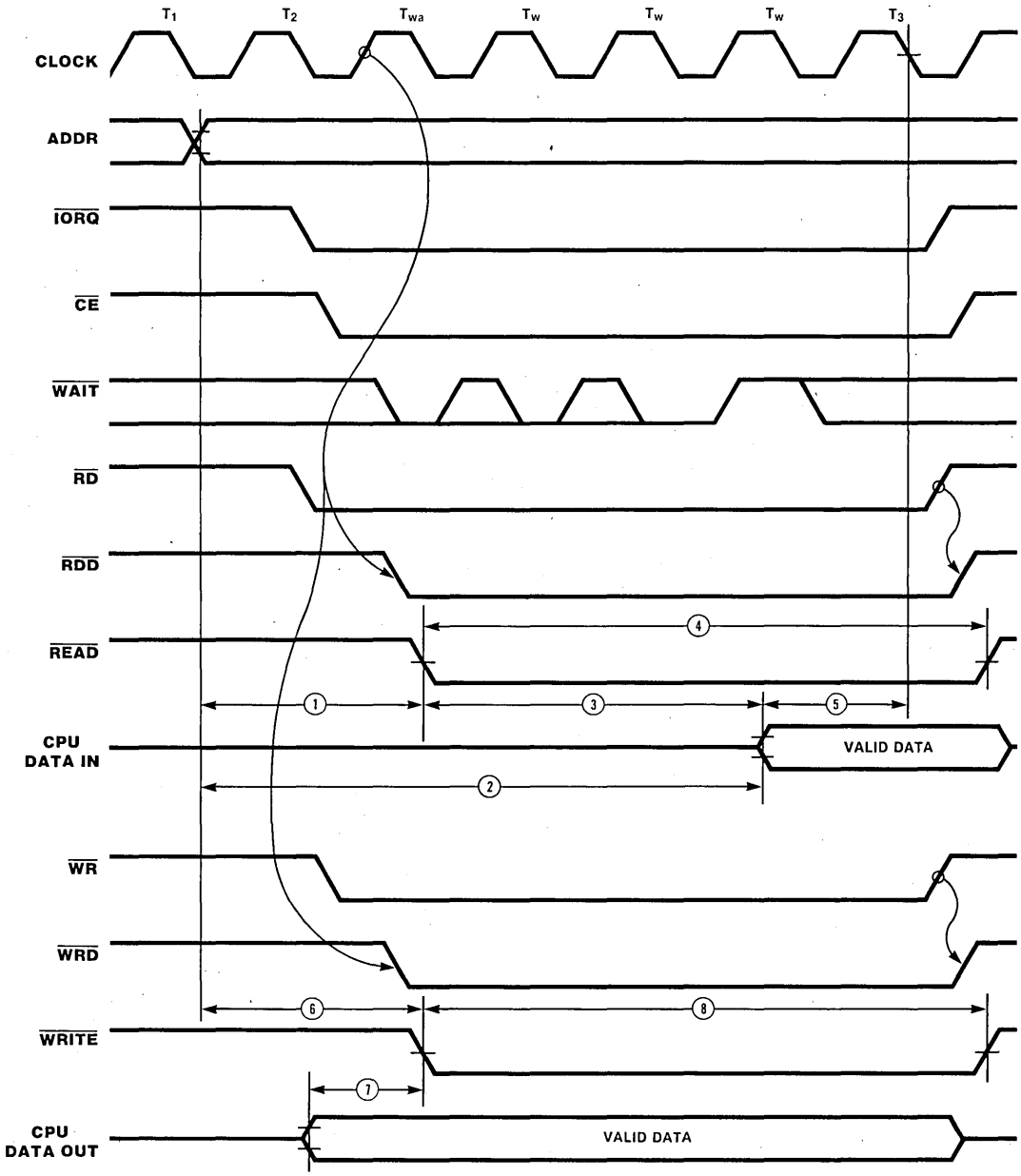


Figure 3c. Z80H CPU to Z8500 Peripheral Minimum I/O Cycle Timing

Z80H CPU to Z8500A Peripherals

During an I/O Read cycle, there are three Z8500A parameters that must be satisfied. Depending upon the loading characteristics of the $\overline{RD}$ signal, the designer may need to delay the leading (falling) edge of $\overline{RD}$ to satisfy the Z8500A timing parameter TsA(RD) (Address Valid to $\overline{RD}$ Setup). Since Z80H timing parameters indicate that the $\overline{RD}$ signal may go Low after the falling edge of T_2, it is recommended that the rising edge of the system clock be used to delay $\overline{RD}$ (if necessary). The CPU must also be placed into a Wait condition long enough to satisfy TdA(DR) (Address Valid to Read Data Valid Delay) and TdRDf(DR) ($\overline{RD}$ Low to Read Data Valid Delay). Assuming that the $\overline{RD}$ signal is delayed, then only one additional Wait state is needed during an I/O Read cycle when interfacing the Z80H CPU to the Z8500A peripherals.

During an I/O Write cycle, there are three other Z8500A parameters that have to be satisfied. Depending upon the loading characteristics of the $\overline{WR}$ signal and the data bus, the designer may need to delay the leading (falling) edge of $\overline{WR}$ to satisfy the Z8500A timing parameters TsA(WR) (Address Valid to $\overline{WR}$ Setup) and TsDW(WR) (Data Valid Prior to $\overline{WR}$ Setup). Since Z80H timing parameters indicate that the $\overline{WR}$ signal may go Low after the falling edge of T_2, it is recommended that the rising edge of the system clock be used to delay $\overline{WR}$ (if necessary). This delay will ensure that both parameters are satisfied. The CPU must also be placed into a Wait condition long enough to satisfy TwWR1 ($\overline{WR}$ Low Pulse Width). Assuming that the $\overline{WR}$ signal is delayed, then only one additional Wait state is needed during an I/O Write cycle when interfacing the Z80H CPU to the Z8500A peripherals.

Figure 3d shows the minimum Z80H CPU to Z8500A peripheral interface timing for the I/O cycles (assuming that the same number of Wait states are used for both cycles and that both $\overline{RD}$ and $\overline{WR}$ need to be delayed). Figure 4 shows two circuits that may be used to delay the leading (falling) edge of either the $\overline{RD}$ or the $\overline{WR}$ signals. There are several methods used to place the Z80A CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500A I/O cycles. Tables 7 and 11 list the Z8500A peripheral and the Z80H CPU timing parameters (respectively) of concern during the I/O cycles. Tables 14 and 15 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 4 and 11 refer to the timing diagram of Figure 3d.

Table 13. Parameter Equations

Z80H Parameter	Z8500 Equation	Value	Units
TsD(Cf)	Address		
	6TcC+TwCh-TdCr(A)-TdA(DR)	135 min	ns
	$\overline{RD}$ - delayed		
	4TcC+TwCh+TfC-TdRD(DR)	300 min	ns

Table 14. Parameter Equations

Z8500A Parameter	Z80H Equation	Value	Units
TsA(RD)	2TcC-TdCr(A)	170 min	ns
TdA(DR)	6TcC+TwCh-TdCr(A)-TsD(Cf)	695 min	ns
TdRDf(DR)	4TcC+TwCh-TsD(Cf)	525 min	ns
TwRDl	4TcC+TwCh+TfC-TdCr(RDf)	503 min	ns
TsA(WR)	$\overline{WR}$ - delayed		
	2TcC-TdCr(A)	170 min	ns
TsDW(WR)		> 0 min	ns
TwWRl	2TcC+TwCh+TfC	313 min	ns

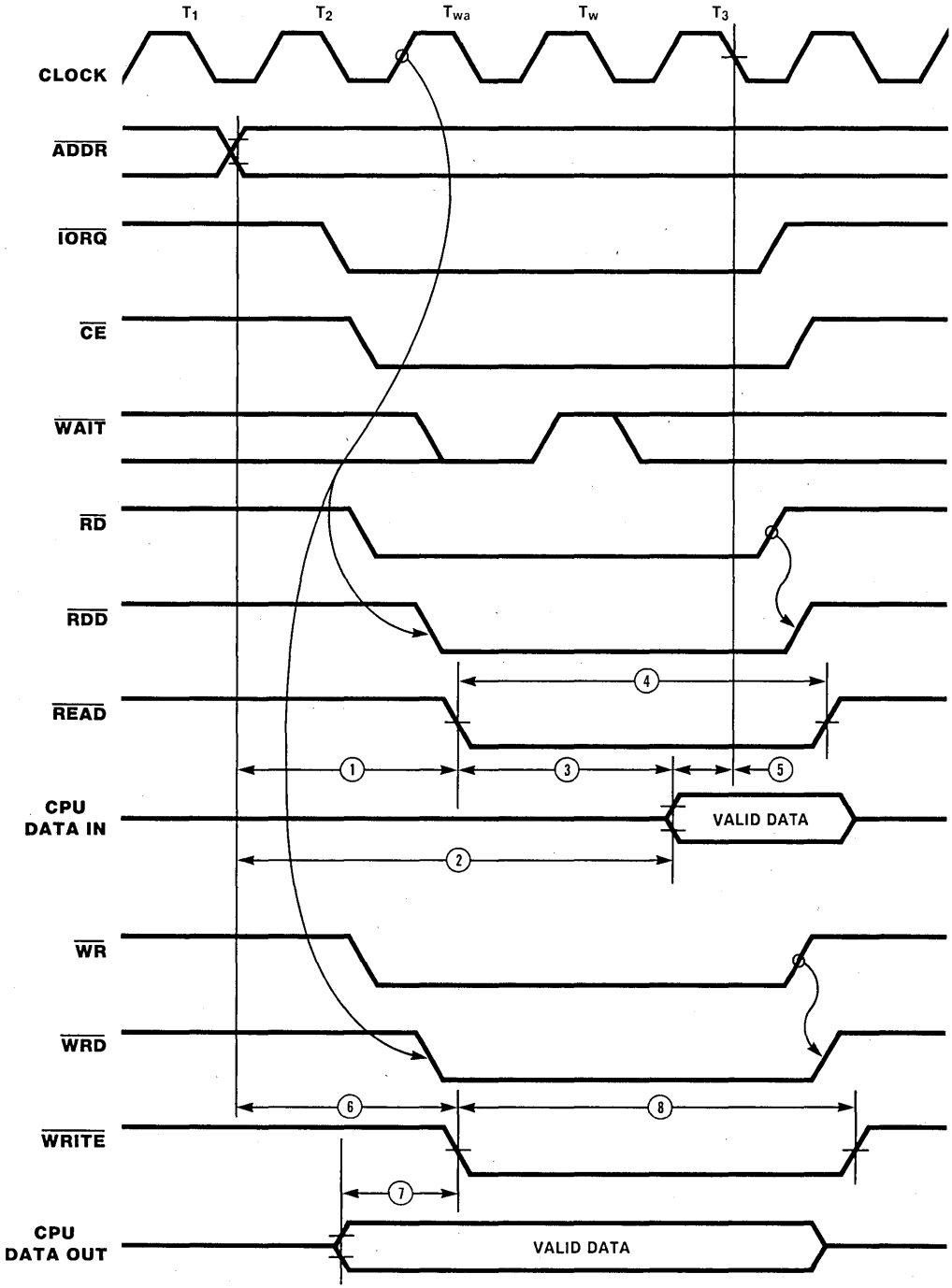


Figure 3d. Z80H CPU to Z8500A Peripheral Minimum I/O Cycle Timing

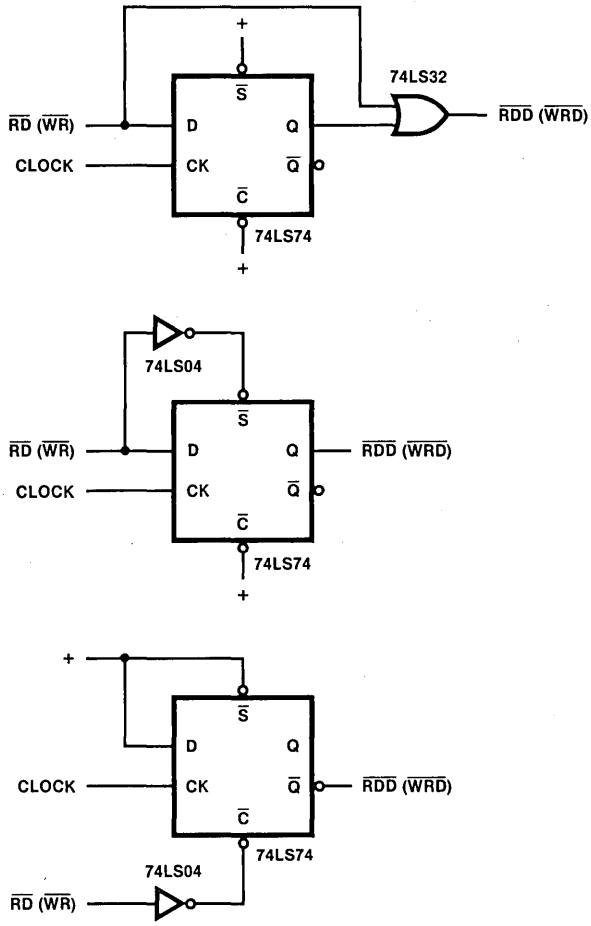


Figure 4. Delaying $\overline{RD}$ or $\overline{WR}$

Table 15. Parameter Equations

Z80H Parameter	Z8500A Equation	Value	Units
TsD(Cf)	Address 4TcC+TwCh-TdCr(A)-TdA(DR) $\overline{RD}$ - delayed 2TcC+TwCh-TdRD(DR)	55 min 125 min	ns ns

INTERRUPT ACKNOWLEDGE CYCLES

The primary timing differences between the Z80 CPUs and Z8500 peripherals occur in the Interrupt Acknowledge cycle. The Z8500 timing parameters that are significant during Interrupt Acknowledge cycles are listed in Table 16, while the Z80 parameters are listed in Table 17. The reference numbers in Tables 16 and 17 refer to Figures 6, 8a, and 8b.

If the CPU and the peripherals are running at different speeds (as with the Z80H interface), the $\overline{INTACK}$ signal must be synchronized to the peripheral clock. Synchronization is discussed in detail under Interrupt Acknowledge for Z80H CPU to Z8500/8500A Peripherals.

During an Interrupt Acknowledge cycle, Z8500 peripherals require both $\overline{INTACK}$ and $\overline{RD}$ to be active at certain times. Since the Z80 CPUs do not issue either $\overline{INTACK}$ or $\overline{RD}$, external logic must generate these signals.

Generating these two signals is easily accomplished, but the Z80 CPU must be placed into a Wait condition until the peripheral interrupt vector is valid. If more peripherals are added to the daisy chain, additional Wait states may be necessary to give the daisy chain time to settle. Sufficient time between $\overline{INTACK}$ active and $\overline{RD}$ active should be allowed for the entire daisy chain to settle.

Since the Z8500 peripheral daisy chain does not use the IP flag except during interrupt acknowledge, there is no need for decoding the RETI instruction used by the Z80 peripherals. In each of the Z8500 peripherals, there are commands that reset the individual IUS flags.

EXTERNAL INTERFACE LOGIC

The following sections discuss external interface logic required during Interrupt Acknowledge cycles for each interface type.

CPU/Peripheral Same Speed

Figure 5 shows the logic used to interface the Z80A CPU to the Z8500 peripherals and the Z80B CPU to Z8500A peripherals during an Interrupt Acknowledge cycle. The primary component in this logic is the Shift register (74LS164), which generates $\overline{INTACK}$, $\overline{READ}$, and $\overline{WAIT}$.

Table 16. Z8500 Timing Parameters Interrupt Acknowledge Cycles

	Worst Case		4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Units
1.	TsIA(PC)	$\overline{INTACK}$ Low to PCLK High Setup	100		100		ns
	ThIA(PC)	$\overline{INTACK}$ Low to PCLK High Hold	100		100		ns
2.	TdIAi(RD)	$\overline{INTACK}$ Low to RD (Acknowledge) Low	350		250		ns
5.	TwRDA	$\overline{RD}$ (Acknowledge) Width	350		250		ns
3.	TdRDA(DR)	$\overline{RD}$ (Acknowledge) to Data Valid		250		180	ns
	TsIEI(RDA)	IEI to $\overline{RD}$ (Acknowledge) Setup	120		100		ns
	ThIEI(RDA)	IEI to $\overline{RD}$ (Acknowledge) Hold	100		70		ns
	TdIEI(IE)	IEI to IEO Delay		150		100	ns

Table 17. Z80 CPU Timing Parameters Interrupt Acknowledge Cycles

	Worst Case		4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	Units
	TdC(M1f)	Clock High to $\overline{M1}$ Low Delay		100		80		70	ns
	TdM1f(IORQf)	$\overline{M1}$ Low to $\overline{IORQ}$ Low Delay	575*		345*		275*		ns
4.	TsD(Cr)	Data to Clock High Setup	35		30		25		ns

*Z80A: 2TcC + TwCh + TfC - 65
Z80B: 2TcC + TwCh + TfC - 50
Z80H: 2TcC + TwCh + TfC - 45

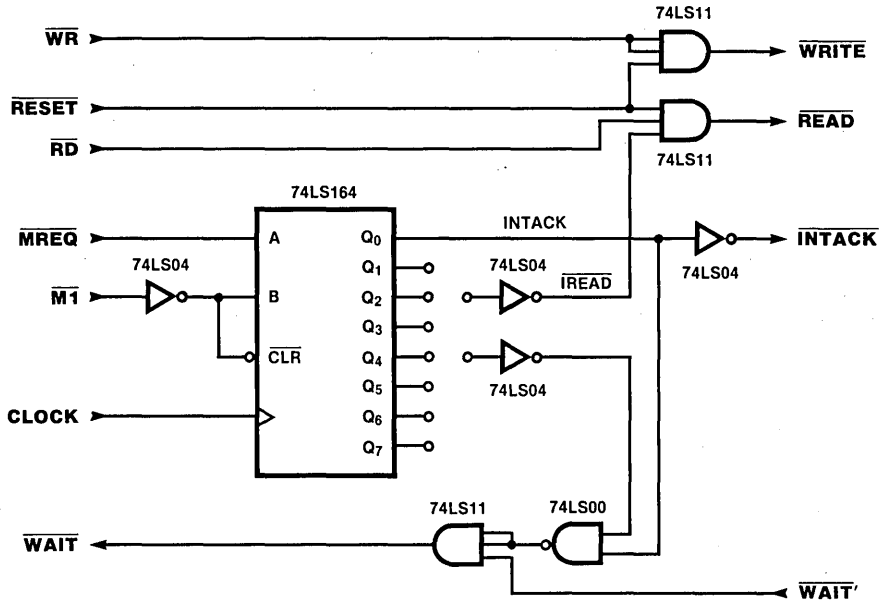


Figure 5. Z80A/Z80B CPU to Z8500/Z8500A Peripheral Interrupt Acknowledge Interface Logic

During I/O and normal memory access cycles, the Shift register remains cleared because the $\overline{MT}$ signal is inactive. During opcode fetch cycles, also, the Shift register remains cleared, because only 0s can be clocked through the register. Since Shift register outputs are Low, $\overline{READ}$, $\overline{WRITE}$, and $\overline{WAIT}$ are controlled by other system logic and gated through the AND gates (74LS11). During I/O and normal memory access cycles, $\overline{READ}$ and $\overline{WRITE}$ are active as a result of the system $\overline{RD}$ and $\overline{WR}$ signals (respectively) becoming active. If system logic requires that the CPU be placed into a Wait condition, the $\overline{WAIT}$' signal controls the CPU. Should it be necessary to reset the system, $\overline{RESET}$ causes the interface logic to generate both $\overline{READ}$ and $\overline{WRITE}$ (the Z8500 peripheral Reset condition).

Normally an Interrupt Acknowledge cycle is indicated by the Z80 CPU when $\overline{MT}$ and $\overline{IORQ}$ are both active (which can be detected on the third rising clock edge after T_1). To obtain an early indication of an Interrupt Acknowledge cycle, the Shift register decodes an active $\overline{MT}$ in the presence of an inactive $\overline{MREQ}$ on the rising edge of T_2.

During an Interrupt Acknowledge cycle, the $\overline{INTACK}$ signal is generated on the rising edge of T_2. Since it is the presence of $\overline{INTACK}$ and an active $\overline{READ}$ that gates the interrupt vector onto the data bus, the logic must also generate $\overline{READ}$ at the proper time. The timing parameter of concern here is TdIAi(RD) [$\overline{INTACK}$ to $\overline{RD}$ (Acknowledge) Low Delay]. This time delay allows the interrupt daisy chain to settle so that the device requesting the interrupt can place its interrupt vector onto the data bus. The Shift register allows a sufficient time delay from the generation of $\overline{INTACK}$ before it generates $\overline{READ}$. During this delay, it places the CPU into a Wait state until the valid interrupt vector can be placed onto the data bus. If the time between these two signals is insufficient for daisy chain settling, more time can be added by taking $\overline{READ}$ and $\overline{WAIT}$ from a later position on the Shift register.

Figure 6 illustrates Interrupt Acknowledge cycle timing resulting from the Z80A CPU to Z8500 peripheral and the Z80B CPU to Z8500A peripheral interface. This timing comes from the logic illustrated in Figure 5, which can be used for both interfaces. Should more Wait states be required, the additional time can be calculated in terms of system clocks, since the CPU clock and PCLK are the same.

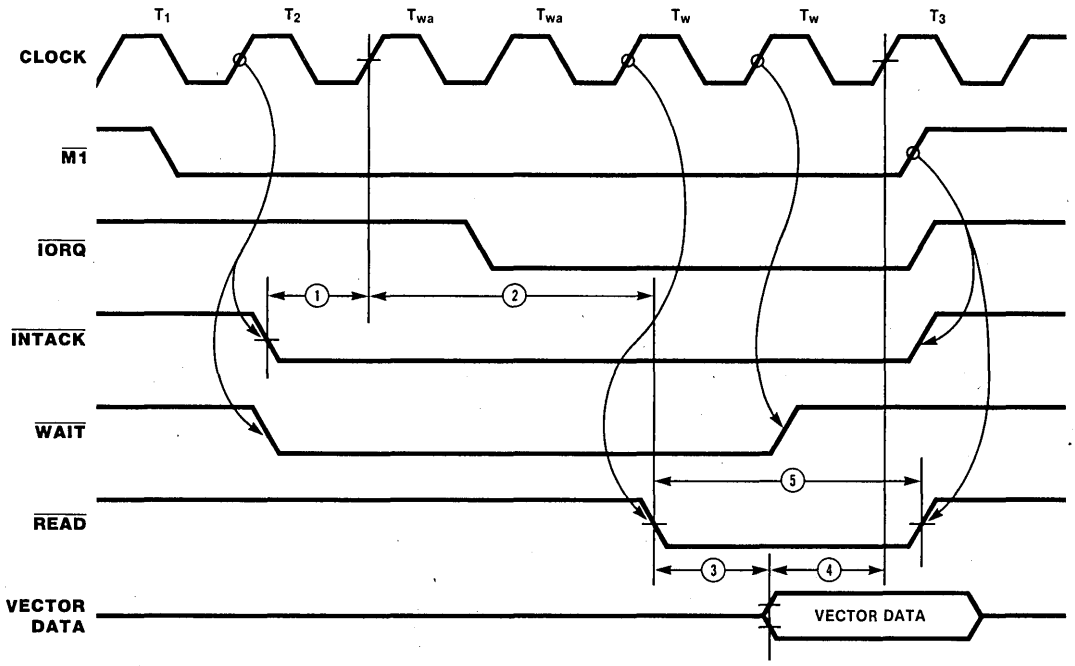


Figure 6. Z80A/Z80B CPU to Z8500/Z8500A Peripheral Interrupt Acknowledge Interface Timing

Z80H CPU to Z8500/Z8500A Peripherals

Figure 7 depicts logic that can be used in interfacing the Z80H CPU to the Z8500/Z8500A peripherals. This logic is the same as that shown in Figure 5, except that a synchronizing flip-flop is used to recognize an Interrupt Acknowledge cycle. Since Z8500 peripherals do not rely upon PCLK except during Interrupt Acknowledge cycles, synchronization need occur only at that time. Since the CPU and the peripherals are running at different speeds, $\overline{INTACK}$ and $\overline{RD}$ must be synchronized to the Z8500 peripherals clock.

During I/O and normal memory access cycles, the synchronizing flip-flop and the Shift register remain cleared because the $\overline{M1}$ signal is inactive. During opcode fetch cycles, the flip-flop and the Shift register again remain cleared, but this time because the $\overline{MREQ}$ signal is active. The synchronizing flip-flop allows an Interrupt Acknowledge cycle to be recognized on the rising edge of T_2 when $\overline{M1}$ is active and $\overline{MREQ}$ is inactive, generating the INTA signal. When INTA is active, the Shift register can clock and generate $\overline{INTACK}$ to the peripheral and $\overline{WAIT}$ to the CPU. The Shift register delays the generation of $\overline{READ}$ to the peripheral until the daisy chain settles. The $\overline{WAIT}$ signal is removed when sufficient time has been allowed for the interrupt vector data to be valid.

Figure 8a illustrates Interrupt Acknowledge cycle timing for the Z80H CPU to Z8500 peripheral interface. Figure 8b illustrates Interrupt Acknowledge cycle timing for the Z80H CPU to Z8500A peripheral interface. These timings result from the logic in Figure 7. Should more Wait states be required, the needed time should be calculated in terms of PCLKs, not CPU clocks.

Z80 CPU to Z80 and Z8500 Peripherals

In a Z80 system, a combination of Z80 peripherals and Z8500 peripherals can be used compatibly. While there is no restriction on the placement of the Z8500 peripherals in the daisy chain, it is recommended that they be placed early in the chain to minimize propagation delays during RETI cycles.

During an Interrupt Acknowledge cycle, the IEO line from the Z8500 peripherals changes to reflect the interrupt status. Time should be allowed for this change to ripple through the remainder of the daisy chain before activating $\overline{IORQ}$' to the Z80 peripherals, or $\overline{READ}$ to the Z8500 peripherals.

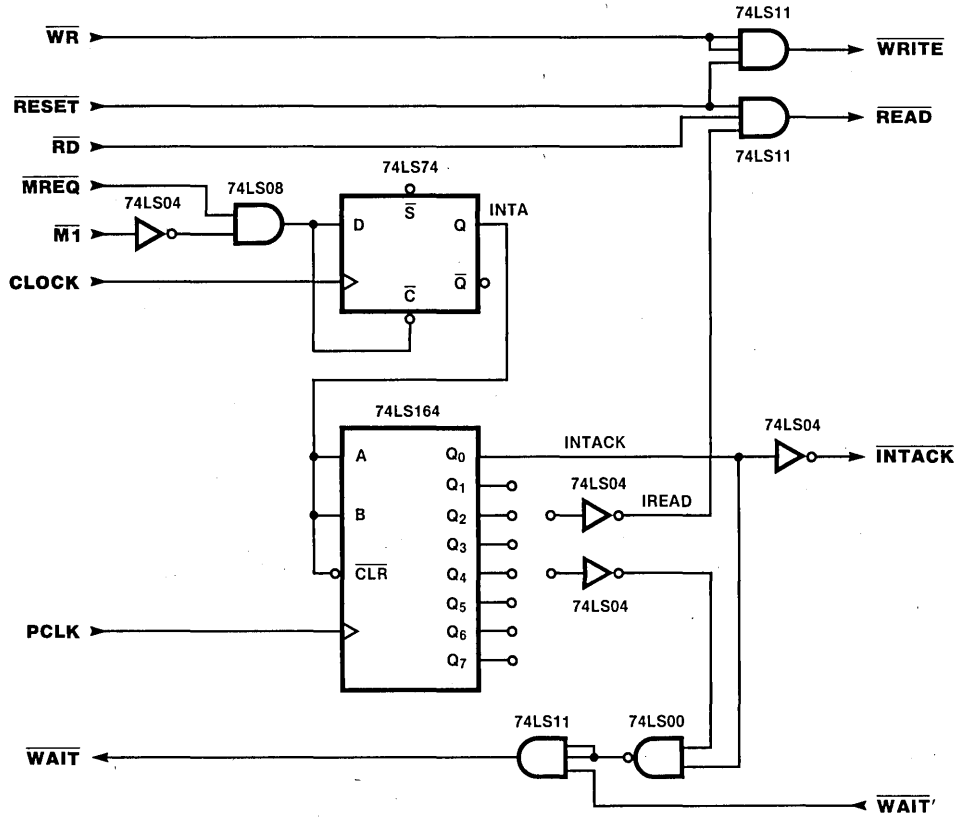


Figure 7. Z80H to Z8500/Z8500A Peripheral Interrupt Acknowledge Interface Logic

During the RETI cycles, the IEO line from the Z8500 peripherals does not change state as in the Z80 peripherals. As long as the peripherals are at the top of the daisy chain, propagation delays are minimized.

The logic necessary to create the control signals for both Z80 and Z8500 peripherals is shown in Figure 9. This logic delays the generation of $\overline{IORQ}'$ to the Z80 peripherals by the same amount of time necessary to generate $\overline{READ}$ for the Z8500 peripherals. Timing for this logic during an Interrupt Acknowledge cycle is depicted in Figure 10.

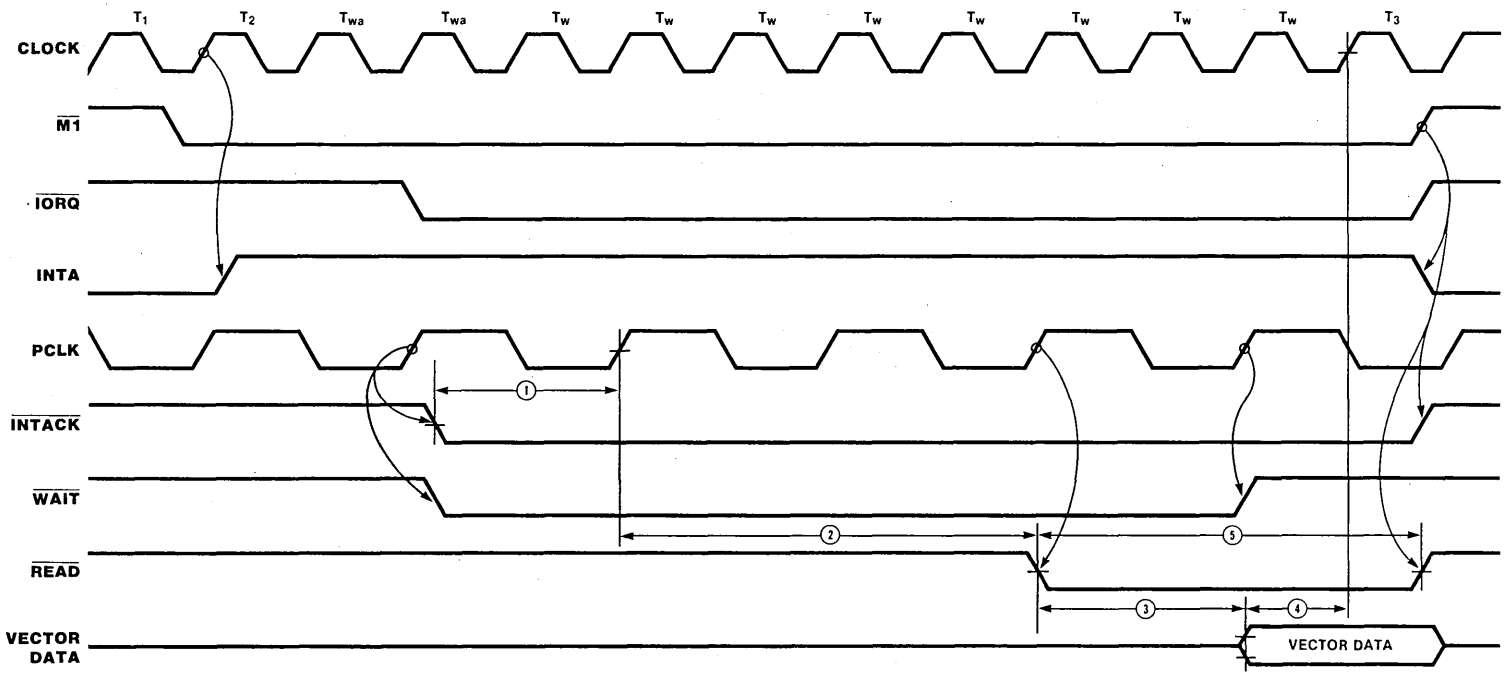


Figure 8a. Z80H CPU to Z8500 Peripheral Interrupt Acknowledge Interface Timing

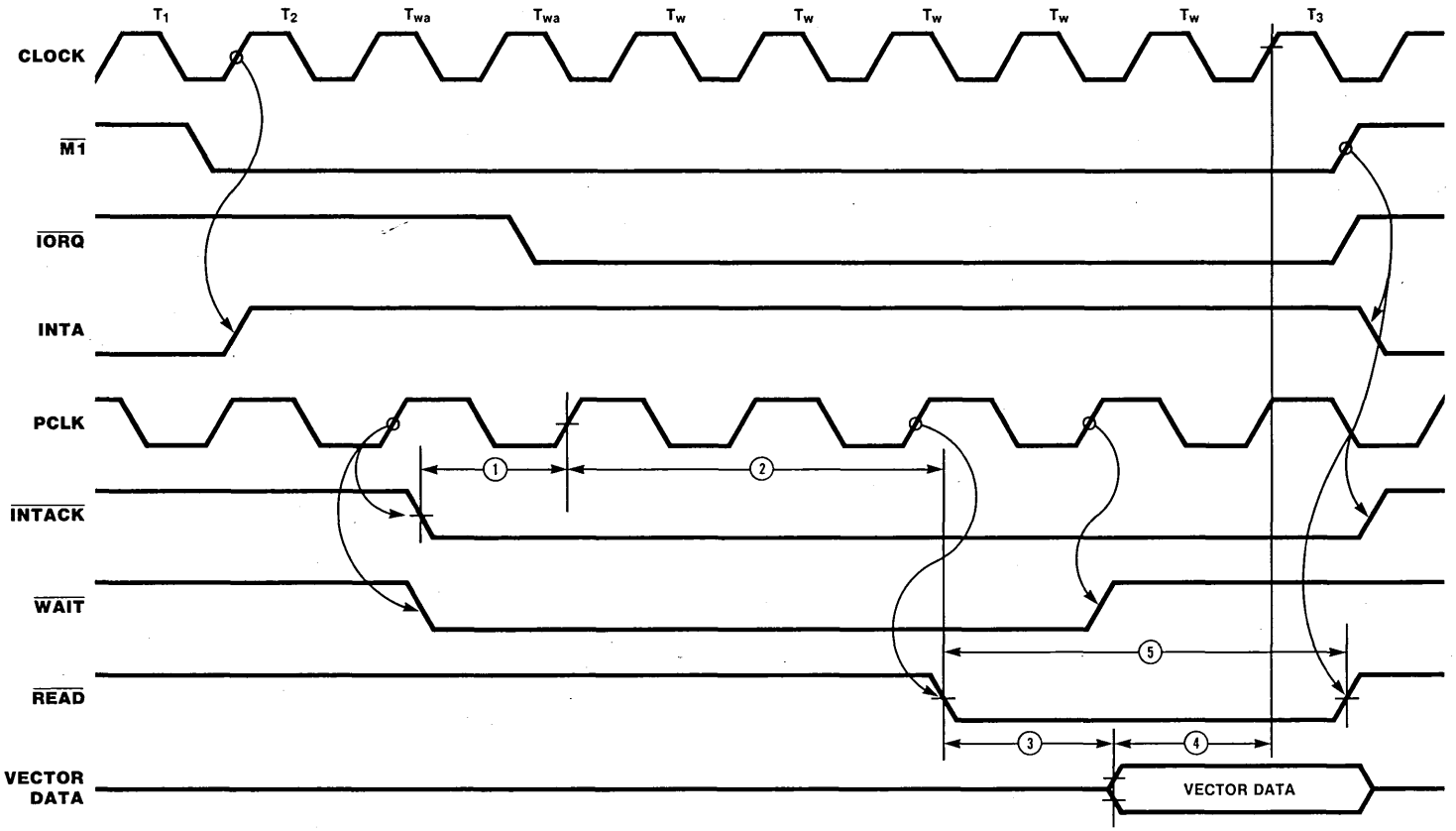


Figure 8b. Z80H CPU to Z8500A Peripheral Interrupt Acknowledge Interface Timing

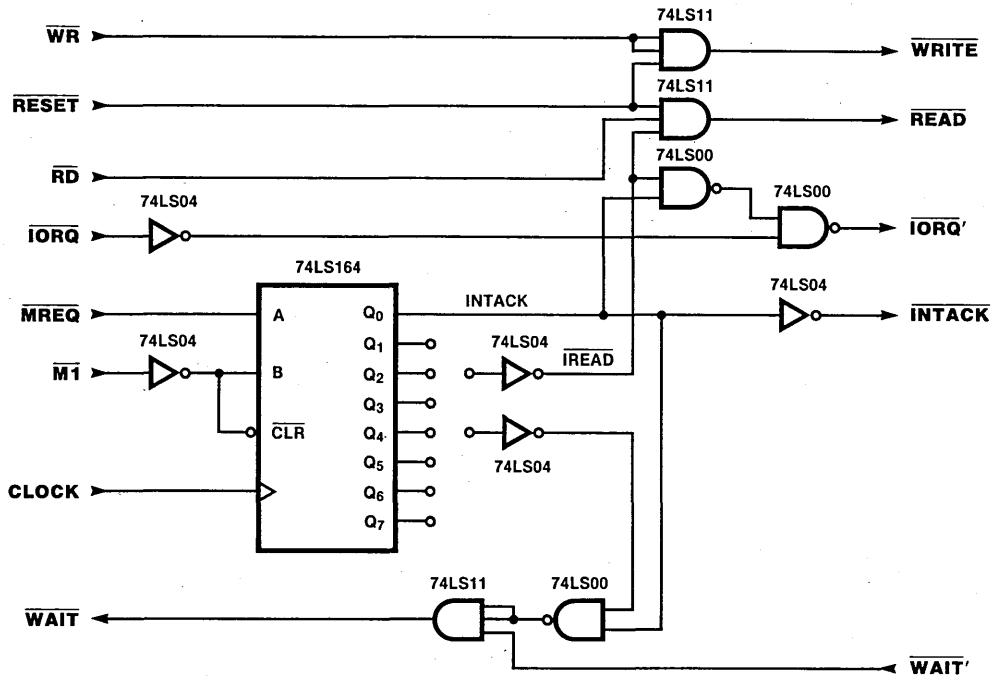


Figure 9. Z80 and Z8500 Peripheral Interrupt Acknowledge Interface Logic

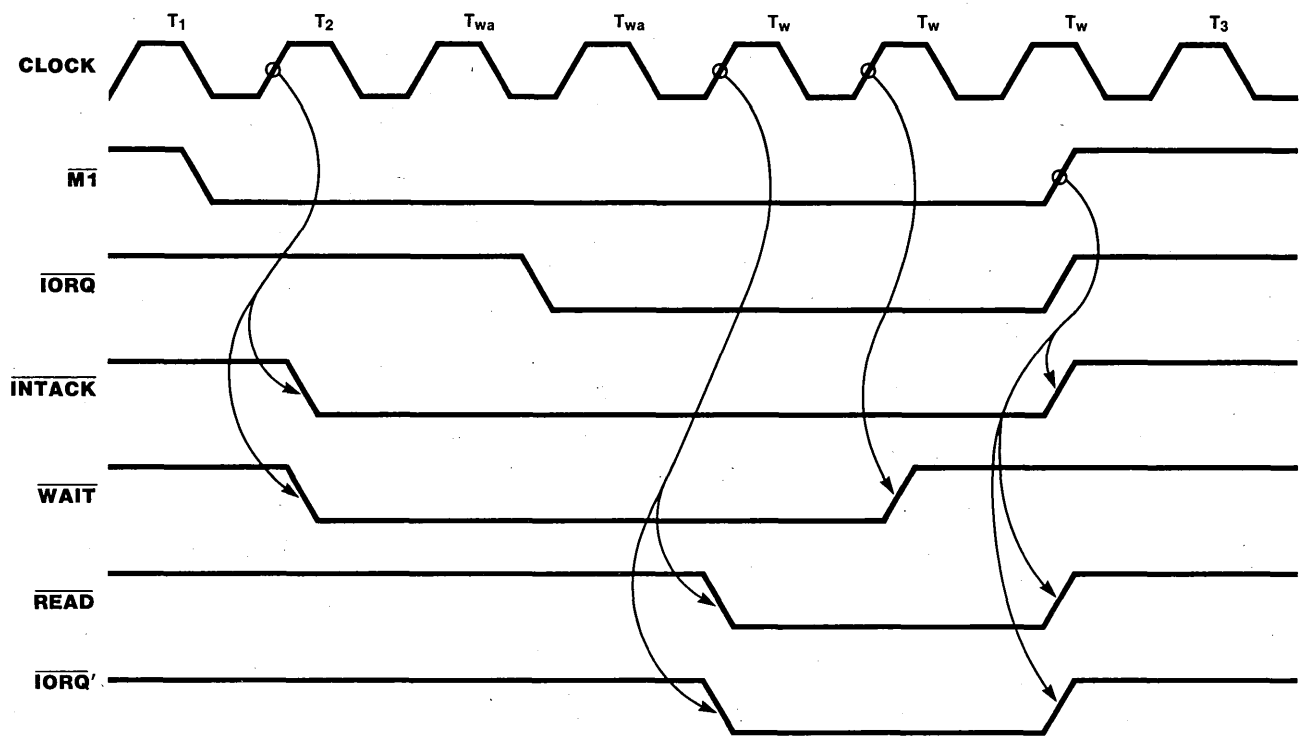


Figure 10. Z80 and Z8500 Peripheral Interrupt Acknowledge Interface Timing

SOFTWARE CONSIDERATIONS -- POLLED OPERATION

There are several options available for servicing interrupts on the Z8500 peripherals. Since the vector or IP registers can be read at any time, software can be used to emulate the Z80 interrupt response. The interrupt vector read reflects the interrupt status condition even if the device is programmed to return a vector that does not reflect the status change (SAV or VIS is not set). The code below is a simple software routine that emulates the Z80 vector response operation.

Z80 Vector Interrupt Response, Emulation by Software

```
;This code emulates the Z80 vector interrupt
;operation by reading the device interrupt
;vector and forming an address from a vector
;table.  It then executes an indirect jump to
;the interrupt service routine.

INDX:   LD      A,CIVREG        ;CURRENT INT. VECT. REG.
        OUT     (CTRL),A        ;WRITE REG. PTR.
        IN      A,(CTRL)        ;READ VECT. REG.
        INC     A               ;VALID VECTOR?
        RET     Z               ;NO INT - RETURN
        AND     00001110B       ;MASK OTHER BITS
        LD      E,A
        LD      D,0             ;FORM INDEX VALUE
        LD      HL,VECTAB
        ADD     HL,DE           ;ADD VECT. TABLE ADDR.
        LD      A,(HL)          ;GET LOW BYTE
        INC     HL
        LD      H,(HL)          ;GET HIGH BYTE
        LD      L,A             ;FORM ROUTINE ADDR.
        JP      (HL)            ;JUMP TO IT

VECTAB: DEFW    INT1
        DEFW    INT2
        DEFW    INT3
        DEFW    INT4
        DEFW    INT5
        DEFW    INT6
        DEFW    INT7
        DEFW    INT8
```

A SIMPLE Z80-Z8500 SYSTEM

The Z8500 devices interface easily to the Z80 CPU, thus providing a system of considerable flexibility. Figure 11 illustrates a simple system using the Z80A CPU and the Z8536 Counter/Timer and Parallel I/O Unit (CIO) in a mode 1 or non-interrupt environment. Since interrupt vectors are not used, the $\overline{\text{INTACK}}$ line is tied High and no additional logic is needed. Because the CIO can be used in a polled interrupt environment, the $\overline{\text{INT}}$ pin is connected to the CPU. The Z80 should not be set for mode 2 interrupts since the CIO will never place a vector onto the data bus. Instead, the CPU should be placed into mode 1 interrupt mode and a global interrupt service routine can poll the CIO to determine what caused the interrupt to occur. In this system, the software emulation procedure described above is effective.

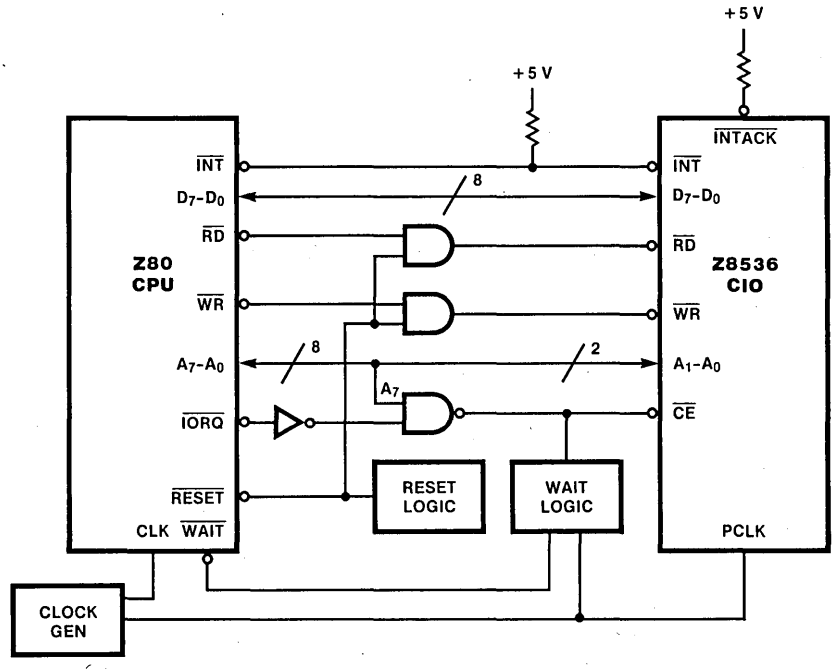


Figure 11. Z80 to Z8500 Simple System Mode 1 Interrupt or Non-Interrupt Structure

Additional Information - Zilog Publications

1. Z80 CPU Technical Manual (03-0029-01)
2. Z80 DMA Technical Manual (00-2013-A0)
3. Z80 PIO Technical Manual (03-0008-01)
4. Z80 CTC Technical Manual (03-0036-02)
5. Z80 SIO Technical Manual (03-3033-01)
6. Z80H CPU AC Characteristics (00-2293-01)
7. Z80 Family Interrupt Structure Tutorial (611-1809-0003)
8. Z8530 SCC Technical Manual (00-2057-01)
9. Z8536 CIO Technical Manual (00-2091-01)
10. Z8038 FIO Technical Manual (00-2051-01)

Interfacing the Z8500 Peripherals to the 68000

October 1988

INTRODUCTION

This application note discusses interfacing Zilog's Z8500 family of peripherals to the 68000 microprocessor. The Z8500 peripheral family includes the Z8536 Counter/Timer and Parallel I/O Unit (CIO), the Z8038 FIFO Input/Output Interface Unit (FIO), and the Z8530 Serial Communications Controller (SCC). This document discusses the Z8500/68000 interfaces and presents hardware examples and verification techniques. One of the three hardware examples given in this application note shows how to implement the Z8500/68000 interface using a single-chip programmable logic array (PAL).

This application note about interfacing supplements the following documents, which discuss the individual components of the interface.

- Z8036 Z-CIO/Z8536 CIO Technical Manual (document number 00-2091-01)
- Z8038 Z-FIO Technical Manual (document number 00-2051-01)
- Z8030/Z8530 SCC Technical Manual (document number 00-2057-01)
- Motorola 16-Bit Microprocessor User's Manual 3rd ed. Englewood Cliffs, N.J., Prentice-Hall, Inc. 1979.
- Monolithic Memories Bipolar LSI 1982 Databook

This application note is divided into four sections. The first section gives a general description of the Z8500 family and discusses pin functions, interrupt structures, and the programming of operating modes. The second section discusses the Z8500 interface itself. It shows how the different Z8500 control signals are generated from the 68000 signals and summarizes the critical timings for the three types of bus cycle. The third section shows three examples of implementing the 68000-to-Zilog-peripheral interface. The fourth section suggests methods of verifying the interface design by checking the three different types of bus cycle: Read, Write, and Interrupt Acknowledge.

GENERAL Z8500 FAMILY DESCRIPTION

The Z8500 family is made up of programmable peripherals that can interface easily to the bus of any nonmultiplexed CPU microprocessor, such as the 68000. The three members of this family, the CIO, SCC, and FIO, can solve many design problems. The peripherals' operating modes can be programmed simply by writing to their internal registers.

Programming the Operating Modes

The CPU can access two types of register: Control and Data. Depending on the peripheral, registers are selected with either the A_0, A_1, $A/\overline{B}$, or $D/\overline{C}$ function pins.

Peripheral operating modes are initialized by programming internal registers. Since these registers are not directly addressable by the CPU, a two-step procedure using the Control register is required: first, the address of the internal register is written to the Control register, then the data is written to the Control register. A state machine determines whether an address or data is being written to the Control register. Reading an internal register follows a similar two-step

procedure: first, the address is written, then the data is read.

The Data registers that are most frequently accessed, for example, the SCC's transmit and receive buffer, can be addressed directly by the CPU with a single read or write operation. This reduces overhead in data transfers between the peripheral and CPU.

GENERATING Z8500 CONTROL SIGNALS

This section shows how to generate the Z8500 control signals. To simplify the discussion, the section is divided into two parts. The first part takes each individual Z8500 signal and shows how it is generated from the 68000 signals. The second part discusses the Z8500 timing that must be met when generating the control signals.

Z8500 Signal Generation

The right-hand side of Table 1 lists the Z8500 signals that must be generated. Each of these signals is discussed in a separate paragraph.

A_0, A_1, $A/\overline{B}$, $D/\overline{C}$. These pins are used to select the peripheral's Control and Data registers that program the different operating modes. They can be connected to the 68000 A_1 and A_2 Address bus lines.

$\overline{CE}$. Each peripheral has an active Low Chip Enable that can be derived by ANDing the selected address decode and the 68000's Address Strobe ($\overline{AS}$). The active Low $\overline{AS}$ guarantees that the 68000 addresses are valid.

D_0-D_7. The Z8500 Data bus can be directly connected to the lowest byte (D_0-D_7) of the 68000 Data bus.

IEI and IEO. The peripherals use these pins to decide the interrupt priority. The highest priority device should have its IEI tied High. Its IEO should be connected to the IEI pin of the next highest priority device. This pattern continues with the next highest priority peripheral, until the peripherals are all connected, as shown in Figure 1.

$\overline{INT}$. The interrupt request pins for each peripheral in the daisy chain can be wire-ORed and connected to the 68000's ILP_n pins. The 68000 has seven interrupt levels that can be encoded into the ILP_0, ILP_1, and ILP_2 pins. Multiple 68000 interrupt levels can be implemented by using a multiplexer like the 74LS148.

Table 1. Z8500 and 68000 Pin Functions

68000 Signals		Z8500 Signals	
Mnemonic	**Function**	**Mnemonic**	**Function**
A_1-A_{23}	Address bus	A_0,A_1,$A/\overline{B}$,$D/\overline{C}$*	Register select
$\overline{AS}$	Address Strobe	$\overline{CE}$	Chip Enable
CLK	68000 clock (8 MHz)	D_0-D_7	Data bus
D_0-D_{15}	Data bus	IEI,IEO	Interrupt daisy chain
$\overline{DTACK}$	Data Transfer Acknowledge		control
FC_0-FC_2	Processor status	$\overline{INT}$	Interrupt Request
ILP_0-ILP_2	Interrupt request	$\overline{INTACK}$	Interrupt Acknowledge
$R/\overline{W}$	Read/Write	PCLK	Peripheral Clock
$\overline{VMA}$	Valid Memory Address	$\overline{RD}$	Read strobe
$\overline{VPA}$	Valid Peripheral Address	$\overline{WR}$	Write strobe

* The register select pins on each peripheral have different names.

$\overline{\text{INTACK}}$. The INTACK pin signals the peripheral that an Interrupt Acknowledge cycle is occurring. The following equation describes how $\overline{\text{INTACK}}$ is generated:

$$\overline{\text{INTACK}} = \overline{(FC_0)\cdot(FC_1)\cdot(FC_2)\cdot(AS)}$$

The 68000 FC_0-FC_2 are status pins that indicate an Interrupt Acknowledge when they are all High. They should be ANDed with inverted $\overline{AS}$ to guarantee their validity. The $\overline{\text{INTACK}}$ signal must be synchronized with PCLK to guarantee set-up and hold times. This can be accomplished by changing the state of $\overline{\text{INTACK}}$ on the falling edge of PCLK. If the $\overline{\text{INTACK}}$ pin is not used, it must be tied High.

PCLK. The SCC and CIO require a clock for internal synchronization. The clock can be generated by dividing down the 68000 CLK.

$\overline{RD}$. The Read strobe goes active Low under three conditions: hardware reset, normal Read cycle, and an Interrupt Acknowledge cycle. The following equation describes how $\overline{RD}$ is generated:

$$\overline{RD} = \overline{[(R/\overline{W})\cdot(AS) + RESET]}$$

The Read strobe timing must meet both the Read timing and Interrupt Acknowledge timing discussed in the following section. In addition to enabling the Data bus drivers, the falling edge of $\overline{RD}$ sets the Interrupt Under Service (IUS) bits during an Interrupt Acknowledge cycle.

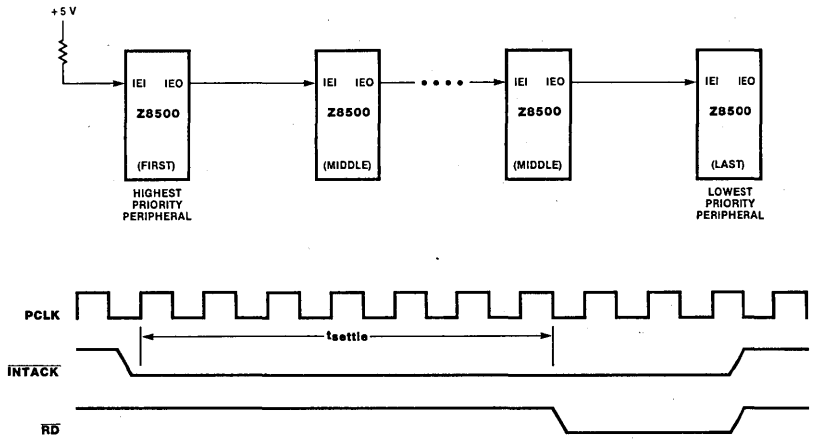
$\overline{WR}$. This signal strobes data into the peripheral. A data-to-write setup time requires that data be valid before $\overline{WR}$ goes active Low. The equation for generating the $\overline{WR}$ strobe is made up of two components: an active reset and a normal Write cycle, as shown in the following equation:

$$\overline{WR} = \overline{[(R/\overline{W})\cdot(AS) + RESET]}$$

Forcing $\overline{RD}$ and $\overline{WR}$ simultaneously Low resets the peripherals.

Z8500 Timing Cycles

This section discusses the timing parameters that must be met when generating the control signals. The Z8500 family uses the control signals to communicate with the CPU via three types of bus cycle: Read, Write, and Interrupt Acknowledge.



tsettle (NS)

PERIPHERAL (4MHz)	FIRST	MIDDLE	LAST
CIO	350	150	100
FIO	350	150	100
SCC	250	120	120

Figure 1. Peripheral Interrupt Daisy Chain

The discussion that follows pertains to the 4 MHz peripherals, but the 6 MHz devices have similar timing considerations.

Although the peripherals have a standard CPU interface, some of their particular timing requirements vary. The worst-case parameters are shown below; the timing can be optimized if only one or two of the Z8500 family devices are used.

Read Cycle

The Read cycle transfers data from the peripheral to the CPU. It begins by selecting the peripheral and appropriate register (Data or Control). The data is gated onto the bus with the $\overline{RD}$ line. A setup time of 80 ns from the time the register select inputs ($A/\overline{B}$, $C/\overline{D}$, A_0, A_1) are stable to the falling edge of $\overline{RD}$ guarantees that the proper register is accessed. The access time specification is usually measured from the falling edge of $\overline{RD}$ to valid data and varies between peripherals. The SCC specifies an additional register select to valid data time. The Read cycle timing is shown in Figure 2.

Write Cycle

The Write cycle transfers data from the CPU to the peripheral. It begins by selecting the peripheral and addressing the desired register. A setup time of 80 ns from register select stable to the falling edge of $\overline{WR}$ is required. The data must be valid prior to the falling edge of $\overline{WR}$. The $\overline{WR}$ pulse width is specified at 400 ns. Write cycle timing is shown in Figure 2.

Interrupt Acknowledge Cycle

The Z8500 peripheral interrupt structure offers the designer many options. In the simplest case, the Z8500 peripherals can be polled with interrupts disabled. If using interrupts, the timing shown in Figure 2 should be observed. (Detailed discussions of the interrupt processing can be found in the Zilog Data Book, document number 00-2034-02.) An interrupt sequence begins with an $\overline{INT}$ going active because of an interrupt condition. The CPU acknowledges the interrupt with an $\overline{INTACK}$ signal.

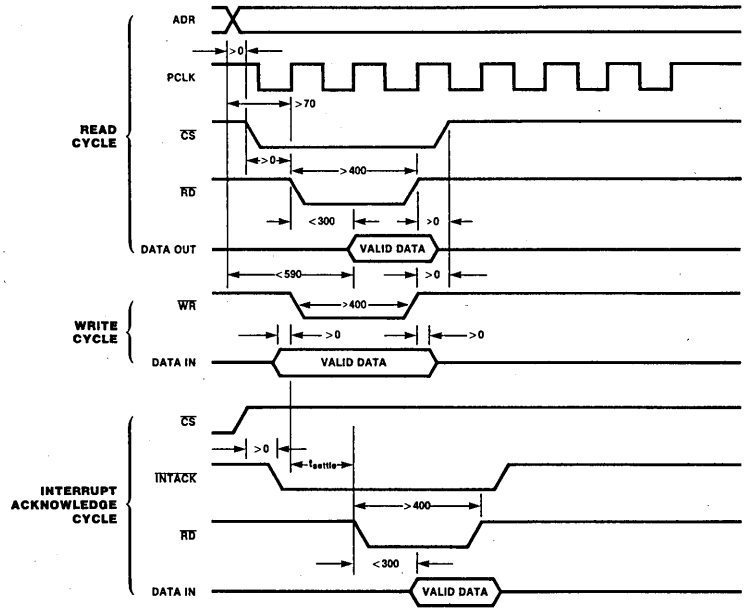


Figure 2. Z8500 Interface Timing (4 MHz)

A daisy-chain settle time (dependent upon the number of devices in the chain) ensures that the interrupts are prioritized. The falling edge of $\overline{RD}$ causes the IUS bit to be set and enables a vector to go out on the bus.

The table given in Figure 1 can be used to calculate the amount of settling time required by a daisy chain. Even if there is only one peripheral in the chain, a minimum settling time is still required because of the internal daisy chain. The first column specifies the amount of settling time for only one peripheral. If there are two peripherals, the time is computed by adding together the times shown in the first and the last columns. For each additional peripheral in the chain, the time specified in the middle column is added.

Recovery Time

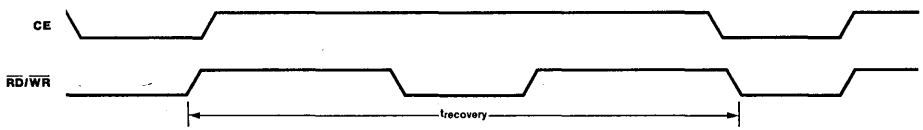
The read/write recovery time specifies a minimum amount of time between Read or Write cycles to the same peripheral. The recovery time differs among peripherals and is summarized in Figure 3. In most cases, this parameter is met because of the time required for instruction fetches. The recovery time specification does not have to be met if $\overline{CE}$ is deselected when Read or Write occurs.

68000 INTERFACE EXAMPLES

This section shows three examples, presented in increasing order of complexity, for interfacing Zilog's 4 MHz Z8500 peripherals to an 8 MHz 68000. Faster CPUs or peripherals can be used by modifying some of the timing. These examples suggest possible ways of implementing the interface but may require some modifications to operate properly. They were chosen because they give the user a variety of interface design ideas. The first example uses a minimum amount of TTL logic to implement the interface because the Valid Peripheral Address ($\overline{VPA}$) cycle meets the Z8500 timing requirements. In this mode the 68000 accepts only nonvectored interrupts. The second example uses the Data Transfer Acknowledge ($\overline{DTACK}$) pin. This interface allows faster operation and makes use of the Z8500's 8-bit vectored interrupts. The third example also uses a $\overline{DTACK}$ cycle and is similar to the second, except the external logic is integrated into a single chip, the PAL20X10 programmable array logic.

EXAMPLE 1: A TTL Interface Using a VPA Cycle

The 68000 has a special input pin, Valid Peripheral Address ($\overline{VPA}$), that can be activated by the Z8500 chip select logic at the beginning of the cycle to indicate to the 68000 that a peripheral is being accessed. This generates a special Read/Write cycle that meets the peripheral timing requirements. This cycle allows the Z8500 control signals to be generated easily. The 68000 responds to interrupts using an autovector and the Z8500 can be programmed not to return a vector.



Peripheral (4 MHz)	Recovery Time
CIO	Greater than 3 PCLK cycles or 1000ns
FIO	Greater than 1000ns
SCC	Greater than 4 PCLK cycles

NOTE: The diagram shows that the recovery time is measured between consecutive reads and writes only if the peripheral is selected.

Figure 3. Recovery Time

Figure 4 shows how the hardware can be implemented. PCLK is generated by dividing down the 68000 CLK. $\overline{RD}$, $\overline{WR}$, and $\overline{INTACK}$ are simply ANDed 68000 signals. The worst-case daisy-chain settle time is 450 ns. Connecting $\overline{INT}$ to IPL_0 generates a level 1 interrupt. The internal registers are accessed by A_0, A_1, $D/\overline{C}$, and $A/\overline{B}$, which can be the 68000 lowest order addresses. The timing is shown in Figure 5.

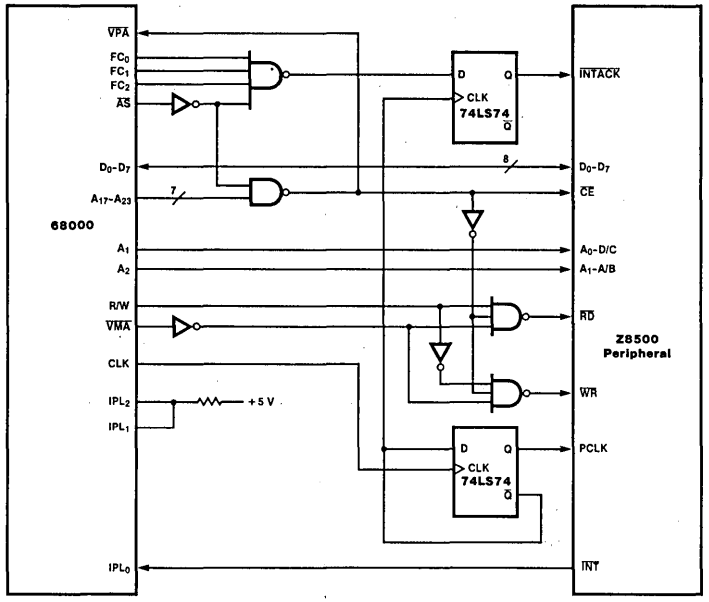


Figure 4. Interface Using the $\overline{VPA}$ Cycle

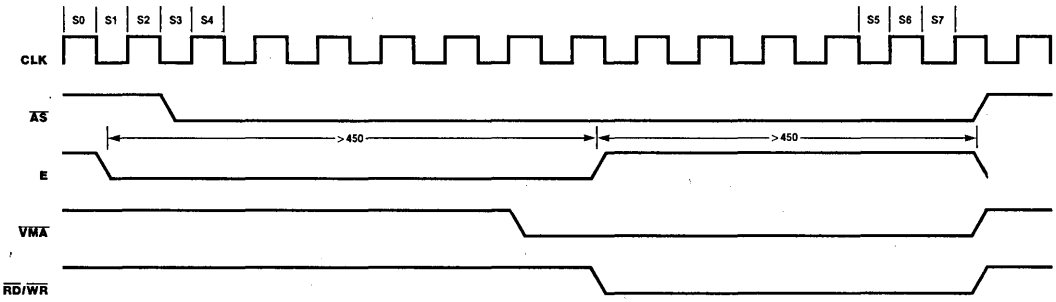


Figure 5. $\overline{VPA}$ Cycle Timing

Functional Description

$\overline{VPA}$ is pulled Low at the beginning of the cycle and the CPU automatically inserts Wait states until E is synchronized.

VPA = [(AS)·(CE)]

RD = [(CE)·(VMA)·(R/$\overline{W}$)]

WR = [(CE)·(VMA)·($\overline{R/W}$)]

INTACK = [(FC0)·(FC1)·(FC2)·(AS)]

EXAMPLE 2: A TTL Interface Using DTACK Cycles

Using the 68000 Data Transfer Acknowledge ($\overline{DTACK}$) cycle is a second way of interfacing to the Z8500 peripherals. The 68000 inserts Wait states until the $\overline{DTACK}$ input is strobed Low to complete the transfer. In addition to generating the control signals, the interface logic must also generate $\overline{DTACK}$.

The timing shown in Figure 6 can be generated by the hardware shown in Figure 7. The 8-bit Shift register (74LS164) is used to generate the proper timing. At the beginning of each cycle, Q_A (Figure 7) is set High for one PCLK cycle and then reset. This pulse is shifted through the Q_A-Q_H outputs and is used to generate $\overline{RD}$, $\overline{WR}$, and $\overline{DTACK}$ signals. Some of the extra Wait states can be eliminated by tapping the Shift register sooner (e.g., Q_C).

EXAMPLE 3: Single-Chip Pal Interface

This example illustrates how to interface the 4 MHz Z8500 peripherals to the 8 MHz 68000 using a PAL20X10 device to generate all the required control signals. The PAL reduces the required interface logic to a single chip, thus minimizing board space. This interface offers flexibility because the internal logic can be reprogrammed without changing the pin functions. The PAL uses 68000 signals to generate Read, Write, and Interrupt Acknowledge cycles. In addition to generating the Z8500 control signals, the PAL also generates a $\overline{DTACK}$ to inform the 68000 of a completed data transfer cycle. This allows the 68000 to use the peripheral's vectored interrupts.

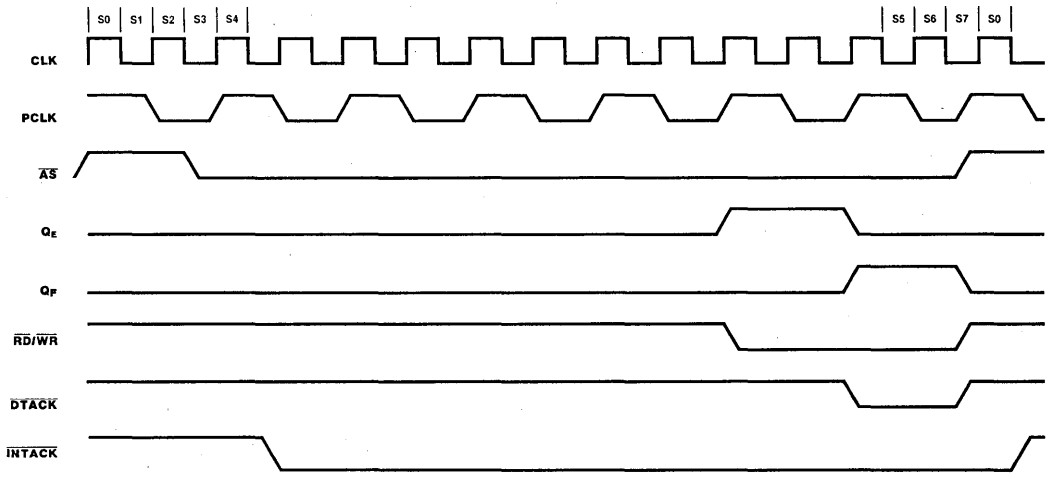


Figure 6. Timing for $\overline{DTACK}$ Interface

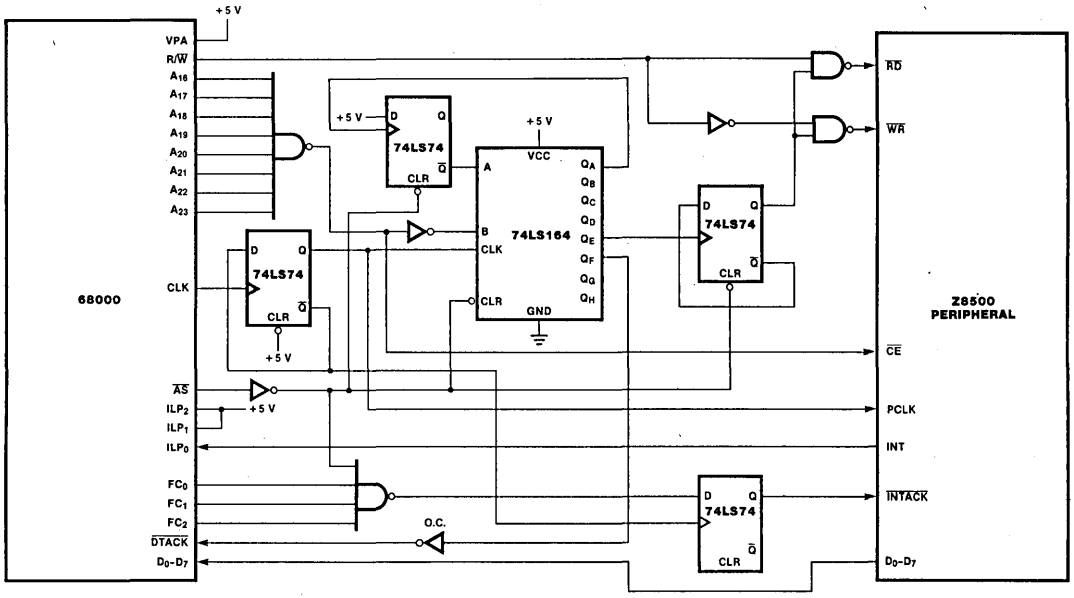


Figure 7. Hardware Diagram for $\overline{\text{DTACK}}$ Interface

Functional Description

Figure 8 shows the PAL's pin functions. The PAL generates five control signals, of which four ($\overline{WR}$, $\overline{RD}$, C_0, and $\overline{INTACK}$) go to the Z8500 and one ($\overline{DTACK}$) goes to the 68000. The remaining signals are used internally to generate these outputs. Timing diagrams for the Read, Write, and Interrupt Acknowledge cycles are shown in Figure 9.

The PAL uses a 4-bit downcounter to generate the proper placement of the control signals where C_0 is the least-significant bit and C_3 is the

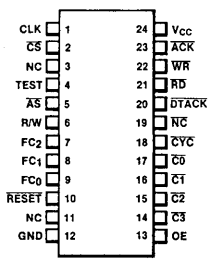


Figure 8. PAL Pinout

most-significant bit. All of the PAL is clocked with the rising edge of the 68000's CLK. The counter toggles between counts 14 and 15 and starts counting down when $\overline{AS}$ goes active. The counter goes back to toggling when $\overline{AS}$ goes inactive. CYC goes active Low at the same time the counter starts counting down. The equations in Figure 10 can be entered into a development board to program the PAL.

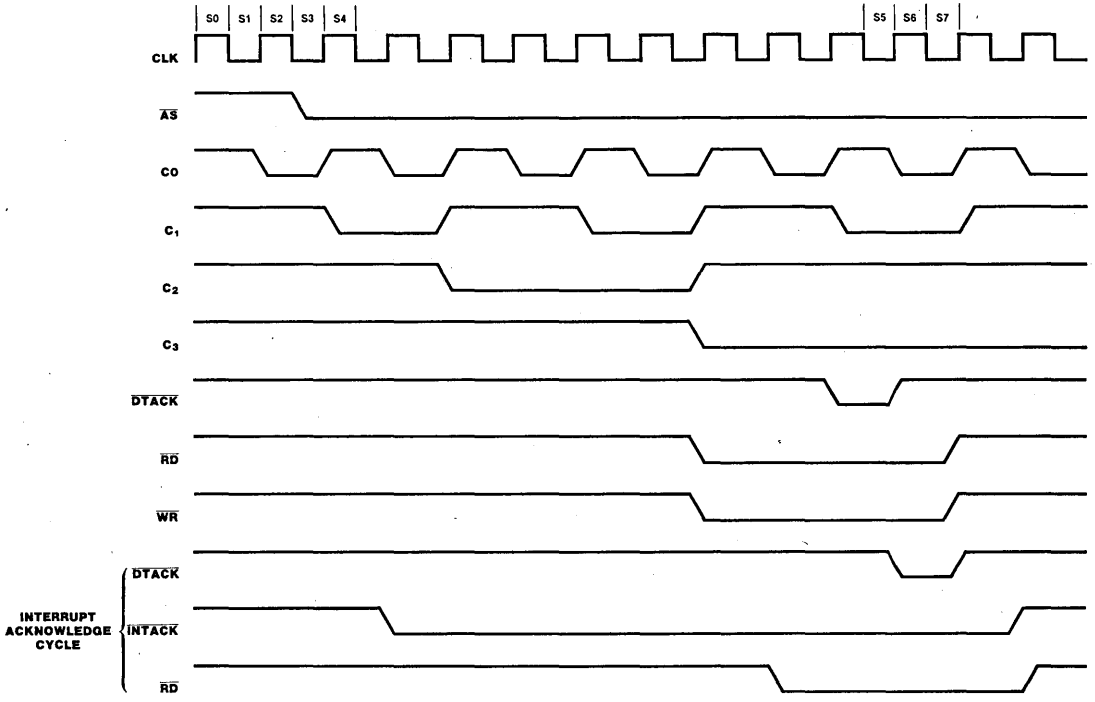


Figure 9. PAL Interface Timing

```
PAL20X10                                        PAL DESIGN SPECIFICATION
P7089 (10)
MC68000 TO ZILOG PERIPHERAL INTERFACE
MMI, SUNNYVALE, CA
CLK /CS  NC TEST /AS RW
FC2 FC1 FC0 /RESET NC GND
/OE /C3 /C2 /C1 /C0 /CYC
NC /DTK /RD /WR /ACK VCC


CO  :=    /CO*/TEST                                ; COUNT/HOLD (LSB)

C1  :=    /RESET*AS*C1                             ; HOLD
    :+:   /RESET*AS*CO                             ; DECREMENT

C2  :=    /RESET*AS*C2                             ; HOLD
    :+:   /RESET*AS*CO*C1                          ; DECREMENT

C3  :=    /RESET*AS*C3                             ; HOLD
    :+:   /RESET*AS*CO*C1*C2                       ; DECREMENT

DTK :=    /RESET*/ACK*CYC*C3*/C2*/C1* CO*CS        ; DTACK FOR RD/WR CYCLE
     +    /RESET* ACK*CYC*C3*/C2* C1*/CO           ; DTACK FOR INTERRUPT
                                                   ; OPERATION

CYC :=    /RESET*AS*/CYC*CO                        ; NEW CYCLE STARTED
     +    /RESET*AS* CYC                           ; PROCESSING OF CYCLE
    :+:   /RESET*CYC*DTK                           ; END OF CYCLE

RD  :=    /RESET*CYC*/ACK*RW* C3*/C2*CS            ; NORMAL READ OPERATION
     +    /RESET*CYC*/ACK*RW*/C3*C2*C1*CO*CS       ; NORMAL READ OPERATION
    :+:   /RESET*CYC* ACK*RW* C3                   ; READ DURING OPERATION
     +     RESET

WR  :=    /RESET*CYC*/ACK*/RW* C3*/C2*CS           ; WRITE
     +    /RESET*CYC*/ACK*/RW*/C3* C2*C1*CO*CS     ; WRITE
    :+:    RESET

ACK :=    /RESET*FCO*FC1*FC2*AS* CYC*/CO           ; INTERRUPT ACKNOWLEDGE
     +    /RESET*FCO*FC1*FC2*CYC                   ; INTERRUPT ACKNOWLEDGE
```

Figure 10. PAL Equations

Hardware Diagram

The hardware diagram of the PAL interface is shown in Figure 11. The 68000 signals CLK, $\overline{CS}$, $\overline{AS}$, R/$\overline{W}$, FC_0, FC_1, and FC_2 are used to generate the Z8500 control signals. The control signals are synchronous with the rising edge of the 68000's CLK. TEST and $\overline{OE}$ must be grounded. $\overline{CS}$ is used to enable $\overline{DTACK}$, $\overline{RD}$, and $\overline{WR}$ as shown in the equations. The Z8500 $\overline{INT}$ is connected to $\overline{ILP}_0$, which generates a 68000 level 1 interrupt. The peripherals are memory-mapped into the highest 64K byte block of memory, where A_{17}-A_{23} equals "FF_H". Addresses A_4-A_6 are used to select the peripheral; A_1-A_3 select the internal registers. Table 2 shows the peripheral's memory map.

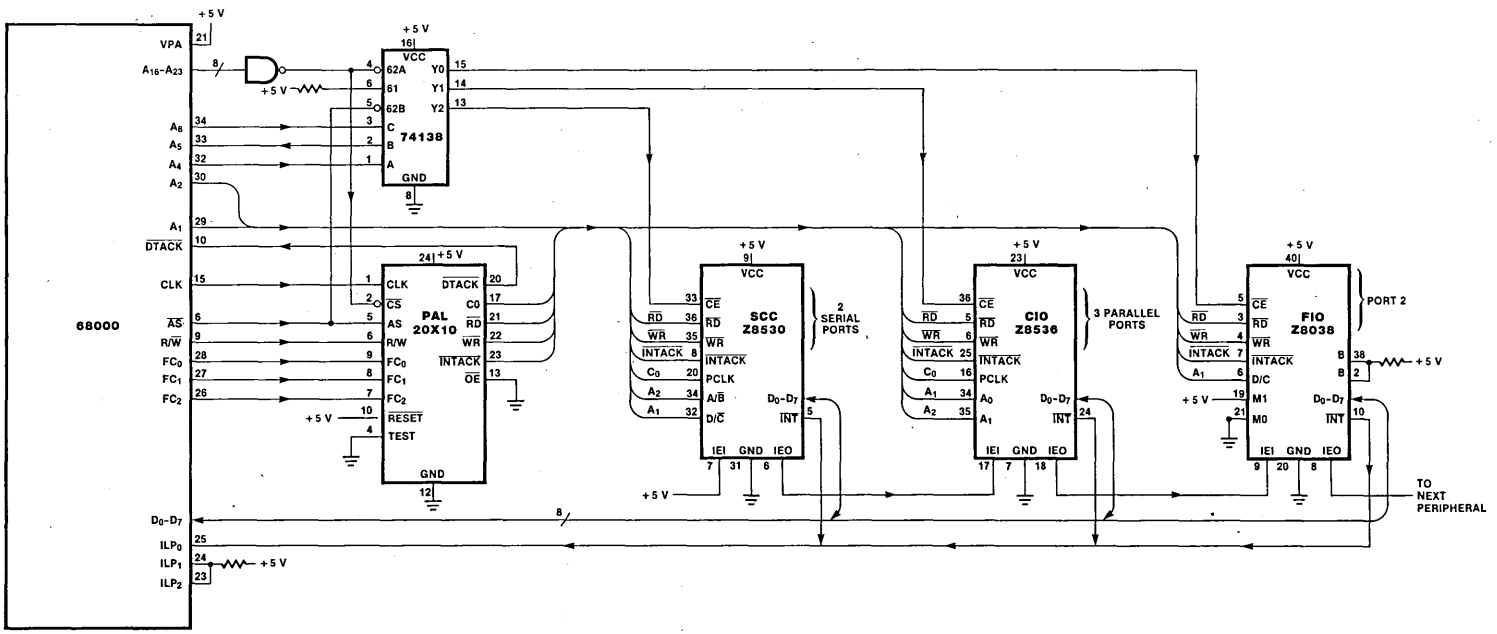


Figure 11. PAL Hardware Diagram

Table 2. Peripheral Memory Map

Peripheral	Register	Hex Address
SCC (Z8530)		
	Channel B Control	FF0020
	Channel B Data	FF0022
	Channel A Control	FF0024
	Channel B Data	FF0026
CIO (Z8536)		
	Port C's Data Register	FF0010
	Port B's Data Register	FF0012
	Port A's Data Register	FF0014
	Control Register	FF0016
FIO (Z8038)		
	Data Registers	FF0000
	Control Registers	FF0002

INTERFACE VERIFICATION TECHNIQUES

This section suggests possible ways of verifying the Read, Write, and Interrupt Acknowledge cycles.

Read Cycle Verification

The Read cycle should be checked first because it is the simplest operation. The Z8500 should be hardware reset by simultaneously pulling $\overline{RD}$ and $\overline{WR}$ Low. When the peripheral is in the reset state, the Control register containing the reset bit can be read without writing the pointer. Reading back the FIO or CIO Control register should yield a 01_H.

The SCC's Read cycle can be verified by reading the bits in RR0. Bits D_2 and D_6 are set to 1 and bits D_0, D_1, and D_7 are 0. Bits D_3-D_5 reflect the input pins DCD, SYNC, and CTS, respectively.

Write Cycle Verification

The Write cycle can be checked by writing to a register and reading back the results. Both the CIO and FIO must have their reset bits cleared by writing 00_H to their Control registers and reading back the result. The SCC can be checked by writing and reading to an arbitrary read/write register, for example, the Time Constant register (WR12 or WR13).

Interrupt Acknowledge Cycle Verification

Verifying an Interrupt Acknowledge ($\overline{INTACK}$) cycle consists of several steps. First, the peripheral makes an Interrupt Request ($\overline{INT}$) to the CPU. When the processor is ready to service the interrupt, it initiates an Interrupt Acknowledge ($\overline{INTACK}$) cycle. The peripheral then puts an 8-bit vector on the bus, and the 68000 uses that vector to get to the correct service routine. This test checks the simplest case.

First, load the Interrupt Vector register with a vector, disable the Vector Includes Status (VIS), and enable interrupts (IE = 1, MIE = 1, IEI = 1). Disabling VIS guarantees that only one vector is put on the bus. The address of the service routine corresponding to the 8-bit vector number must be loaded into the 68000's vector table.

Initiating an interrupt sequence in the FIO and CIO can be accomplished by setting one of the interrupt pending (IP) bits and seeing if the 68000 jumps to the service routine (setting a breakpoint at the beginning of the service routine is an easy way to check if this has happened).

Initiating an interrupt sequence in the SCC is not quite as simple because the IP bits are not as accessible to the user. An interrupt can be generated indirectly via the CTS pin by enabling the following: CTS IE (WR15 20), EXT INT EN (WR1 01), and MIE (WR9 08). Any transition on the CTS pin can initiate the interrupt sequence. The interrupt can be re-enabled by RESET EXT/STATUS INT (WR0 10) and RESET HIGHEST IUS (WR0 38).

CONCLUSION

Zilog's Z8500 family of nonmultiplexed Address/Data bus peripherals can interface easily with the 68000 and provide all the support required in a high-performance microprocessor system. The many features offered by the SCC, FIO, and CIO solve many system design problems by making interfacing to the external world easy. These intelligent peripherals also greatly enhance the system performance by relieving the CPU of many burdensome overhead tasks. Additionally, the powerful interrupt structure allows the 68000 to use vectors and reduce interrupt response time.

Design Considerations Using Quartz Crystals with Zilog's Components

October 1988

INTRODUCTION

Many times, the designer of microprocessor-based systems has considerable expertise in digital logic design but knows little about analog circuits. This designer needs to know about crystal operation with microprocessors because a crystal in conjunction with an oscillator (on-chip/off-chip) provides the clock to the microprocessor. This application note is intended to answer simple questions about the operation of crystals and their use with Zilog's components.

BRIEF THEORY ON CRYSTALS

A quartz crystal is a piezoelectric device that transforms electrical energy into mechanical energy. When an electrical potential is applied to the quartz crystal, it starts oscillating at a frequency dependent on the characteristics of the crystal.

The schematic representation of a simple quartz crystal is shown in Figure 1. The equivalent circuit of the crystal is shown in Figure 2.

In Figure 2, L represents the motional inductance as a result of the motion of the mechanical mass; this is similar to mechanical resonators. C represents motional capacitance while R represents motional resistance. Cs, the shunt capacitance, is the electrostatic capacitance due to the crystal electrodes, with the quartz plates as a dielectric. The value of the capacitance Cs depends on the area and thickness of the quartz plates. Cs also depends on contact wires, the crystal holder, and the angle of cut from the mother crystal. These cuts are chosen to optimize the temperature coefficient, the frequency range, and other crystal parameters.

The performance of the crystal is dependent on mechanical and electrical characteristics. The mechanical characteristics include the method of lead attachment, package sealing method, and internal environment (e.g., vacuum, partial pressure, etc.). The electrical characteristics include load capacitance, crystal resistance, drive level, temperature coefficient/turning point, and frequency tolerance.

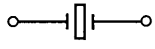


Figure 1. Schematic Representation of Quartz Crystal

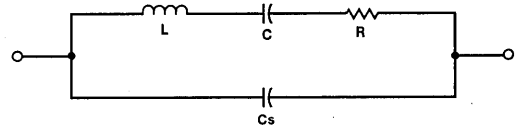


Figure 2. Equivalent Circuit of Quartz Crystal

SERIES RESONANCE AND PARALLEL RESONANCE

Manufacturers of crystals specify crystals as either series-resonant or parallel-resonant (Figure 3).

At the frequency of series resonance:

$$fs = \frac{1}{2\pi\sqrt{L*C}}$$

When the magnitude of X_C (reactance of C) and X_L (reactance of L) are equal, one effectively cancels the other, resulting in an equivalent of R shunted by Cs. If R is very small compared to X_{CS}, series resonance is indicated by minimum impedance and zero phase shift. The series-resonant crystals are resistive and produce an output in phase with the input.

At frequencies slightly higher than series resonance, X_L increases and X_C decreases, resulting in a net inductive reactance, X_L. When $X_L = X_{CS}$, the result is parallel resonance with the frequency of parallel resonance:

$$fp = \frac{1}{2\pi\sqrt{L*\frac{C*C_S}{C+C_S}}}$$

Parallel resonance is indicated by maximum impedance across the crystal terminals. Parallel-resonant crystals are inductive and produce the output shifted in phase from the input.

If a series crystal is used with a parallel-resonant oscillator, the crystal is forced to operate in the parallel region (and vice versa). The clock frequency produced is shifted a small percentage (about 350 ppm) from the specified series crystal frequency (Figure 3). Any attempts to fine tune the frequency by changing the value of external capacitance may cause the crystal to stop oscillating.

From the equivalent circuit of the crystal and the expression of fs, it can be seen that the series-resonant frequency of the crystal cannot be changed by reactance across the crystal terminals because there is no connection to the junction of L and C. In the case of parallel resonant frequency, Cs appears in the expression and its effective value can be changed by reactance across the terminals.

The load capacitance (C_L) is the capacitance that the crystal sees at its terminals. In parallel-resonant mode, load capacitance is very important, because C_L in combination with crystal inductance determines the frequency.

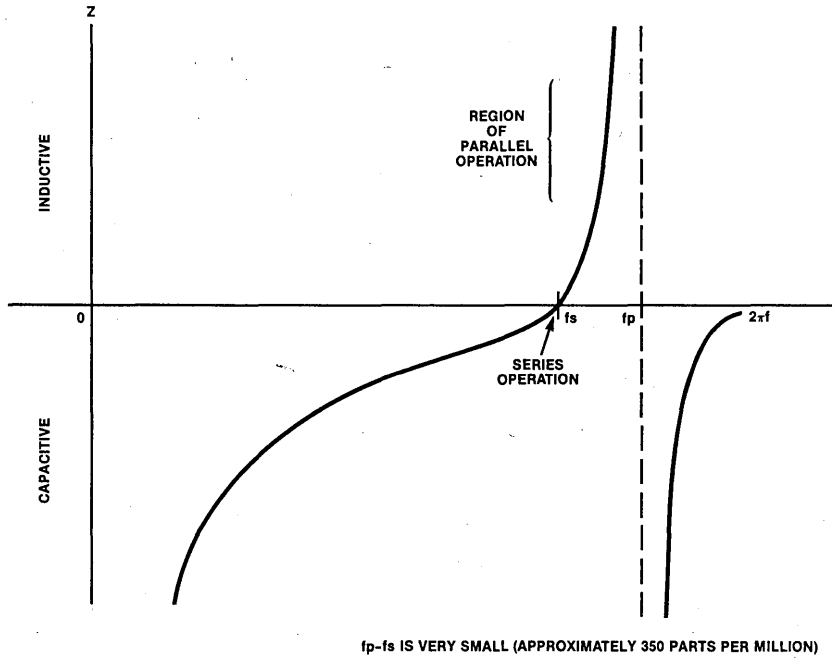


Figure 3. Series vs. Parallel Resonance

Crystal selection is based on the oscillator design used to provide the clock to the system. Series-resonant crystals should be used with non-inverting oscillators because series-resonant crystals have no phase shift. Parallel-resonant crystals should be used with inverting oscillators because parallel-resonant crystals have some phase shift due to their inductive nature.

DRIVE LEVEL

Drive level is critical because the crystal can dissipate only limited power (10 mW typical) and still meet all specifications. Overdrive may cause the temperature to rise in the crystal. Further overdrive may cause the quartz to operate in a non-linear region producing a frequency shift and possibly causing permanent damage to the crystal.

SERIES CONFIGURATION

When the internal oscillator is non-inverting, the recommended crystal is series-resonant and the circuit diagram shown in Figure 4 is used.

Many supposedly series-resonant IC oscillators actually operate below series resonance. C1 is used to compensate for this inductive nature of the series-type oscillator. C1 provides enough capacitive reactance to cancel the inductive shift caused by the IC. This brings the overall frequency back to series resonance.

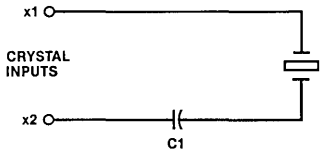


Figure 4. Circuit for Series-Resonant Crystals

PARALLEL CONFIGURATION

When the internal oscillator is inverting, the recommended crystal is parallel-resonant and the circuit diagram shown in Figure 5 is used.

The load capacitance, C_L, is determined by the following equation:

$$C_L = \frac{C_1 * C_2}{C_1 + C_2} + C_C$$

where C_C is the parasitic circuit capacitance.

The parallel resonance frequency is determined by the following equation:

$$fp = \frac{1}{2\pi \sqrt{L^* \frac{C * C_t}{C + C_t}}}$$

where $C_t = C_L + C_S$

Typically, the value of C_L ranges between 20pf and 30pf, and the value of the ratio C1/C2 ranges between 1 and 2. The required frequency can be fine tuned by varying the value of C2. The value of C_L, derived from C1 and C2, depends on the required frequency, characteristics of the oscillator, and the crystal used.

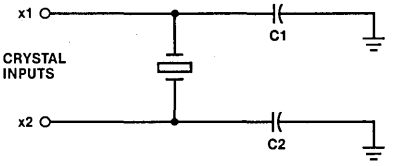


Figure 5. Circuit for Parallel Resonant Crystals

RECOMMENDED CIRCUIT FOR Z8030/8530 AND Z80C30/85C30

This device has two channels. Since this device has an internal inverting oscillator, a parallel resonant crystal is recommended for each channel. The crystal is connected between the RTxC & SYNC of the respective channels. The circuit configuration of both the channels is shown in Figure 6. The preferred value of the load capacitance of the crystal (CL) for both the channels is 22pf. For channel A the preferred value of C1A and C2A is 33pf, which produces a ratio (C1A/C2A) of one. C2A can be made variable to fine tune the required frequency. Similarly for channel B, the preferred value of C1B and C2B is 33pf, which produces a ratio (C1B/C2B) of one. C2B can be made variable to fine tune the required frequency.

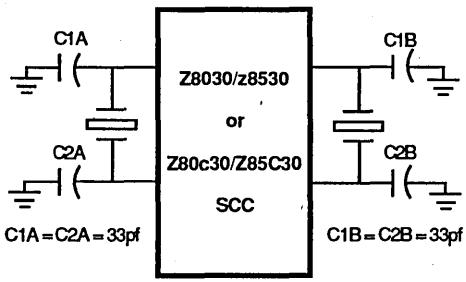


Figure 6. Circuit Configuration for Z8030/Z8530 or Z80C30/Z85C30

Since the internal oscillator of the Zilog Z8581 is inverting, a parallel-resonant type crystal is recommended. The circuit configuration is shown in Figure 7. The preferred value of the Load Capacitance (CL) of the crystal is about 22pf. The preferred value of C1 and C2 is 33pf, which produces a ratio (C1/C2) of one. C2 can be made variable to fine tune the required frequency.

The output of one oscillator can also be connected to the crystal inputs of the second oscillator, using a 4700 ohms pull-up resistor at the output (Figure 7).

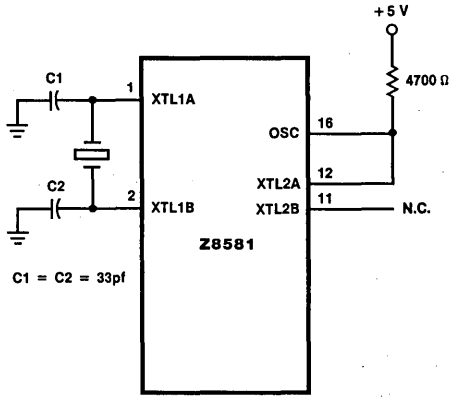


Figure 7. Circuit Configuration for Z8581

SPECIFICATIONS

For proper operation when using Zilog's Z8581, use a fundamental, parallel-type crystal. The following crystal specifications are suggested:

Frequency tolerance: Application dependent
C_L, load capacitance: Approximately 22pf (acceptable range is from 20-30pf)
Rs, equivalent-series resistance: © 150 ohms
Drive level
(for © 10 MHz crystal): 10 milliwatts
(for □ 10 MHz crystal): 5 milliwatts

Holder specifications are user-determined.

For frequencies lower than 4 MHz, the recommended holder type is HC-33/U (0.75″W × 0.765″H, 1.5″ lead length with specing of 0.486″).

For frequencies higher than 4 MHz, the recommended holder type is HC-18/U (0.435″W × 0.530″H, 1.5″ lead length with specing of 0.192″).

SUGGESTED VENDORS

The wide variety of applications and frequencies in which crystals are used makes it necessary to custom manufacture all but a very few crystal types. With the crystal specifications given above, most vendors can make the desired crystal even though it may not be a standard off-the-shelf item.

The following two vendors supply crystals to the specifications given above. The user is not limited to these vendors because these suppliers are among many suppliers who might meet your specifications.

Midland-Ross Corporation
NEL UNIT
357 Beloit Street
Burlington, WI 53105
Telephone: (414) 763-3591

CTS KNIGHTS, INC.
400 E. Reimann Ave.
Sandwich, Illinois 60548
Telephone: (815) 786-8411

Using Z8581 Clock Stretches in Z80® CPU Applications

October 1988

INTRODUCTION

Today's high-speed microprocessors are quickly becoming faster than the peripherals and memory devices they must interface. For example, the Zilog Z80H runs at 8 MHz, while its supporting peripherals run at 4 MHz.

In many applications, this is not a problem because the applications are CPU-bound rather than I/O-bound. And, when necessary, wait states can be used to idle the CPU while its supporting devices catch up.

Now, Zilog has come up with a more elegant and flexible means of slowing the Z80 CPU. The Zilog Z8581 Clock Generator and Controller (CGC) chip, when used to generate system clock pulses, can stretch them selectively. That is, the duration of individual clock pulses can be increased, adding time to the cycle without using wait states.

The Z80 $\overline{WAIT}$ line has two major limitations: a) the delay must be for an entire clock cycle, and b) it must occur between clock cycles T2 and T3 in the machine cycle. On the other hand, clock stretches can be implemented for half, whole, or one and one-half clock cycles, and they can occur almost anywhere in the machine cycle.

This application note shows several circuit applications that use the Z8581 clock-stretch capability to minimize the use of wait states, resulting in faster performance, and sometimes saving PC board real estate.

The designs shown here use standard TTL parts to decode local signals and generate stretch requests when required.

CLOCK STRETCH PIN FUNCTIONS

The following Z8581 pins are directly related to the clock stretch operation:

$\overline{ADD1}$, $\overline{ADD2}$. *Add Delay* (input, active Low). These pins request a stretch and control its length. $\overline{ADD1}$ causes a one-half clock cycle stretch, $\overline{ADD2}$ causes a full clock cycle stretch, and both together cause a one and one-half clock cycle stretch.

C0, C1. *ZCLK Count* (output, active High). These pins keep a binary count of clock cycles, starting when $\overline{STRT}$ is activated. They increment on the rising edges of ZCLK, the oscillator output used as a system clock.

$\overline{INH}$. *Inhibit Delay* (input, active Low). When held Low, this pin masks $\overline{ADD1}$ and $\overline{ADD2}$, preventing clock stretching.

$\overline{STRH}$. *Stretch Clock* (input, active Low). This pin causes an immediate clock stretch which lasts as long as it remains Low.

$\overline{STRT}$. *Start Count* (input, active Low). This pin resets the binary counter, C0 and C1.

CIRCUIT APPLICATION EXAMPLES

The following examples show circuit applications.

Opcode Fetch

Figure 1 shows a circuit that uses $\overline{M1}$, $\overline{MREQ}$, and some TTL logic to generate a clock stretch in T2 during an opcode fetch. Figure 1 shows the logic and Figure 2 shows the timing.

This circuit makes use of the fact that C0 and C1 identify the current clock cycle. $\overline{M1}$ starts the circuit, but only the right combination of C0, C1, and $\overline{MREQ}$ can actually generate the clock stretch.

Sequence

$\overline{M1}$ activates $\overline{STRT}$, which restarts the internal count on C0 and C1. Both immediately go Low.

C1 and ZCLK go to an inverting OR gate that controls $\overline{INH}$. C1 remains Low for now, but the rising edge of ZCLK causes $\overline{INH}$ to go High. During T1 this does not effect the Z8581 because $\overline{ADD1}$ and $\overline{ADD2}$ are inactive (no stretch request is present).

$\overline{ADD1}$ is controlled by an inverting AND gate driven by C0 and $\overline{MREQ}$. When C0 drops Low, the gate is blocked and $\overline{ADD1}$ cannot be generated. This keeps ZCLK normal for T1.

In the middle of T1, $\overline{MREQ}$ goes Low and C0 goes High. This causes the AND gate to assert $\overline{ADD1}$. $\overline{INH}$ is forced High by ZCLK at the same time, so a single clock stretch is inserted in T2.

By T3, $\overline{MREQ}$ and $\overline{ADD1}$ are deactivated, so no clock stretch is inserted. As long as $\overline{MREQ}$ is High, the AND gate prevents $\overline{ADD1}$ from being activated. As soon as $\overline{MREQ}$ goes Low, the C0 output controls $\overline{ADD1}$.

Variations

This circuit can be modified slightly to increase the length of the clock stretch, and to make it work for memory accesses or for I/O accesses. Figure 3 shows a circuit that adds as much time as a single wait state.

Length of Stretch. To control the length of the clock stretch, connect the output of the AND gate as follows:

Connections	Length of Stretch
$\overline{ADD1}$	½ clock cycles (125ns @ 8MHz)
$\overline{ADD2}$	1 clock cycle (250ns @ 8MHz)
$\overline{ADD1}$ and $\overline{ADD2}$	1½ clock cycles (375ns @ 8MHz)

Memory Fetch. Because $\overline{M1}$ is not present during memory fetches, $\overline{MREQ}$ can be used to drive both $\overline{STRT}$ and $\overline{ADD1}$ and/or $\overline{ADD2}$. Referring to the circuit shown in Figure 1, disconnect $\overline{M1}$, and replace it with $\overline{MREQ}$. (Be sure to use $\overline{MREQ}$ before it goes through the inverting gate.) $\overline{MREQ}$ will activate $\overline{STRT}$, and still allow C0 to control the AND gate when it goes High.

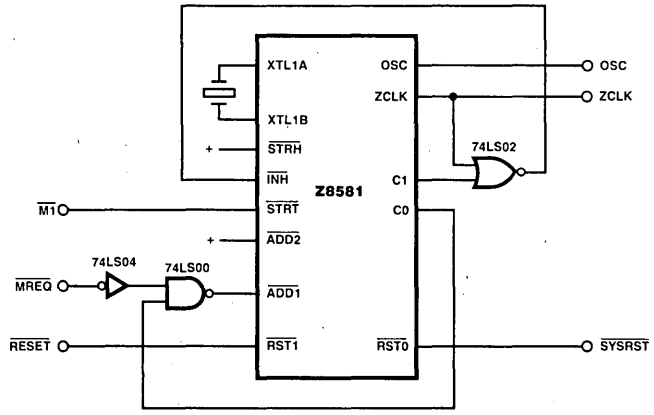


Figure 1. Opcode Fetch

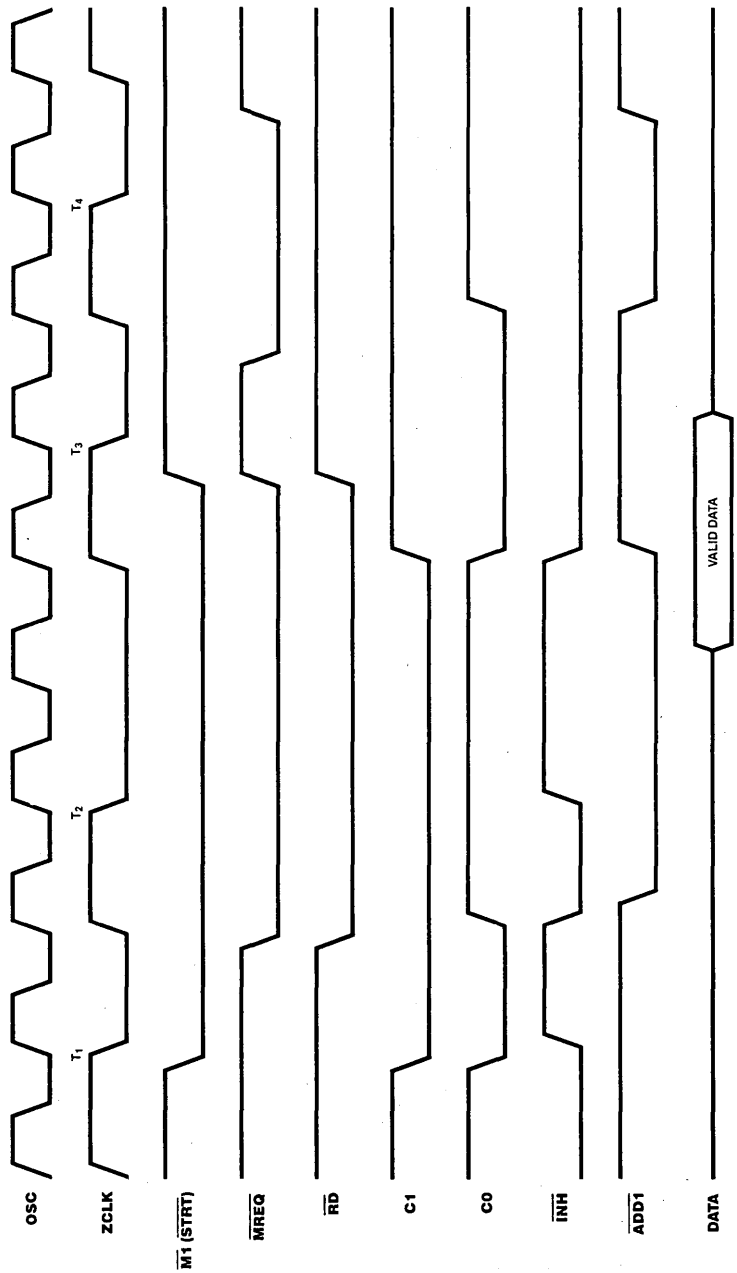


Figure 2. Opcode Fetch Timing

I/O Accesses. The circuit shown in Figure 3 can be changed to work for I/O accesses by using $\overline{IORQ}$ instead of $\overline{MREQ}$. Again, $\overline{IORQ}$ must be used before it is inverted, and the length of the stretch can be adjusted via the connections to $\overline{ADD1}$ and/or $\overline{ADD2}$.

Interrupt Acknowledge Cycle

To Z80 peripherals, the presence of $\overline{M1}$ without $\overline{RD}$ signals an interrupt acknowledge cycle. They freeze their internal daisy chain to prevent conflict, determine which interrupting device has the highest priority, then wait for $\overline{IORQ}$. Upon receipt of $\overline{IORQ}$, the highest priority device with an interrupt pending places its vector on the data bus.

The time between $\overline{M1}$ and $\overline{IORQ}$ is sometimes called the daisy-chain settle time. The peripherals need this time to determine which gets its interrupt serviced.

If the daisy-chain settle time is not long enough, $\overline{IORQ}$ has to be delayed. But the Z80 CPU's behavior is automatic in this respect; after T2, it activates $\overline{IORQ}$ and inserts two wait states.

Traditional designs insert a delay in $\overline{IORQ}$ between the time it leaves the CPU and arrives at the peripherals. This requires additional circuits, including a shift register. A typical circuit, and its timing, appear in Figures 4 and 5.

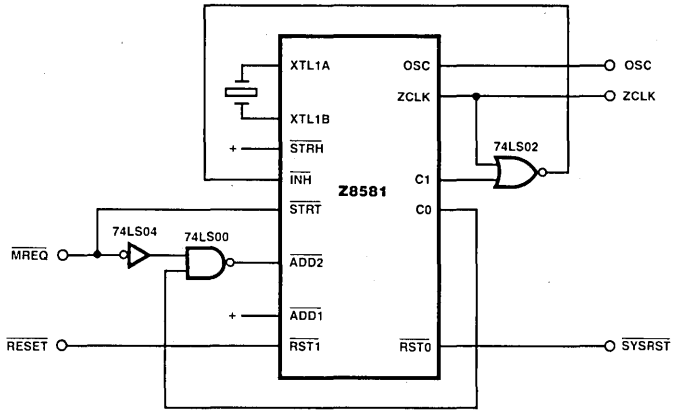


Figure 3. Memory Fetch with Full Clock Cycle Stretch

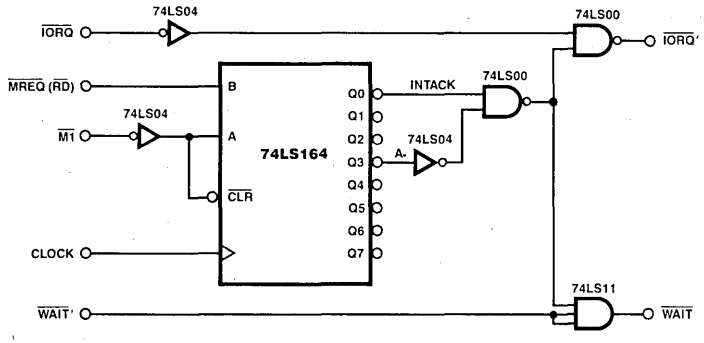


Figure 4. Traditional IORQ Delay

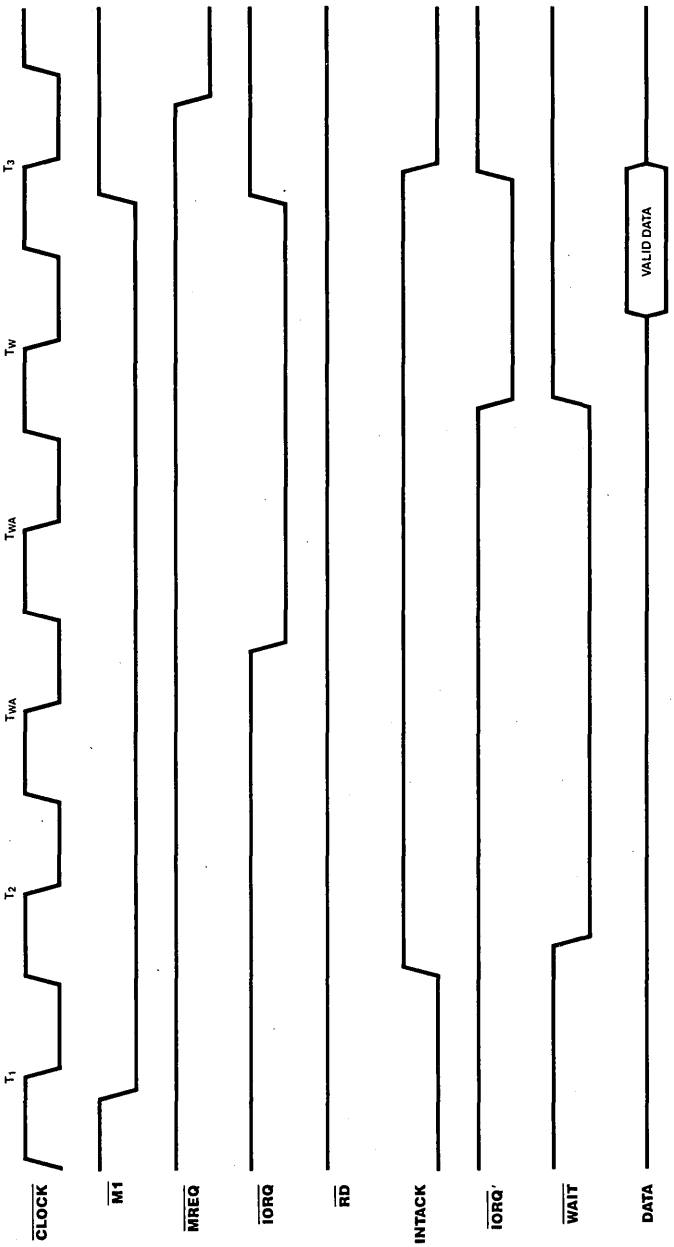


Figure 5. Traditional IORQ Timing

Figure 6 shows a simpler solution, using the Z8581. Timing for this circuit appears in Figure 7.

Sequence

$\overline{M1}$ activates $\overline{STRT}$, which resets the counters C0 and C1.

During this entire cycle, $\overline{MREQ}$ stays High, allowing C0 to control the AND gate. This way, $\overline{MREQ}$ can block the clock stretch during memory accesses.

C0 goes High at the beginning of T1, causing the AND gate to generate $\overline{ADD1}$. However, for the second half of T1, ZCLK is High, causing $\overline{INH}$ to block the request.

At the beginning of T2, while C1 is still Low, ZCLK goes Low, driving $\overline{INH}$ High.

Meanwhile, C0 is still Low, so $\overline{ADD1}$ is still present. The combination of INH being High and $\overline{ADD1}$ being Low causes a clock stretch in the low half of T2.

Because $\overline{IORQ}$ must wait for T2 to be completed, it is delayed for the length of the stretch.

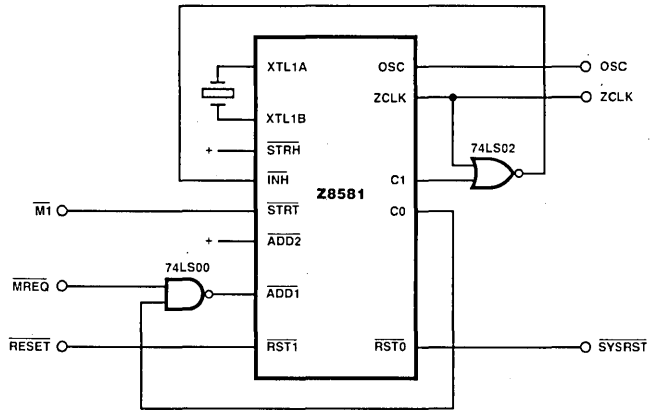


Figure 6. Interrupt Acknowledge

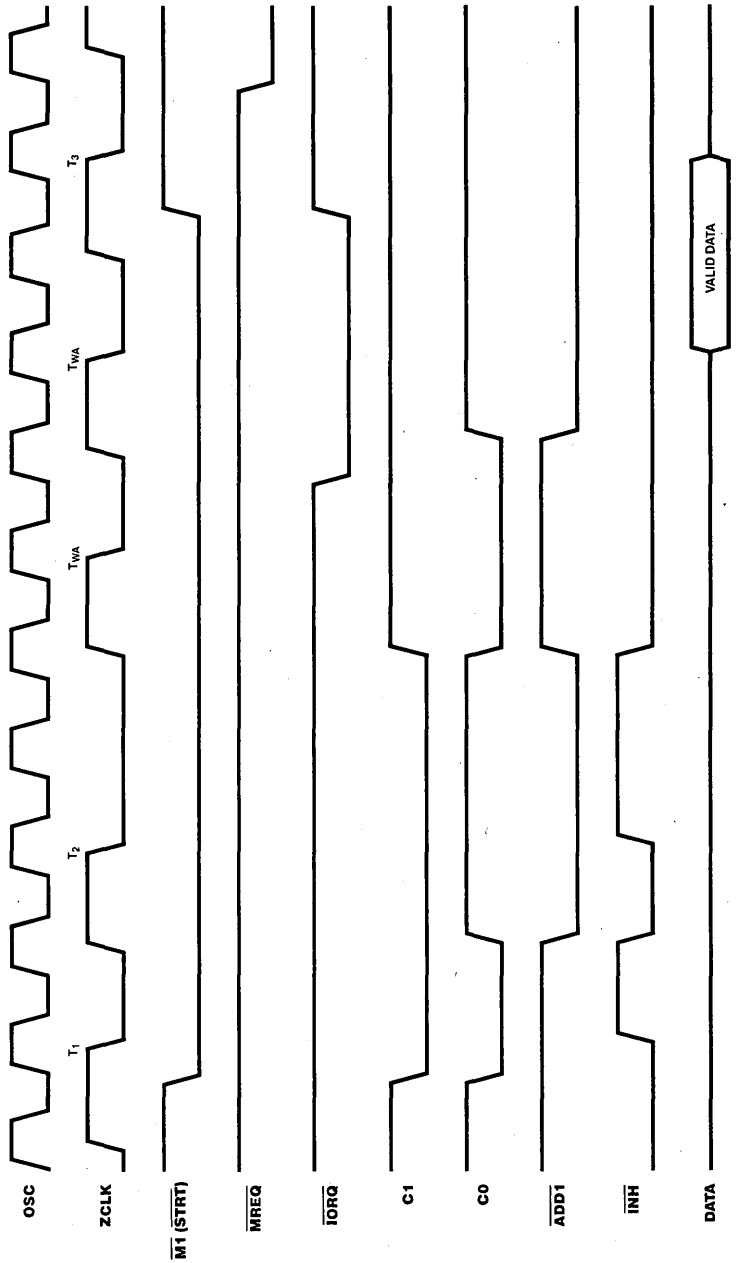


Figure 7. Interrupt Acknowledge Timing

RAS Precharge Timing

In dynamic memory systems, $\overline{RAS}$ must be inactive for a certain amount of time before it is reactivated. This time is known as the $\overline{RAS}$ precharge time.

The circuit shown in Figure 8 uses clock stretches for two purposes: to delay the timing for opcode fetches, and to increase the length of the $\overline{MREQ}$ signal from which $\overline{RAS}$ is (usually) derived. As shown in Figure 9, this circuit stretches the High portion of T2 for opcode fetches, and does the same for T3 to keep $\overline{MREQ}$ High long enough to satisfy the $\overline{RAS}$ precharge time.

Sequence

The exclusive OR gate across C0 and C1 goes High whenever these two match—that is, during the first half of T1 and the last half of T3.

This signal is ANDed with ZCLK; when both are High, $\overline{INH}$ is asserted and no stretch can be inserted.

$\overline{ADD1}$ is active only when $\overline{MREQ}$ is present and when C0 and C1 are different. That is, from the last half of T1 to the last half of T3.

Because a stretch request ($\overline{ADD1}$) is always present during this time, $\overline{INH}$ enables stretches when it goes High. This occurs during the rising edges of T2 and T3, causing stretches during the first half of T2 and T3.

CONCLUSIONS

These examples show that the Z8581 is a versatile device, capable of doing everything that a wait-state generator can, and more. Not only can it save PC board real estate, but it allows the use of slower (less expensive!) memories and I/O devices.

There are many other possibilities. All the designs shown could be implemented in the same system by exchanging the discreet TTL components for Programmable Array Logic (PAL) chips. And these designs didn't even use the $\overline{STRH}$ line, which can be used to stretch the clock more flexibly than $\overline{ADD1}$ and $\overline{ADD2}$.

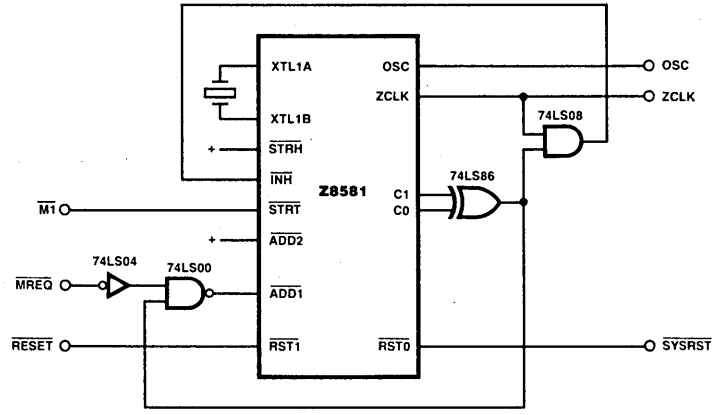


Figure 8. RAS Precharge Circuit

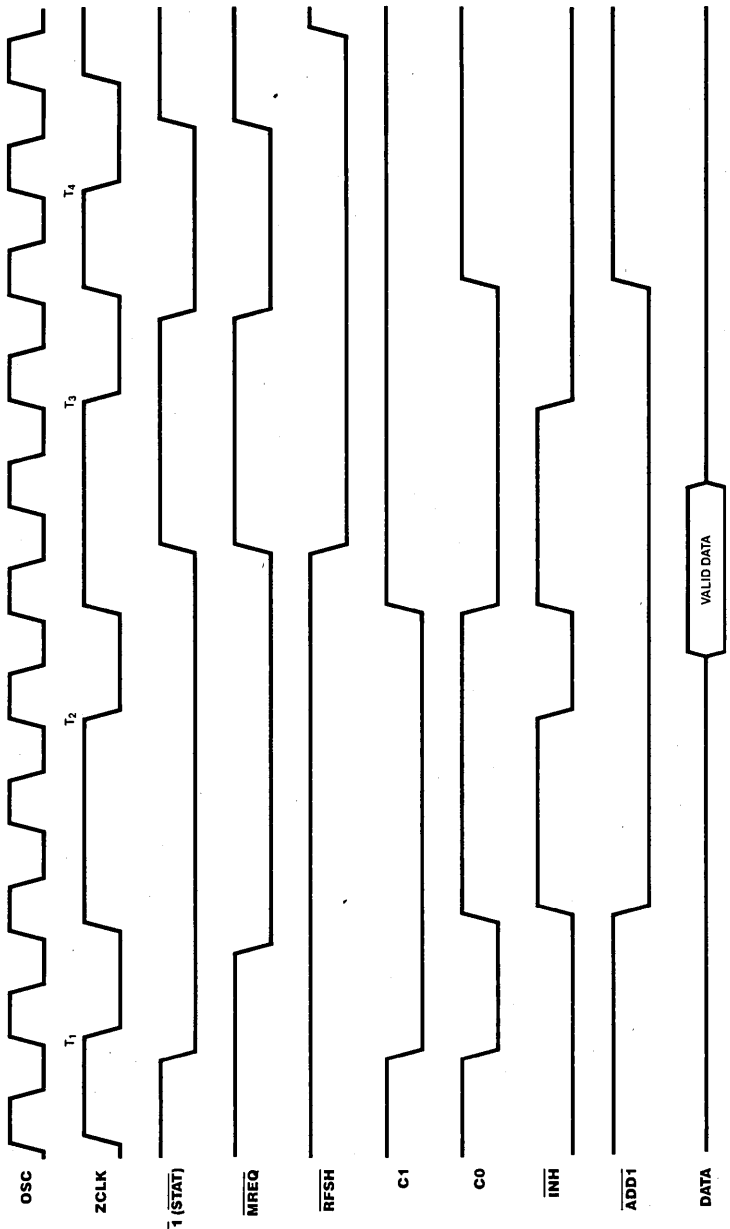


Figure 9. RAS Precharge Timing

Interfacing Z-Bus® Peripherals to the V20/V30/8086/8088

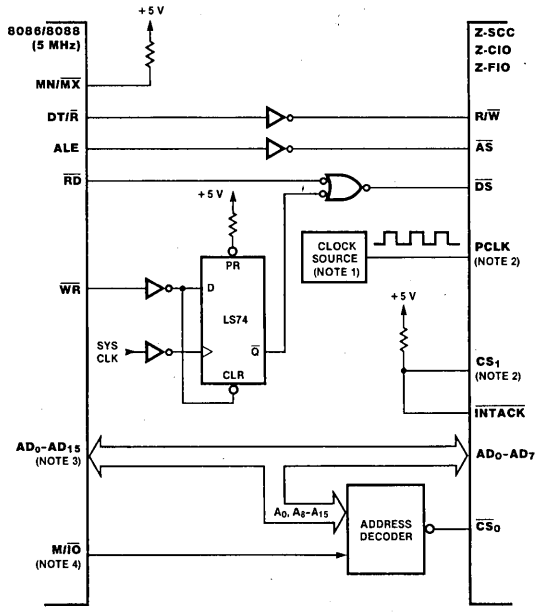
October 1988

INTRODUCTION

Microcomputer systems based on Intel's 8086 and 8088 CPUs can take advantage of the advanced features of Zilog's Z8000®series of microprocessor peripherals with a minimal amount of external logic. These devices are easily integrated and can satisfy many of the peripheral support requirements in a typical 8086/8088-based system. This Application Note discusses a general design that enables the 8086/8088 to interface with Zilog's Serial Communications Controller (Z8030 Z-SCC), Counter/Timer - Parallel I/O Unit (Z8036 Z-CIO), and FIFO I/O Controller (Z8038 Z-FIO). Discussions of the Z8500 peripherals (non-multiplexed address and data bus versions) can be found in other Zilog documents.

BUS INTERFACE

The Z8000 peripherals (also called Z-BUS peripherals) lend themselves conveniently to 8086/8088 - based designs because of the multiplexed address/data bus architecture. There is no need for an external address latch because the Z8000 peripherals latch addresses internally at the beginning of each bus cycle. Furthermore, the peripherals allow the CPU direct access to all of their data and control registers. Figure 1 shows the interface logic that translates the signals generated by the 8086/8088 into the necessary Z-BUS signals, and Table 1 gives a description of each signal.



Note:
1. The source of PCLK can, but need not, be derived from the System CLK.
2. Does not apply to Z FIO
3. AD_0-AD_7 and A_8-A_{15} on 8088.
4. IO/M on 8088.

Table 1. Signal Descriptions

8086/8088 Signals

MN/$\overline{MX}$	Minimum/$\overline{Maximum}$. This input is pulled high so that the CPU will operate in the "Minimum Mode."
DT/$\overline{R}$	Data Transmit/$\overline{Receive}$. DT/$\overline{R}$ is high on write operations and low on read operations.
ALE	Address Latch Enable. ALE is used to latch addresses during the first T state of each bus cycle so that the bus can then be free to transfer data.
$\overline{RD}$	Read. $\overline{RD}$ strobes data into the CPU on read operations.
$\overline{WR}$	Write. $\overline{WR}$ strobes data out of the CPU on write operations.
AD_0-AD_{15}	This is the 16-bit, multiplexed address/data bus on the 8086. The 8088 has a low order address/data bus, AD_0-AD_7, and a high order address bus, A_8-A_{15}.
M/$\overline{IO}$	Memory/$\overline{Input\text{-}Output}$. This output distinguishes between memory and I/O accesses. On the 8086 it is high on memory accesses and low on I/O accesses. On the 8088, the polarity is reversed (IO/$\overline{M}$).

Z-BUS Signals

R/$\overline{W}$	Read/$\overline{Write}$. This input tells the peripheral whether the present access is a read or write. It is generated by inverting DT/$\overline{R}$ of the 8086/8088.
$\overline{AS}$*	Address Strobe. $\overline{AS}$ is the main clock signal for the Z-BUS peripherals. It is used to initiate bus cycles by latching the address along with $\overline{CS_0}$ and $\overline{INTACK}$. It is generated by inverting ALE of the 8086/8088.
$\overline{DS}$*	Data Strobe. When the Z-BUS peripheral is selected, $\overline{DS}$ gates data onto or from the bus, depending on the state of R/$\overline{W}$. It is generated from the 8086/8088 signals $\overline{RD}$ and $\overline{WR}$ as shown in Figure 1.
$\overline{INTACK}$	Interrupt Acknowledge. When low, this signal tells the peripheral that the present cycle is an Interrupt Acknowledge cycle.
AD_0-AD_7	Address/Data Bus. This bus is connected directly to AD_0-AD_7 of the 8086/8088. It is possible to connect it to AD_8-AD_{15} of the 8086 as long as the 8086 doesn't expect to read an interrupt vector from the peripheral during interrupt acknowledge transactions.
$\overline{CS_0}$,CS_1	Chip selects. $\overline{CS_0}$ is active low and is latched with the rising edge of $\overline{AS}$. CS_1 is active high and is unlatched. In this interface, CS_1 is pulled high while $\overline{CS_0}$ is generated from the address decode logic.
PCLK	Peripheral Clock. This signal does not apply to the Z-FIO. It can also be omitted from the Z-CIO interface if the chip is not used as a timer, its REQUEST/WAIT logic is disabled, and it does not employ deskew timers in its handshake operations. The maximum frequency of PCLK is 4 or 6 MHz, depending on the grade of the component, and it can be asynchronous to the system clock.

*A hardware reset of a Z-BUS peripheral is performed by driving $\overline{AS}$ and $\overline{DS}$ low simultaneously.

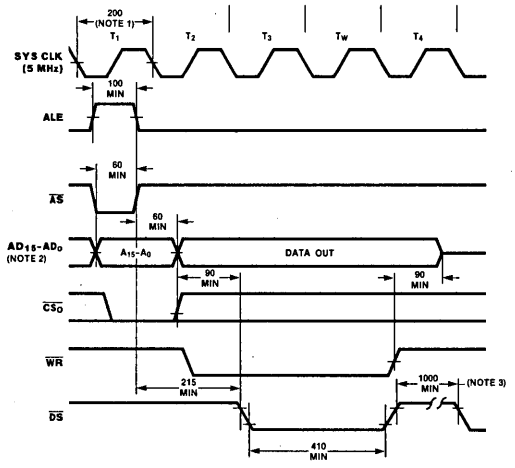
BUS TIMING

Each 8086/8088 bus cycle begins with an ALE pulse, which is inverted to become Address Strobe (AS). The trailing edge of this strobe latches the register address, as well as the states of CS_0 and INTACK within the peripheral. DS is then used to gate data into (write) or from (read) the selected register, provided that an active CS_0 has been latched. To assure proper timing, the AC Characteristics of both the 8086/8088 and the Z-BUS peripherals, must be examined. The paragraphs that follow discuss all of the significant timing considerations that pertain to Read/Write operations in this interface.

ADDRESS AND CHIP SELECT (CS_0) SETUP TIMES. The 4 MHz Z-BUS peripherals require that the stable address setup time prior to AS be at least 30 ns. Since the 5 MHz 8086/8088 is guaranteed to provide valid addresses at least 60 ns before Address Latch Enable (ALE) goes low, this requirement is easily satisfied. The CS_0 setup time is of no concern because the Z8000 peripherals require no CS_0 setup time prior to AS.

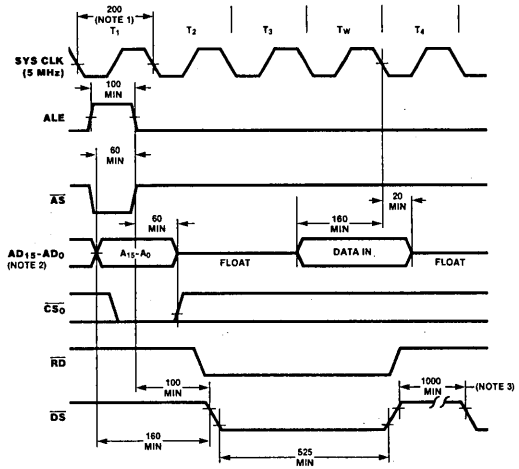
ADDRESS AND CHIP SELECT ($\overline{CS_0}$) HOLD TIMES. The Z-BUS specifications require that the address and $\overline{CS_0}$ remain valid a certain period of time after the rising edge of $\overline{AS}$. These minimum values are 50 and 60 ns respectively for the 4 MHz devices. At 5 MHz, the 8086/8088 will hold its addresses at least 60 ns after ALE goes inactive. Although this is equal to the minimum $\overline{CS_0}$ hold time, a safe margin will be maintained if the propogation delay between the address going invalid to $\overline{CS_0}$ rising, exceeds the propogation delay between ALE falling and $\overline{AS}$ rising.

ADDRESS STROBE ($\overline{AS}$) TO DATA STROBE ($\overline{DS}$) DELAY. The 4 MHz peripherals need a 60 ns delay between $\overline{AS}$ rising and $\overline{DS}$ falling. This parameter is of no concern on write cycles because the D-flop will delay $\overline{DS}$ until the beginning of T_3 (See Figure 2). On read cycles, $\overline{DS}$ follows $\overline{RD}$, so the delay between $\overline{AS}$ and $\overline{DS}$ is approximately equal to the delay between ALE and $\overline{RD}$. If ALE falls at its latest possible point in time and $\overline{RD}$ falls at its earliest point, the time between these two edges would be about 60 ns. This result is unrealistic, however, because a delay in the termination of ALE



Note:
1. All timing in ns.
2. A_{15}-A_8 and AD_7-AD_0 on 8088.
3. This parameter only applies to consecutive accesses to the same device.

Figure 2. Write Cycle Timing



Note:
1. All timing in ns.
2. A_{15}-A_8 and AD_7-AD_0 on 8088.
3. This parameter only applies to consecutive accesses to the same device.

Figure 3. Read Cycle Timing

will always lead to a delay in the activation of $\overline{RD}$. The actual time between the two edges is well over 100 ns.

ADDRESS SETUP TIME TO DATA STROBE ($\overline{DS}$). The 4 MHz Z-CIO and Z-FIO require that the stable address setup time to $\overline{DS}$ be at least 130 ns. Since the delay between $\overline{AS}$ rising and $\overline{DS}$ falling is well over 100 ns, and since the address setup time to $\overline{AS}$ is at least 60 ns, this requirement is easily satisfied.

DATA STROBE ($\overline{DS}$) LOW WIDTH. The minimum Data Strobe Low Width of the 4 MHz Z-BUS peripherals is 390 ns. On read cycles, $\overline{DS}$ will have the same width as $\overline{RD}$, which is at least 325 + $200N_W$ ns, where N_W is the number of wait states in the bus cycle. On write cycles, the D-flop will shorten this minimum width to 210 + N_W 200 ns. One wait state (T_W) in the bus cycle will ensure a sufficiently wide Data Strobe for both types of bus cycles. A discussion of wait state generation is presented in the next section.

WRITE DATA SETUP AND HOLD TIMES. On write cycles, the Z-BUS peripherals require the CPU to put valid data on the bus at least 30 ns before $\overline{DS}$ goes active, and to hold it there at least 30 ns after $\overline{DS}$ terminates. D-flip-flop in Figure 2 guarantees the setup time by delaying the falling edge of $\overline{WR}$ until the next falling edge of SYS CLK (Figure 2.). The Hold Time is also guaranteed because the 8086/8088 will hold valid data at least 90 ns after the termination of $\overline{WR}$.

READ DATA SETUP AND HOLD TIMES. When the 8086/8088 reads from memory or peripherals, it requires them to put valid data on the bus at least 30 ns before the falling edge of SYS CLK at the beginning of T_4. It also requires them to hold the valid data at least 10 ns after this edge. Since the Z-BUS peripherals will provide valid data early in T_W and will hold it until after DS terminates, these parameters are well within the specifications.

VALID ACCESS RECOVERY TIME. This parameter refers to the time between consecutive accesses to a given peripheral. If the 4 MHz Z-SCC is accessed twice, then the time between $\overline{DS}$ rising in the first access and $\overline{DS}$ falling in the second access, must be at least 4 PCLK cycles (i.e. 1000 ns for a 4 MHz PCLK). The Valid Access Recovery Time for the 4 MHz Z-CIO is also 1000 ns, and this can't possibly be violated with a 5 MHz 8086/8088 since there will always be at least one instruction fetch cycle in between I/O accesses, and 1000 ns translates into only 5 clock cycles at 5 MHz.

WAIT STATE GENERATION

The previous section explained why the 4 MHz Z8000 peripherals need to place a wait state in I/O bus cycles when interfaced to the 5 MHz 8086/8088. The following two examples illustrate how wait state generation can be implemented. Since 8086/8088 - based systems typically use an 8284 Clock Chip, which synchronizes the CPU's READY input with the system clock, the task reduces to designing a circuit that will control the RDY1 input of the 8284 (RDY2 is assumed to be grounded).

SINGLE WAIT STATE GENERATION. For the processor to enter a wait state after T_3, the RDY1 input must be low during the falling edge of SYS CLK at the end of T_2. Then, for the processor to enter T_4 after the wait state, RDY1 must be high during the next falling edge of SYS CLK. To make sure that these levels are well-established during their sampling windows, the single wait state generator should toggle RDY1, using the clock edges that precede the sampling edges (Figure 4). The circuit in Figure 5 performs this function and generates a single wait state when one of the $\overline{CS_0}$ inputs is active.

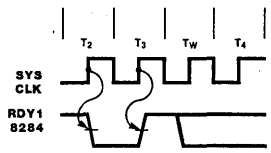


Figure 4. RDY1 Timing for Single Wait State

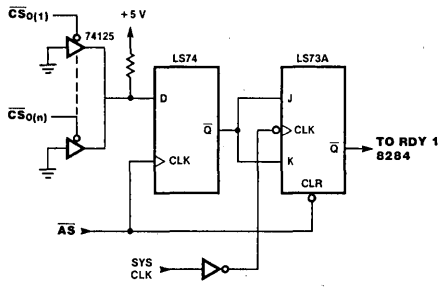


Figure 5. Single Wait State Generator

MULTIPLE WAIT STATE GENERATION. Though Read/Write operations require only one wait state, Interrupt Acknowledge transactions need multiple wait states to allow for daisy-chain settling, which is explained in the next section. The following discussion introduces a multiple wait state generator and serves as a basis for understanding the subsequent Interrupt Acknowledge Circuit.

In the preceeding discussion of the single wait state generator, we established that RDY1 must be high at the end of T_3 for the processor to enter T_4 after the wait state. In general, the 8086/8088 will continue to insert wait states until RDY1 is driven high. In fact, the number of wait states will be equal to the number of clock cycles that RDY1 is held low after the rising clock edge in T_2.

A convenient way to implement a multiple wait state generator is to use a serial shift register such as a 74LS164. Figure 6 shows a wait state generator that requests one wait state on Read/Write cycles, and up to seven wait states on Interrupt Acknowledge cycles. When $\overline{RD}$, $\overline{WR}$, or $\overline{INTA}$ goes active, the 74LS164 is taken out of the clear state and logic "ones" are allowed to shift sequentially from Q_A to Q_H. On Read/Write cycles, RDY1 is held low until the leading "one" appears at Q_B, and on Interrupt Acknowledge cycles, RDY1 is held low until the leading "one" appears at Q_H. The next section shows how $\overline{INTACK}$ can be generated and discusses the complete interrupt interface.

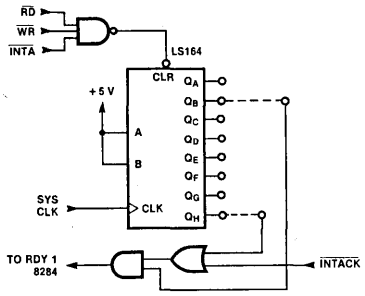


Figure 6. Multiple Wait State Generator

INTERRUPTS

In Figure 1 the $\overline{INTACK}$ input to the Z-BUS peripherals is pulled high. This does not mean that the peripheral can't interrupt the CPU; it just means that it won't respond to the CPU's interrupt acknowledge. The designer can, however, implement a circuit that will drive $\overline{INTACK}$, and allow the 8086/8088 to properly acknowledge the interrupts of the Z-BUS peripherals. This section examines the interrupt acknowledge protocols of the Z-BUS peripherals and the 8086/8088, then proceeds to show how they can be made compatible.

Z-BUS INTERRUPT ACKNOWLEDGE PROTOCOL. The Z-BUS peripherals typically use the daisy-chain technique of priority interrupt control. In this scheme the peripherals are connected together via an interrupt daisy chain formed with their IEI (Interrupt Enable Input) and IEO (Interrupt Enable Output) pins (Figure 7). The interrupt sources within a device are similarly chained together, with the overall effect being a daisy chain connecting all of the interrupt sources. The daisy chain allows higher priority interrupt sources to preempt lower priority sources and, in the case of simultaneous interrupt requests, determines which request will be acknowledged.

In each bus cycle the Z-BUS peripherals use the rising edge of $\overline{AS}$ to latch the state of $\overline{INTACK}$. If a low $\overline{INTACK}$ is latched, then the present cycle is an Interrupt Acknowledge cycle and the daisy chain determines which interrupt source is being acknowledged in the following way. Any interrupt source that has an interrupt pending and is not masked from the chain will hold its IEO low.

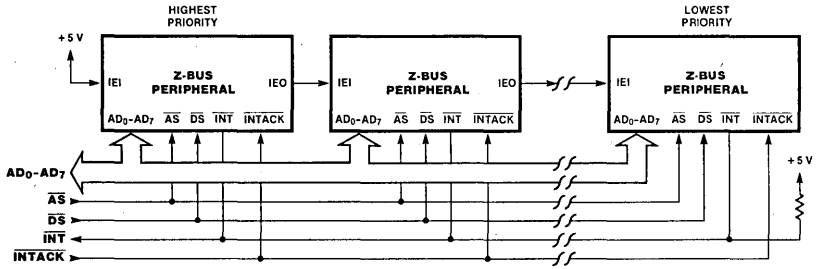


Figure 7. A Z-BUS Interrupt Daisy Chain

Similarly, sources that are currently under service (i.e. have their IUS bit set) will also hold their IEO lines low. All other interrupt sources make IEO follow IEI. The result is that only the highest priority, unmasked source with an interrupt pending will have a high IEI input; only this peripheral will be allowed to transfer its vector to the system bus when the Data Strobe is issued during the Interrupt Acknowledge cycle.

To make sure that the daisy chain has settled by the time $\overline{DS}$ gates the vector onto the bus, the Z-BUS peripherals require a sufficient delay between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$ in INTACK cycles. The amount of delay required can be calculated using Table 2. For a particular daisy chain, the minimum delay is: T_{high} for the highest priority device, plus T_{low} for the lowest priority device, plus T_{mid} for each device in between.

Table 2. Daisy Chain Settling Times for the Z-BUS Peripherals (in ns)

	T_{high}		T_{mid}		T_{low}	
	4MHz	6MHz	4MHz	6MHz	4MHz	6MHz
Z-SCC	250	250	120	100	120	100
Z-CIO	350	250	150	100	100	70
Z-FIO	350	250	150	100	100	70

8086/8088 INTERRUPT ACKNOWLEDGE PROTOCOL. If the 8086/8088 receives an interrupt request (via its INTR pin) while its Interrupt Flag is set, then it will execute an Interrupt Acknowledge sequence. The sequence consists of two identical INTA bus cycles with two idle clock cycles in between (Figure 8). In both bus cycles, $\overline{RD}$ and $\overline{WR}$ remain inactive while an $\overline{INTA}$ strobe is issued with the same timing as a $\overline{WR}$ strobe. The 8086/8088 requires an interrupt vector to appear on AD_0 - AD_7 at least 30 ns before the beginning of T_4 in the second INTA cycle. This protocol is normally used to read vectors from the 8259A Interrupt Controller but it can easily be adapted to the Z-BUS Interrupt Acknowledge Protocol, as illustrated in the following paragraphs.

INTERRUPT ACKNOWLEDGE COMPATIBILITY. The first function of the Interrupt Acknowledge circuit, shown in figure 9, is to generate the Z-BUS $\overline{INTACK}$ signal using $\overline{INTA}$ from the 8086/8088. Since $\overline{INTA}$ goes active after ALE has terminated, the peripherals will not latch an active $\overline{INTACK}$ during the first INTA cycle. However, if the rising edge of $\overline{INTA}$ is used to toggle $\overline{INTACK}$, then an active $\overline{INTACK}$ latches with the rising edge of $\overline{AS}$ in the second INTA cycle. Thus a rising-edge triggered toggle flip-flop, as configured in Figure 9, can be used to generate $\overline{INTACK}$. Figure 10 shows the timing relationship between $\overline{INTA}$ and $\overline{INTACK}$.

The next function of the Interrupt Acknowledge circuit can be broken down into three operations: first, it must cause the CPU to enter a series of wait states after T_3 in the second INTA cycle; then, it must activate $\overline{DS}$ after a sufficient daisy chain settling time; lastly, it must bring the CPU out of the wait state condition when the vector is available on the bus.

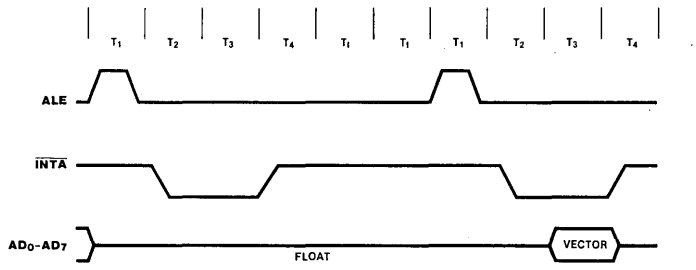


Figure 8. 8086/8088 INTA Sequence

Figure 9 shows how the multiple wait state generator, discussed in the previous section, can be used to perform each of these operations.

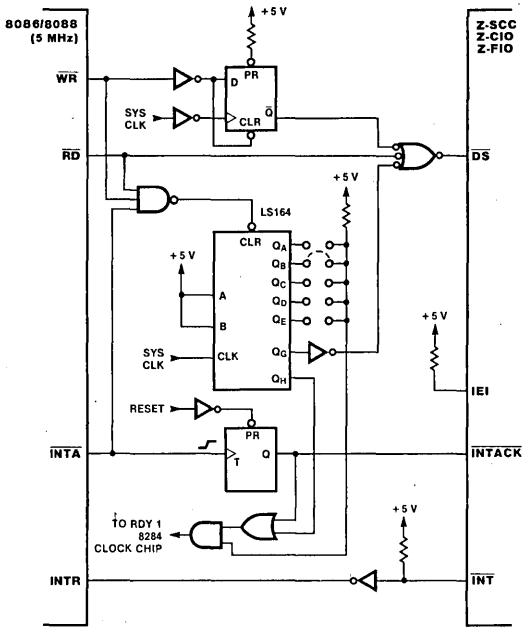


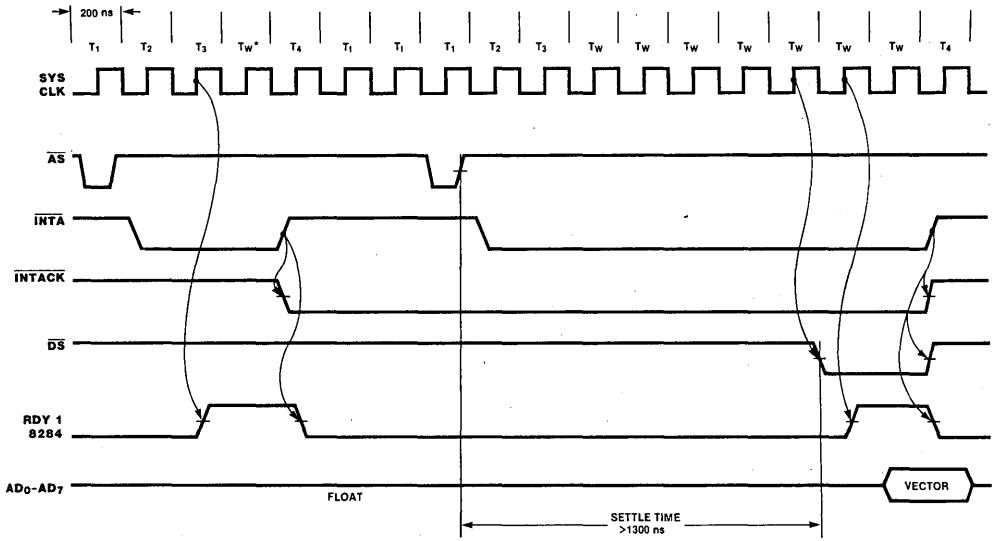
Figure 9. Interrupt Acknowledge Circuit

While $\overline{\text{INTACK}}$ is high the circuit operates normally; the number of wait states it requests is determined by the positioning of the jumper on the Q outputs. When $\overline{\text{INTACK}}$ goes low, it operates as follows: the next activation of $\overline{\text{INTA}}$ brings the shift register out of the clear state, and logic "ones" shift into Q_A until they fill the entire register. When the leading "one" appears at Q_G, $\overline{\text{DS}}$ is driven low; when it appears at Q_H, the CPU is taken out of the wait state condition.

This arrangement takes advantage of the full length of the shift register and provides a daisy-chain settling time of more than 1300 ns, which allows the implementation of a chain with as many as seven Z-BUS devices. Figure 10 shows the timing of the important signals in the Interrupt Acknowledge transaction.

HARDWARE RESET

The designer may want to incorporate a hardware reset in the interface design. This can be accomplished with two NOR gates as shown in Figure 11. The NOR gates allow the system RESET signal to pull $\overline{\text{AS}}$ and $\overline{\text{DS}}$ low simultaneously, and hence put the peripheral in a reset state. A hardware reset is not necessary, however, because all of the peripherals are equipped with software reset commands.



Note:
* This assumes that Q_B is the selected output.

Figure 10. Interrupt Acknowledge Timing

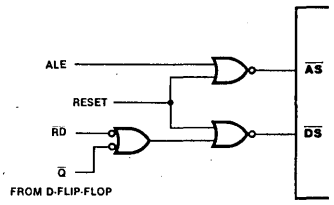


Figure 11. Hardware Reset

SUMMARY

The Z-SCC, Z-CIO, and Z-FIO can easily be designed into 8086/8088 - based systems. Their data and control registers can be mapped directly into the I/O address space, and the Z-BUS control signals can be generated with a minimal amount of external logic. The user can also take advantage of the devices' interrupt control capabilities because a simple interface circuit makes their interrupt structure compatible with that of the 8086/8088.

Interfacing The Z-Bus Peripherals

The Z-Bus peripherals can interface easily to most multiplexed address/data bus structures, while the Z8500 peripherals interface to most non-multiplexed address/data bus structures.

by John D. Kennedy

It has always been customary for designers of microprocessor-based products to look to their CPU suppliers to provide them with the necessary peripheral support components. This is because peripherals have traditionally been designed as members of particular component families and have been bound to parent CPUs. Zilog, Inc. has an array of third-generation peripherals which can free designers from this restriction: the Z8000 peripherals (or Z-Bus peripherals) can interface easily to most multiplexed address/data bus structures, while the Z8500 peripherals (or Universal peripherals) interface to most non-multiplexed address/data bus structures.

The following is a discussion of the Z-Bus and other multiplexed bus structures, along with techniques for interfacing the Z-Bus peripherals. A detailed example is given using Intel's 8086/8088 CPU. Software sequences are also included to provide examples of how to initialize the Z-Bus peripherals for some simple applications.

At the heart of the Z-Bus is a set of multiplexed address/data lines and the signals that control these lines. Multiplexing data and addresses onto the same lines makes more efficient use of pins and facilitates expansion of the number of data and address bits. Multiplexing also allows straightforward addressing of a peripheral's internal registers, which greatly simplifies I/O programming.

The Z-Bus uses a daisy-chained priority mechanism to resolve interrupt and resource requests, thus allowing distributed control of the bus and eliminating the need for separate priority controllers. In an interrupt daisy-chain, for example, the peripherals are connected together via their Interrupt Enable Input (IEI) and Interrupt Enable Output (IEO) pins. The interrupt sources within a device are similarly chained together, with the overall effect being a daisy-chain connecting all the interrupt sources. The daisy chain allows higher priority interrupt sources to preempt lower priority sources and, in the case of simultaneous interrupt requests, determines which request will be acknowledged.

The Zilog microprocessors that can act as master of the Z-Bus include: Z8000, Z800 (16-bit bus ver-

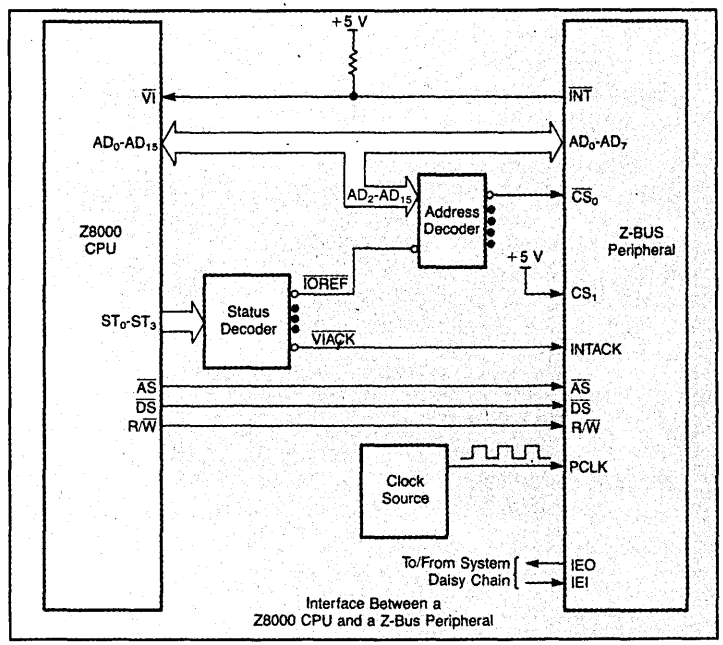


Figure 1: Interface between a Z8000 CPU and a Z-Bus peripheral.

John Kennedy is an Applications Engineer with Zilog Corp., 1315 Dell Ave., Campbell, CA 95008.

sion), and Z8. The Z-Bus peripherals include:

- **Z8016 DTC Direct Memory Access Transfer Controller**
- **Z8030 Z-SCC Serial Communications controller**
- **Z8036 Z-CIO Counter/Timer and Parallel I/O Unit**
- **Z8038 Z-FIO FIFO Input/Output Interface Unit**
- **Z8060 FIFO Buffer Unit and Z-FIO Expander**
- **Z8068 DCP Data Cyphering Processor**

Figure 1 shows an interface between a Z8000 CPU and a Z-Bus peripheral, while **Table 1** gives descriptions of each signal involved in the interface.

Interfacing Non-Zilog CPUs

Among the non-Zilog CPUs that can easily interface to the Z-Bus peripherals are: 8085, 8086, 8088, and NS 16032. The Z-Bus peripherals lend themselves conveniently to designs based on these processors because of the multiplexed address/data bus architecture and the fact that they allow the CPU direct access to all of their data and control registers. Futhermore, the Z-Bus peripherals are asynchronous in the sense that they do not need to be synchronized with the CPU clock; all timing information is provided by the control signals. Of course, the designer must provide the interface logic but, as will be shown, the amount of external hardware is minimal. **Table 2** associates the Z8000 control signals with the analogous control signals of the non-Zilog CPUs.

In the simplest case (where the CPU must merely read from and write to the peripheral), the only bus cycles that need to be considered are the I/O Read and I/O Write bus cycles. The peripherals need an Address Strobe ($\overline{AS}$) to tell them when the address and the state of $\overline{CS_0}$ are valid, a Data Strobe ($\overline{DS}$) to tell them when the bus contains valid data (write) or when to drive data on to the bus (read), and a Read/Write (R/$\overline{W}$) signal to tell them whether the present transaction is a read or write.

All of the processors in **Table 2** generate a signal that is analogous to the Z-Bus $\overline{AS}$. The NS 16032 generates the $\overline{ADS}$ signal, which has the same polarity as AS, while the 8085, 8086, and 8088 generate the ALE (Address Latch Enable) signal which must be inverted if it is to drive $\overline{AS}$. Similarly, all of the processors generate signals from which the Z-Bus $\overline{DS}$ can be derived. If $\overline{DS}$ is not directly available then it can easily be generated using external logic. For example, **Figure 2** (in the hardware example) shows how the $\overline{RD}$ (Read) and $\overline{WR}$ (Write) signals of the 8085, 8086, and 8088 can be used to control $\overline{DS}$. The Z-Bus R/$\overline{W}$ signal can be driven by DDIN of the NS 16032 or by an inversion of DT/$\overline{R}$ of the 8086/8088. Since the 8085 does not generate a DT/$\overline{R}$ signal, it requires an external flip-flop which is set when ALE goes high and reset when $\overline{WR}$ goes low.

In the case of a system with a daisy-chain controlled interrupt scheme, the designer must consider the Interrupt Acknowledge bus cycle. The Z-Bus peripherals interpret the present bus cycle as an Interrupt Acknowledge cycle whenever the $\overline{INTACK}$ input is latched low with the rising edge of AS. During the Interrupt Acknowledge cycle, the daisy-chain determines which interrupt source is being acknowledged in the following way: Any interrupt source that has an interrupt pending and is not masked from the chain will hold its IEO low. Similarly, sources that are currently under service (i.e. have their IUS bit set) will also hold their IEO low. All other interrupt sources make IEO follow IEI. The result is that only the highest priority, unmasked source will have a high IEI input; only this interrupt source will be acknowledged. The hardware example that follows shows how the Z-Bus interrupt ac-

$\overline{INT}$	Interrupt Request (output, open-drain, active Low). This signal is pulled low when the peripheral requests an interrupt. In **Figure 1** it is connected to the VI (Vectored Interrupt) input of the Z8000.
AD_0-AD_{15}	Z-Bus Address/Data lines (bidirectional/3-state). These multiplexed Address/Data lines are used for transfers between the CPU and the peripheral.
$\overline{CS_0}$ and CS_1	Chip Select 0 (input, active Low) and Chip Select 1 (input, active High). CS is latched with the rising edge of AS while CS is unlatched. In **Figure 1**, CS is tied high and CS is generated with the address decode logic.
$\overline{INTACK}$	Interrupt Acknowledge (input, active Low). This signal indicates to the peripheral that an Interrupt Acknowledge cycle is in progress. In **Figure 1**, INTACK is driven by the VIACK (Vectored Interrupt Acknowledge) output of the status decoder.
$\overline{AS}$	Address Strobe (input, active Low). The Z-Bus peripherals sample addresses, INTACK, and CS while AS is low.
$\overline{DS}$	Data Strobe (input, active Low). Data strobe provides timing for the transfer of data into or out of the Z-Bus peripherals.
R/$\overline{W}$	Read/Write (input). R/W indicates that the CPU is performing a read (High) or a write (Low) operation.
PCLK	Peripheral Clock (input, TTL-compatible). The maximum frequency is 4 or 6 MHz depending on the grade of the component, and it can be asynchronous to the system clock.

Table 1: The Z-Bus signals. A hardware reset of a Z-Bus peripheral is performed by drawing AS and DS low simultaneously.

Z8000 CONTROL SIGNALS	ANALOGOUS CONTROL SIGNALS OF NON-ZILOG CPUs 8085/8086/8088	NS 16032
$\overline{AS}$	ALE (Address Latch Enable)	$\overline{ADS}$ (Address Strobe)
$\overline{DS}$	$\overline{RD}$, $\overline{WR}$ (Read, Write Strobes)	$\overline{DS}$/ $\overline{FLT}$ (Data Strobe/Float)
R/$\overline{W}$	DT/$\overline{R}$ (Data Transmit/Receive)[1]	DDIN (Data Direction In)
$\overline{VI}$	INTR (Interrupt)	$\overline{INT}$ (Interrupt)
$\overline{IOREF}$[2] (I/O Reference)	IO/$\overline{M}$ (Input-Output/Memory)[3]	—
$\overline{VIACK}$[2]	$\overline{INTA}$ (Interrupt Acknowledge)	IAM (Interrupt Acknowledge, Master)[2]

Note:
1. DT/$\overline{R}$ is not available on the 8085.
2. Must be decoded from CPU status.
3. M/$\overline{IO}$ on 8086.

Table 2: Z8000 Control Signals and Analogous Control Signals of non-Zilog CPUs.

knowledge protocol can be made compatible with the 8086/8088.

A Hardware Example

The Z-Bus peripherals are easily integrated into, and can satisfy many of the peripheral support requirements of a typical 8086/8088-based system. **Figure 2** shows the interface logic that links the 8086/8088 to Zilog's Serial Communications Controller (Z8030 Z-SCC), Counter/Timer—Parallel I/O Unit (Z8036 Z-CIO), and FIFO I/O Controller (Z8038 Z-FIO). The 8086/8088 is assumed to be running at 5 MHz in the Minimum Mode.

Bus Timing. Each bus cycle begins with an ALE pulse, which is inverted to become $\overline{AS}$. The trailing edge of this strobe latches the address, as well as the states of $\overline{CS_0}$ and $\overline{INTACK}$ within the peripheral. $\overline{DS}$ is then used to gate data to (write) or from (read) the selected register, provided that an active $\overline{CS_0}$ has been latched. The important timing parameters to consider are the Data Strobe Width and the Write Data Setup Time to $\overline{DS}$.

The Z-Bus peripherals have a minimum requirement for the width of Data Strobe: 390 ns for the 4 MHz peripherals, and 250 ns for the 6 MHz peripherals. On read cycles, $\overline{DS}$ has the same width as $\overline{RD}$ which is at least $325 + 200N_W$ ns, where N_W is the number of wait states in the bus cycle. On write cycles, the D flip-flop shortens this minimum width to $210 + 299N_W$ ns. It is clear that one wait state (T_W) in the bus cycle ensures a sufficiently wide Data Strobe for both types of bus cycles.

The Z-Bus specifications also require that the CPU put valid data on the bus prior to the falling edge of $\overline{DS}$ on write cycles. The minimum setup time is 30 ns for the 4 MHz devices, and 20 ns for the 6 MHz devices. The D flip-flop guarantees this setup time by delaying the falling edge of $\overline{WR}$ until the next falling edge of SYS CLK (**Figures 3 and 4** are the Read and Write cycle timing diagrams).

Wait State Generation. Since 8086/8088-based systems typically use an 8284 Clock Chip, which synchronizes the CPU's READY input to the system clock, the task of designing a wait state generator reduces to designing a circuit that controls RDY1 of the 8284 (RDY2 is assumed to be grounded).

For the processor to enter a wait state after T_3, the RDY1 input must be low during the falling edge of SYS CLK at the end of T. Then, for the processor to enter T_4 after the wait state, RDY1 must be high during the next falling edge of SYS CLK. To make sure that these levels are valid within the required set-

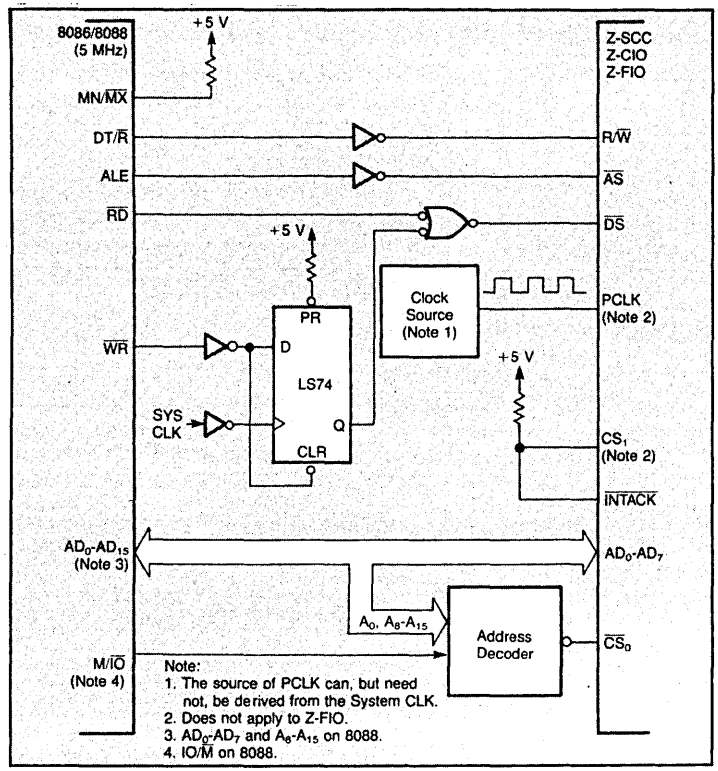


Figure 2: Interface Logic.

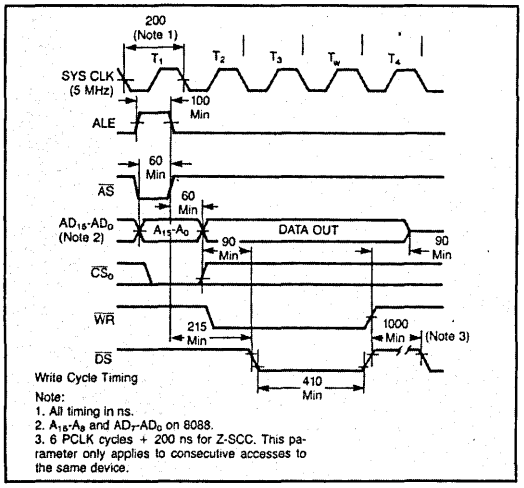


Figure 3: Read and Write cycle timing.

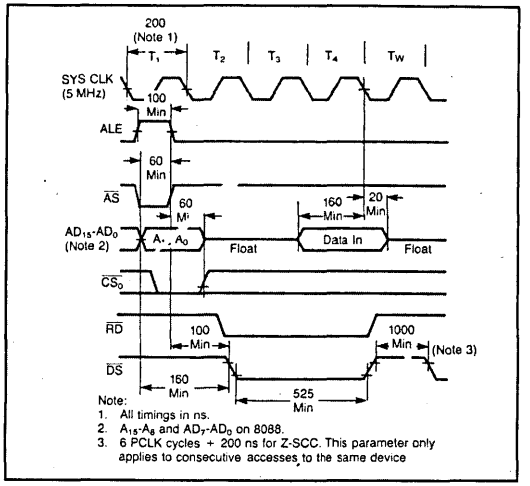


Figure 4: Read and Write cycle timing.

up time, the single wait state generator should toggle RDY1 using the clock edges that precede the sampling edges. The circuit in **Figure 5** performs this function and generates a single wait state whenever one of the $\overline{CS}_O$ inputs is active.

Though read/write operations require only one wait state, Interrupt Acknowledge transactions require multiple wait states to allow for daisy-chain settling, which is explained in the next section. In the preceding discussion of the single wait state generator, we established that RDY1 must be high at the end of T_3 for the processor to enter T_4 after the wait state. In general, the 8086/8088 will continue to insert wait states until RDY1 is driven high, and the number of wait states is equal to the number of clock cycles that RDY1 is held low after the rising clock edge in T_2.

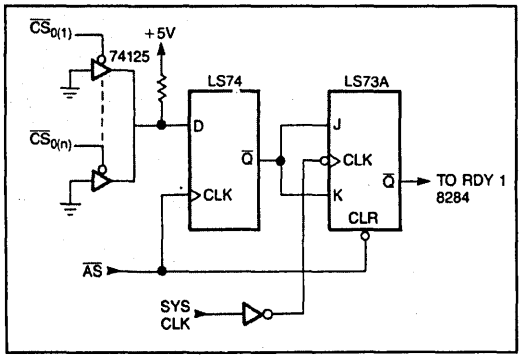
A convenient way to implement a multiple wait state generator is to use a serial shift register such as an 74LS164. **Figure 6** shows a wait state generator that requests one wait state on Read/Write cycles, and up to seven wait states on Interrupt Acknowledge cycles. When $\overline{RD}$, $\overline{WR}$, or $\overline{INTA}$ goes active, the 74LS164 is taken out of the clear state and logic "ones" are allowed to shift sequentially from Q_A to Q_H. On Read/Write cycles, RDY1 is held low until the leading "one" appears at Q_B, and on Interrupt Acknowledge cycles, RDY1 is held low until the leading "one" appears at Q_H. The next section shows how $\overline{INTACK}$ can be generated and discusses the complete interrupt interface.

Interrupts. In **Figure 1** the $\overline{INTACK}$ input to the Z-Bus peripheral is tied high. This does not mean that the peripheral can't interrupt the CPU; it just means that it won't respond to the CPU's interrupt acknowledge. The designer can, however, implement a circuit that will drive $\overline{INTACK}$, and allow the 8086/8088 to properly acknowledge the interrupts of the Z-Bus peripherals.

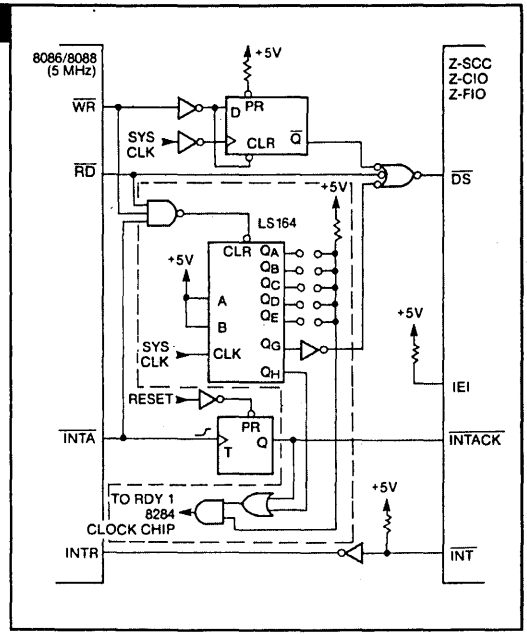
Recall that a Z-Bus interrupt daisy-chain resolves interrupt priority by way of a signal that propagates from high priority interrupt sources to low priority sources during $\overline{INTACK}$ cycles. To make sure that this daisy-chain signal has settled by the time $\overline{DS}$ gates the vector onto the bus, the Z-Bus peripherals require a sufficient delay between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$ during $\overline{INTACK}$ cycles. For a particular daisy-chain, the delay is T_{high} for the highest priority device, plus T_{low} for the lowest priority device, plus T_{mid} for each device in between.

The 8086/8088 Interrupt Acknowledge (INTA) sequence consists of two identical INTA bus cycles with two clock cycles in-between. In both cycles, $\overline{RD}$ and $\overline{WR}$ remain inactive while an $\overline{INTA}$ strobe is issued with the same timing of a $\overline{WR}$ strobe. The 8086/8088 requires an interrupt vector to appear on AD_O—AD_7 at least 30 ns before the beginning of T_4 in the second INTA cycle. This protocol is normally used to read vectors from an 8259 Interrupt Controller but it can easily be adapted to the Z-Bus Interrupt Acknowledge protocol.

The first function of the Interrupt Acknowledge circuit is to generate the Z-Bus $\overline{INTACK}$ signal from $\overline{INTA}$ of the 8086/8088. Since $\overline{INTA}$ goes active after ALE has terminated, the peripherals do not latch an active $\overline{INTACK}$ during the first INTA cycle. However, if the rising edge of $\overline{INTA}$ is used to toggle $\overline{INTACK}$, then an active $\overline{INTACK}$ is latched with the rising edge of $\overline{AS}$ in the second INTA cycle. Thus a rising-edge triggered toggle flip-flop can be used to gen-



▲*Figure 5: Single wait state generator.*



▶*Figure 6: Interrupt acknowledge circuit (dotted line shows wait state generator).*

erate $\overline{\text{INTACK}}$ (see **Figure 6**).

The next function of the Interrupt Acknowledge circuit can be broken down into three operations: first, it must cause the CPU to enter a series of wait states after T_3 in the second INTA cycle; then it must activate $\overline{\text{DS}}$ after a sufficient daisy-chain settling time; lastly, it must bring the CPU out of the wait state condition when the vector is available on the bus.

The multiple wait state generator can be used to perform each of these operations. While $\overline{\text{INTACK}}$ is high it operates normally; the number of wait states it requests is determined by the positioning of the jumper on the Q outputs. When $\overline{\text{INTACK}}$ goes low, it operates as follows: the next activation of $\overline{\text{INTA}}$ brings the shift register out of the clear state, and logic "ones" shift into Q until they fill the entire register. When the leading "one" appears at Q_6, $\overline{\text{DS}}$ is driven low; when it appears at Q_H, the CPU is taken out of the wait state condition.

This arrangement takes advantage of the full length of the shift register and provides a daisy-chain settling time of more than 1300 ns, which allows the implementation of a chain with as many as seven Z-Bus devices.

Hardware reset. The designer may want to incorporate a hardware reset in the interface design. This can be accomplished with two NOR gates. The NOR gates allow a system RESET signal to pull $\overline{\text{AS}}$ and $\overline{\text{DS}}$ low simultaneously, and hence put the peripheral in a reset state. A hardware reset is not necessary, however, because all of the peripherals are equipped with software reset commands.

Designers need not be limited to peripherals offered by their CPU suppliers. Zilog's Z-Bus peripherals are designed with the flexibility to interface to most multiplexed address/data bus structures. □

Using SCC With Z8000 In SDLC Protocol

Zilog

Application Note

October 1982

This application note describes the use of the Z8030 Serial Communications Controller (Z-SCC) with the Z8000™ CPU to implement a communications controller in a Synchronous Data Link Control (SDLC) mode of operation. In this application, the Z8002 CPU acts as a controller for the Z-SCC. This application note also applies to the non-multiplexed Z8530.

One channel of the Z-SCC communicates with the remote station in Half Duplex mode at 9600 bits/second. To test this application, two Z8000 Development Modules are used. Both are loaded with the same software routines for initialization and for transmitting and receiving messages. The main program of one module requests the transmit routine to send a message of the length indicated by the 'COUNT' parameter. The other system receives the incoming data stream, storing the message in its resident memory.

DATA TRANSFER MODES

The Z-SCC system interface supports the following data transfer modes:

- **Polled Mode.** The CPU periodically polls the Z-SCC status registers to determine if a received character is available, if a character is needed for transmission, and if any errors have been detected.
- **Interrupt Mode.** The Z-SCC interrupts the CPU when certain previously defined conditions are met.
- **Block/DMA Mode.** Using the Wait/Request ($\overline{W}/\overline{REQ}$) signal, the Z-SCC introduces extra wait cycles in order to synchronize the data transfer between a controller or DMA and the Z-SCC.

The example given here uses the block mode of data transfer in its transmit and receive routines.

SDLC PROTOCOL

Data communications today require a communications protocol that can transfer data quickly and reliably. One such protocol, Synchronous Data Link Control (SDLC), is the link control used by the IBM Systems Network Architecture (SNA) communications package. SDLC is a subset of the International Standards Organization (ISO) link control called High-Level Data Link Control (HDLC), which is used for international data communications.

SDLC is a bit-oriented protocol (BOP). It differs from byte-control protocols (BCPs), such as Bisync, in that it uses only a few bit patterns for control functions instead of several special character sequences. The attributes of the SDLC protocol are position dependent rather than character dependent, so the data link control is determined by the position of the byte as well as by the bit pattern.

A character in SDLC is sent as an octet, a group of eight bits. Several octets combine to form a message frame, in which each octet belongs to a particular field. Each message contains: opening flag, address, control, information, Frame Check Sequence (FCS), and closing flag (figure 1).

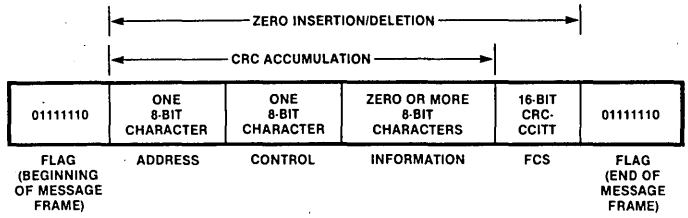


Figure 1. Fields of the SDLC Transmission Frame

Both flag fields contain a unique binary pattern, 01111110, which indicates the beginning or the end of the message frame. This pattern simplifies the hardware interface in receiving devices so that multiple devices connected to a common link do not conflict with one another. The receiving devices respond only after a valid flag character has been detected. Once communication is established with a particular device, the other devices ignore the message until the next flag character is detected.

The address field contains one or more octets, which are used to select a particular station on the data link. An address of eight 1s is a global address code that selects all the devices on the data link. When a primary station sends a frame, the address field is used to select one of several secondary stations. When a secondary station sends a message to the primary station, the address field contains the secondary station address, i.e., the source of the message.

The control field follows the address field and contains information about the type of frame being sent. The control field consists of one octet that is always present.

The information field contains any actual transferred data. This field may be empty or it may contain an unlimited number of octets. However, because of the limitations of the error-checking algorithm used in the frame-check sequence, however, the maximum recommended block size is approximately 4096 octets.

The frame check sequence field follows the information or control field. The FCS is a 16-bit Cyclic Redundancy Check (CRC) of the bits in the address, control, and information fields. The FCS is based on the CRC-CCITT code, which uses the polynomial $(x^{16} + x^{12} + x^5 + 1)$. The Z8030 Z-SCC contains the circuitry necessary to generate and check the FCS field.

Zero insertion and deletion is a feature of SDLC that allows any data pattern to be sent. Zero insertion occurs when five consecutive 1s in the data pattern are transmitted. After the fifth 1, a 0 is inserted before the next bit is sent. The extra 0 does not affect the data in any way and is deleted by the receiver, thus restoring the original data pattern.

Zero insertion and deletion insures that the data stream will not contain a flag character or abort sequence. Six 1s preceded and followed by 0s indicate a flag sequence character. Seven to fourteen 1s signify an abort; 15 or more 1s indicate an idle (inactive) line. Under these three conditions, zero insertion and deletion are inhibited. Figure 2 illustrates the various line conditions.

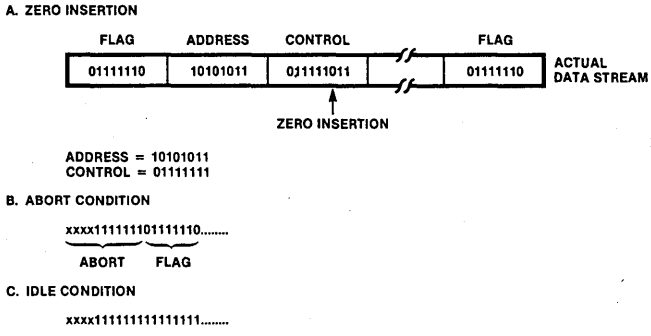


Figure 2. Bit Patterns for Various Line Conditions

The SDLC protocol differs from other synchronous protocols with respect to frame timing. In Bisync mode, for example, a host computer might temporarily interrupt transmission by sending sync characters instead of data. This suspended condition continues as long as the receiver does not time out. With SDLC, however, it is invalid to send flags in the middle of a frame to idle the line. Such action causes an error condition and disrupts orderly operation. Thus, the transmitting device must send a complete frame without interruption. If a message cannot be transmitted completely, the primary station sends an abort sequence and restarts the message transmission at a later time.

SYSTEM INTERFACE

The Z8002 Development Module consists of a Z8002 CPU, 16k words of dynamic RAM, 2k words of EPROM monitor, a Z80A SIO providing dual serial ports, a Z80A CTC peripheral device providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire wrap area for prototyping. The block diagram is depicted in Figure 3. Each of the peripherals in the development module is connected in a prioritized daisy chain configuration. The Z-SCC is included in this configuration by tying its IEI line to the IEO line of another device, thus making it one step lower in interrupt priority compared to the other device.

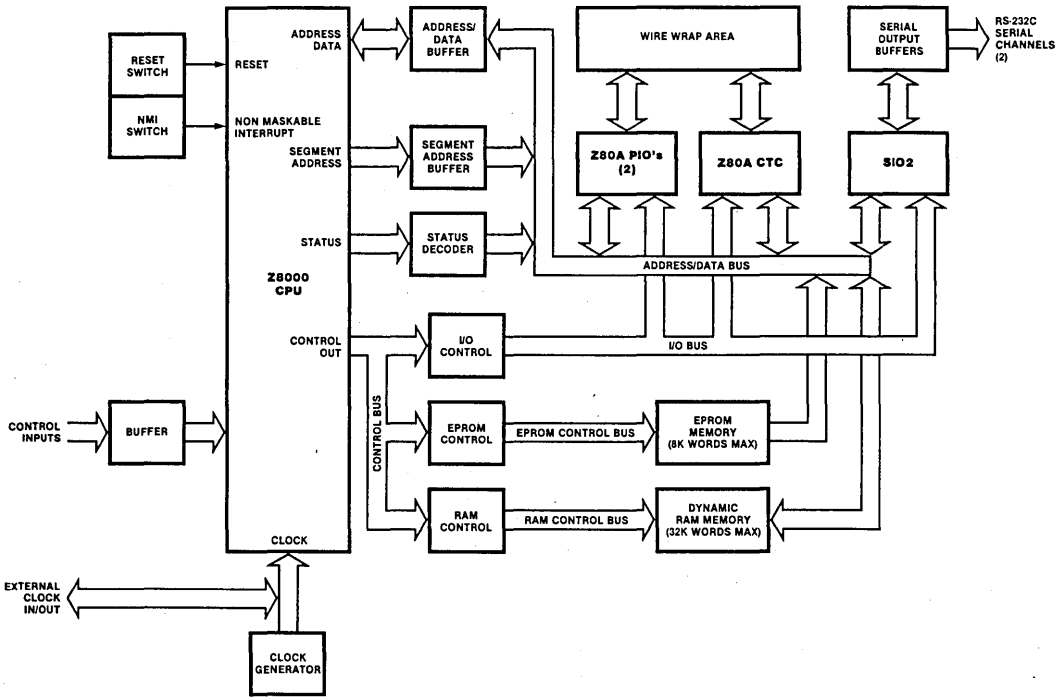


Figure 3. Block Diagram of Z8000 DM

Two Z8000 Development Modules containing Z-SCCs are connected as shown in Figure 4 and Figure 5. The Transmit Data pin of one is connected to the Receive Data pin of the other and vice versa. The Z8002 is used as a host CPU for loading the modules' memories with software routines.

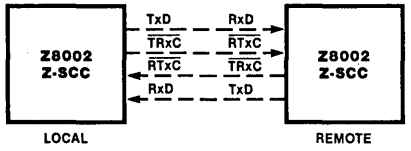


Figure 4. Block Diagram of Two Z8000 CPUs

The Z8002 CPU can address either of the two bytes contained in 16-bit words. The CPU uses an even address (16 bits) to access the most significant byte of a word and an odd address for the least significant byte of a word.

When the Z8002 CPU uses the lower half of the Address/Data bus (AD_0-AD_7 the least significant byte) for byte read and write transactions during I/O operations, these transactions are performed between the CPU and I/O ports located at odd I/O addresses. Since the Z-SCC is attached to the CPU on the lower half of the A/D bus, its registers must appear to the CPU at odd I/O addresses. To achieve this, the Z-SCC can be programmed to select its internal registers using lines AD1-AD5. This is done either automatically with the Force Hardware Reset command in WR9 or by sending a Select Shift Left Mode command to WR0B in channel B of the Z-SCC. For this application, the Z-SCC registers are located at I/O port address 'FExx'. The Chip Select signal ($\overline{CS0}$) is derived by decoding I/O address 'FE' hex from lines AD_8-AD_{15} of the controller.

To select the read/write registers automatically, the Z-SCC decodes lines AD_1-AD_5 in Shift Left mode. The register map for the Z-SCC is depicted in Table 1.

Table 1. Register Map

Address (hex)	Write Register	Read Register
FE01	WR0B	RR0B
FE03	WR1B	RR1B
FE05	WR2	RR2B
FE07	WR3B	RR3B
FE09	WR4B	
FE0B	WR5B	
FE0D	WR6B	
FE0F	WR7B	
FE11	B DATA	B DATA
FE13	WR9	
FE15	WR10B	RR10B
FE17	WR11B	
FE19	WR12B	RR12B
FE1B	WR13B	RR13B
FE1D	WR14B	
FE1F	WR15B	RR15B
FE21	WR0A	RR0A
FE23	WR1A	RR1A
FE25	WR2	RR2A
FE27	WR3A	RR3A
FE29	WR4A	
FE2B	WR5A	
FE2D	WR6A	
FE2F	WR7A	
FE31	A DATA	A DATA
FE33	WR9	
FE35	WR10A	RR10A
FE37	WR11A	
FE39	WR12A	RR12A
FE3B	WR13A	RR13A
FE3D	WR14A	
FE3F	WR15A	RR15A

INITIALIZATION

The Z-SCC can be initialized for use in different modes by setting various bits in its write registers. First, a hardware reset must be

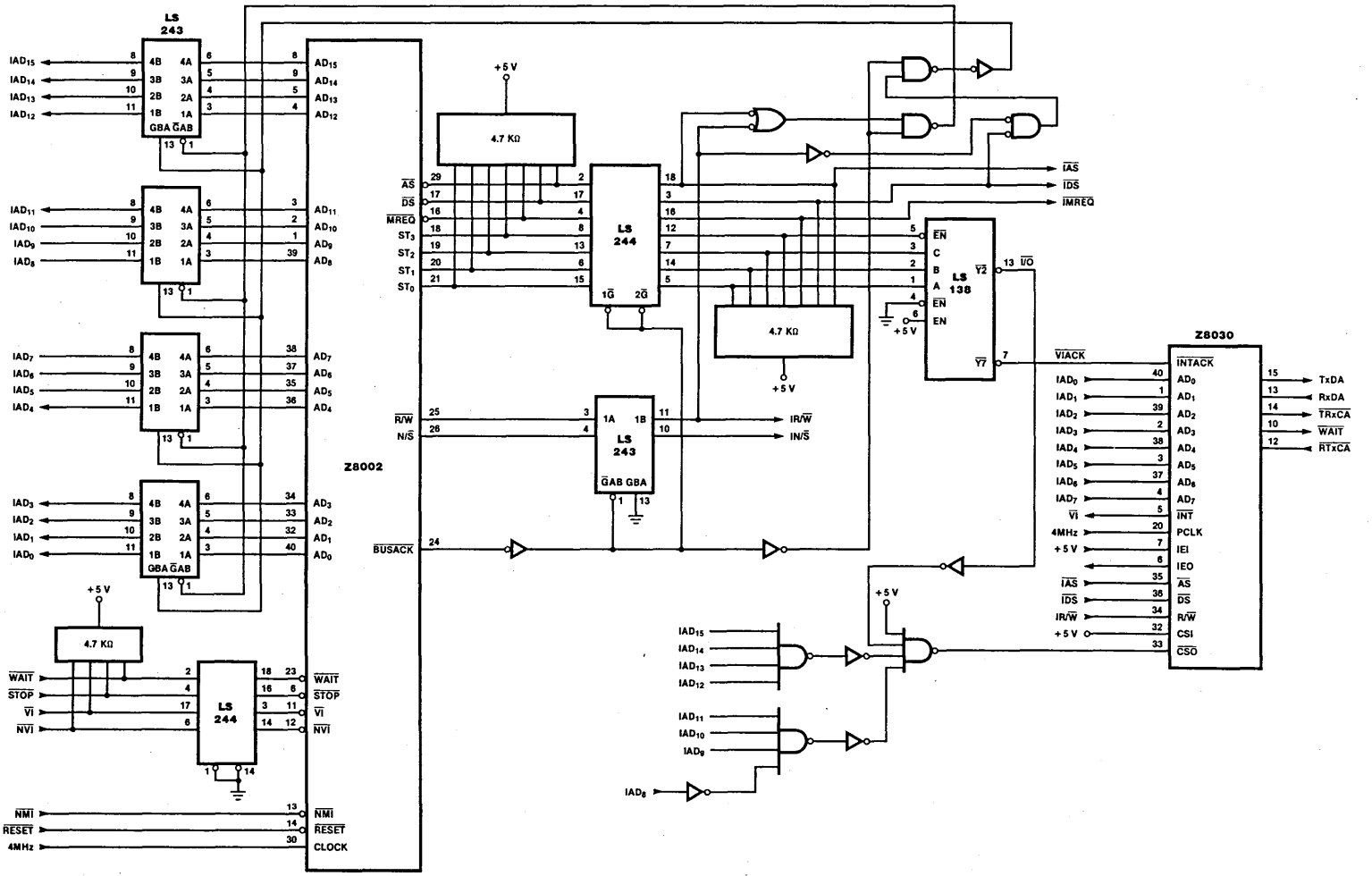


Figure 5. Z8002 With SCC

performed by setting bits 7 and 6 of WR9 to one; the rest of the bits are disabled by writing a logic zero.

SDLC protocol is established by selecting a SDLC mode, sync mode enable, and a x1 clock in WR4. A data rate of 9600 baud, NRZ encoding, and a character length of eight bits are among the other options that are selected in this example (Table 2).

Note that WR9 is accessed twice, first to perform a hardware reset and again at the end of the initialization sequence to enable interrupts. The programming sequence depicted in Table 2 establishes the necessary parameters for the receiver and transmitter so that they are ready to perform communication tasks when enabled.

Table 2. Programming Sequence for Initialization

Register	Value (hex)	Effect
WR9	C0	Hardware reset
WR4	20	x1 clock, SDLC mode, sync mode enable
WR10	80	NRZ, CRC preset to one
WR6	AB	Any station address e.g. "AB"
WR7	7E	SDLC flag (01111110) = "7E"
WR2	20	Interrupt vector "20"
WR11	16	Tx clock from BRG output, $\overline{TRxC}$ pin = BRG out
WR12	CE	Lower byte of time constant = "CE" for 9600 baud
WR13	0	Upper byte = 0
WR14	03	BRG source bit = 1 for PCLK as input, BRG enable
WR15	00	External Interrupt Disable
WR5	60	Transmit 8 bits/character SDLC CRC
WR3	C1	Rx 8 bits/character, Rx enable (Automatic Hunt mode)
WR1	08	RxInt on 1st char & sp. cond., ext int. disable
WR9	09	MIE, VIS, status Low

The Z8002 CPU must be operated in System mode to execute privileged I/O instructions. So the Flag and Control Word (FCW) should be loaded with system normal (S/$\overline{N}$), and the Vectored Interrupt Enable (VIE) bits set. The Program Status Area Pointer (PSAP) is loaded with the address %4400 using the Load Control instruction (LDCTL). If the Z8000 Development Module is intended to be used, the PSAP need not be loaded by the programmer because the development module's monitor loads it automatically after the NMI button is pressed.

Since VIS and Status Low are selected in WR9, the vectors listed in Table 3 will be returned during the Interrupt Acknowledge cycle. Of the four interrupts listed, only two, Ch A Receive Character Available and Ch A Special Receive Condition, are used in the example given here.

Table 3. Interrupt Vectors

Vector (hex)	PS Address* (hex)	Interrupt
28	446E	Ch A Transmit Buffer Empty
2A	4472	Ch A External Status Change
2C	4476	Ch A Receive Char. Available
2E	447A	Ch A Special Receive Condition

*Assuming that PSAP has been set to 4400 hex, "PS Address" refers to the location in the Program Status Area where the service routine address is stored for that particular interrupt.

TRANSMIT OPERATION

To transmit a block of data, the main program calls up the transmit data routine. With this routine, each message block to be transmitted is stored in memory, beginning with location 'TBUF'. The number of characters contained in each block is determined by the value assigned to the 'COUNT' parameter in the main module.

To prepare for transmission, the routine enables the transmitter and selects the Wait On Transmit function; it then enables the wait function. The Wait On Transmit function indicates to the CPU whether or not the Z-SCC is ready to accept data from the CPU. If the CPU attempts to send data to the Z-SCC when the transmit buffer is full, the Z-SCC asserts its $\overline{Wait}$ line and keeps it Low until the buffer is empty. In response, the CPU extends its I/O cycles until the $\overline{Wait}$ line goes inactive, indicating that the Z-SCC is ready to receive data.

The CRC generator is reset and the Transmit CRC bit is enabled before the first character is sent, thus including all the characters sent to the Z-SCC in the CRC calculation.

The Z-SCC's transmit underrun/EOM latch must be reset sometime after the first character is transmitted by writing a Reset Tx Underrun/EOM command to WR0. When this latch is reset, the Z-SCC automatically appends the CRC characters to the end of the message in the case of an underrun condition.

Finally, a three-character delay is introduced at the end of the transmission, which allows the Z-SCC sufficient time to transmit the last data byte and two CRC characters before disabling the transmitter.

RECEIVE OPERATION

Once the Z-SCC is initialized, it can be prepared to receive the message. First, the receiver is enabled, placing the Z-SCC in Hunt mode and thus setting the Sync/Hunt bit in status register RR0 to 1. In Hunt mode, the receiver searches the incoming data stream for flag characters. Ordinarily, the receiver transfers all the data received between flags to the receive data FIFO. If the receiver is in Hunt mode, however, no data transfer takes place until an opening flag is received. If an abort sequence is received, the receiver automatically re-enters Hunt mode. The Hunt status of the receiver is reported by the Sync/Hunt bit in RR0.

The second byte of an SDLC frame is assumed by the Z-SCC to be the address of the secondary stations for which the frame is intended. The Z-SCC provides several options for handling this address. If the Address Search Mode bit D2 in WR3 is set to zero, the address recognition logic is disabled and all the received data bytes are transferred to the receive data FIFO. In this mode, software must perform any address recognition. If the Address Search Mode bit is set to one, only those frames with addresses that match the address programmed in WR6 or the global address (all 1s) will be transferred to the receive data FIFO. If the Sync Character Load Inhibit bit (D1) in WR3 is set to zero, the address comparison is made across all eight bits of WR6. The comparison can be modified so that only the four most significant bits of WR6 need match the received address. This alteration is made by setting the Sync Character Load Inhibit bit to one. In this mode, the address field is still eight bits wide and is transferred to the FIFO in the same manner as the data. In this application, the address search is performed.

When the address match is accomplished, the receiver leaves the Hunt mode and establishes the Receive Interrupt on First Character mode. Upon detection of the receive interrupt, the CPU generates an Interrupt Acknowledge Cycle. The Z-SCC returns the programmed vector %2C. This vector points to the location %4472 in the Program Status Area which contains the receive interrupt service routine address.

The receive data routine is called from within the receive interrupt service routine. While expecting a block of data, the Wait On Receive function is enabled. Receive read buffer RR8 is read and the characters are stored in memory location RBUF. The Z-SCC in SDLC mode automatically enables the CRC checker for all data between opening and closing flags and ignores the Receive CRC Enable bit (D3) in WR3. The result of the CRC calculation for the entire frame in RR1 becomes valid only when the End Of Frame bit is set in RR1. The processor does not use the CRC bytes, because the last two bits of the CRC are never transferred to the receive data FIFO and are not recoverable.

When the Z-SCC recognizes the closing flag, the contents of the Receive Shift register are transferred to the receive data FIFO, the Residue Code (not applicable in this application) is latched, the CRC error bit is latched in the status FIFO, and the End Of Frame bit is set in the receive status FIFO. When the End Of Frame bit reaches the top of the FIFO, a special receive condition interrupt occurs. The special receive condition register RR1 is read to determine the result of the CRC calculation. If the CRC error bit is zero, the frame received is assumed to be correct; if the bit is 1, an error in the transmission is indicated.

Before leaving the interrupt service routine, Reset Highest IUS (Interrupt Under Service), Enable Interrupt on Next Receive Character, and Enter Hunt Mode commands are issued to the Z-SCC.

If receive overrun error is made, a special condition interrupt occurs. The Z-SCC presents vector %2E to the CPU, and the service routine located at address %447A is executed. Register RR1 is read to determine which error occurred. Appropriate action to correct the error should be taken by the user at this point. Error Reset and Reset Highest IUS commands are given to the Z-SCC before returning to the main program so that the other lower-priority interrupts can occur.

In addition to searching the data stream for flags, the receiver also scans for seven consecutive 1s, which indicates an abort condition. This condition is reported in the Break/Abort bit (D7) in RR0. This is one of many possible external status conditions. As a result transitions of this bit can be programmed to cause an external status interrupt. The abort condition is terminated when a zero is received, either by itself or as the leading zero of a flag. The receiver leaves Hunt mode only when a flag is found.

SOFTWARE

Software routines are presented in the following pages. These routines can be modified to include various other options (e.g., SDLC Loop, Digital Phase Locked Loop etc.). By modifying the WR10 register, different encoding methods (e.g., NRZI, FM0, FM1) other than NRZ can be used.

Appendix

Software Routines

```
plzasm    1.3
LOC     OBJ CODE    STMT SOURCE STATEMENT

                      1
                      2
                      3         SDLC MODULE
                        $LISTON $TTY
                        CONSTANT
                        WR0A    :=  %FE21               !BASE ADDRESS FOR WR0 CHANNEL A!
                        RR0A    :=  %FE21               !BASE ADDRESS FOR RR0 CHANNEL A!
                        RBUF    :=  %5400               !BUFFER AREA FOR RECEIVE CHARACTER!
                        PSAREA  :=  %4400               !START ADDRESS FOR PROGRAM STAT AREA!
                        COUNT   :=   12                 !NO. OF CHAR. FOR TRANSMIT ROUTINE!
0000                    GLOBAL MAIN PROCEDURE
                        ENTRY

0000 7601                       LDA     R1,PSAREA
0002 4400
0004 7D1D                       LDCTL   PSAPOFF,R1      !LOAD PSAP!
0006 2100                       LD      R0,#%5000
0008 5000
000A 3310                       LD      R1(#%1C),R0     !FCW VALUE(%5000) AT %441C FOR VECTORED!
000C 001C                                               !INTERRUPTS!
000E 7600                       LDA     R0,REC
0010 00D6'
0012 3320                       LD      R1(#%76),R0     !EXT. STATUS SERVICE ADDR. AT %4476 IN!
0014 0076                                               !PSA!
0016 7600                       LDA     R0,SPCOND
0018 00FA'
001A 3310                       LD      R1(#%7A),R0     !SP.COND.SERVICE ADDR AT %447A IN PSA!
001C 007A
001E 5F00                       CALL    INIT
0020 0034'
0022 5F00                       CALL    TRANSMIT
0024 008C'
0026 E8FF                       JR      $

0028 AB                 TBUF:   BVAL    %AB             !STATION ADDRESS!
0029 48                         BVAL    'H'
002A 45                         BVAL    'E'
002B 4C                         BVAL    'L'
002C 4C                         BVAL    'L'
002D 4F                         BVAL    'O'
002E 20                         BVAL    ' '
002F 54                         BVAL    'T'
0030 48                         BVAL    'H'
0031 45                         BVAL    'E'
0032 52                         BVAL    'R'
0033 45                         BVAL    'E'

0034                            END     MAIN
```

```
                        !****************** INITIALIZATION ROUTINE FOR Z-SCC ***********************!

0034                            GLOBAL  INIT PROCEDURE
                                ENTRY
0034 2100                       LD      R0,#15          !NO.OF PORTS TO WRITE TO!
0036 000F
0038 7602                       LDA     R2,SCCTAB       !ADDRESS OF DATA FOR PORTS!
003A 004E'
003C 2101               ALOOP:  LD      R1,#WR0A
003E FE21
0040 0029                       ADDB    RL1,@R2
0042 A920                       INC     R2
0044 3A22                       OUTIB   @R1,@R2,R0      !POINT TO WR0A,WR1A ETC THRO LOOP!
0046 0018
0048 8D04                       TEST    R0              !END OF LOOP?!
004A EEF8                       JR      NZ,ALOOP        !NO,KEEP LOOPING!
004C 9E08                       RET
004E 12                 SCCTAB: BVAL    2*9
004F C0                         BVAL    %C0             !WR9=HARDWARE RESET!
0050 08                         BVAL    2*4
0051 20                         BVAL    %20             !WR4=X1 CLK,SDLC,SYNC MODE!
0052 14                         BVAL    2*10
0053 80                         BVAL    %80             !WR10=CRC PRESET ONE,NRZ,FLAG ON IDLE,!
                                                            !FLAG ON UNDERRUN!
0054 0C                         BVAL    2*6
0055 AB                         BVAL    %AB             !WR6= ANY ADDRESS FOR SDLC STATION!
0056 0E                         BVAL    2*7
0057 7E                         BVAL    %7E             !WR7=SDLC FLAG CHAR!
0058 04                         BVAL    2*2
0059 20                         BVAL    %20             !WR2=INT VECTOR %20!
005A 16                         BVAL    2*11
005B 16                         BVAL    %16             !WR11=Tx CLOCK & TRxC OUT=BRG OUT!
005C 18                         BVAL    2*12
005D CE                         BVAL    %CE             !WR12= LOWER TC=CE!
005E 1A                         BVAL    2*13
005F 00                         BVAL    0               !WR13= UPPER TC=0!
0060 1C                         BVAL    2*14
0061 03                         BVAL    %03             !WR14=BRG ON,BRG SRC=PCLK!
0062 1E                         BVAL    2*15
0063 00                         BVAL    %00             !WR15=EXT INT. DISABLE!
0064 0A                         BVAL    2*5
0065 60                         BVAL    %60             !WR5=Tx 8 BITS/CHAR, SDLC CRC!
0066 06                         BVAL    2*3
0067 C5                         BVAL    %C5             !WR3=ADDR SRCH,REC ENABLE!
0068 02                         BVAL    2*1
0069 08                         BVAL    %08             !WR1=RX INT ON 1ST & SP COND,!
                                                            !EXT INT DISABLE!
006A 12                         BVAL    2*9
006B 09                         BVAL    %09             !WR9= MIE,VIS,STATUS LOW!
006C                    END     INIT


                        !****************** RECEIVE ROUTINE ************************************!

                        !                   RECEIVE A BLOCK OF MESSAGE                          !

006C                            GLOBAL  RECEIVE PROCEDURE
                                ENTRY
006C C828                       LDB     RL0,#%28        !WAIT ON RECV.!
006E 3A86                       OUTB    WR0A+2,RL0
0070 FE23
0072 6008                       LDB     RL0,%A8
0074 00A8
0076 3A86                       OUTB    WR0A+2,RL0      !ENABLE WAIT FNC. SP. COND. INT!
0078 FE23
007A 2101                       LD      R1,#RR0A+16
007C FE31
007E 2102                       LD      R2,#COUNT+2     !COUNT+2 CHARACTERS TO READ!
0080 000E
0082 2103                       LD      R3,#RBUF        !RECEIVE BUFFER IN MEMORY!
0084 5400
0086 3A18                       INDRB   @R3,@R1,R2      !READ THE ENTIRE MESSAGE!
0088 0230
008A 9E08                       RET
008C                    END RECEIVE
```

```
                        |*****************  TRANSMIT ROUTINE ************************************|
                        |                   SEND A BLOCK OF EIGHT  DATA CHARACTERS               |
                        |                   THE BLOCK STARTS AT LOCATION TBUF                    |

008C                            GLOBAL  TRANSMIT PROCEDURE
                                ENTRY
008C 2102                               LD      R2,#TBUF        !PTR TO START OF BUFFER!
008E 0028'
0090 C868                               LDB     RL0,#%68
0092 3A86                               OUTB    WR0A+10,RL0     !ENABLE TRANSMITTER!
0094 FE2B
0096 C800                               LDB     RL0,#%00        !WAIT ON TRANSMIT!
0098 3A86                               OUTB    WR0A+2,RL0
009A FE23
009C C888                               LDB     RL0,#%88
009E 3A86                               OUTB    WR0A+2,RL0      !WAIT ENABLE!
00A0 FE23
00A2 C880                               LDB     RL0,#%80
00A4 3A86                               OUTB    WR0A,RL0        !RESET TxCRC GENERATOR!
00A6 FE21
00A8 2101                               LD      R1,#WR0A+16     !WR8A SELECTED!
00AA FE31
00AC 2100                               LD      R0,#1
00AE 0001
00B0 C869                               LDB     RL0,#%69        !SDLC CRC!
00B2 3A86                               OUTB    WR0A+10,RL0     !WR5A=TxCRC ENABLE!
00B4 FE2B
00B6 3A22                               OTIRB   @R1,@R2,R0      !SEND ADDRESS!
00B8 0010
00BA C8C0                               LDB     RL0,#%C0
00BC 3A86                               OUTB    WR0A,RL0        !RESET TxUND/EOM LATCH!
00BE FE21
00C0 2100                               LD      R0,#COUNT-1
00C2 000B
00C4 3A22                               OTIRB   @R1,@R2,R0      !SEND MESSAGE!
00C6 0010
00C8 2100                               LD      R0,#926         !CREATE DELAY BEFORE DISABLING!
00CA 039E
00CC F081                       DEL:    DJNZ    R0,DEL          !TRANSMITTER SO THAT CRC CAN BE!
00CE C800                               LDB     RL0,#0          !SENT!
00D0 3A86                               OUTB    WR0A+10,RL0     !DISABLE TRANSMITTER!
00D2 FE2B
00D4 9E08                               RET
00D6                            END TRANSMIT


                        |*************** RECEIVE INT. SERVICE ROUTINE ***************************|

00D6                              GLOBAL REC PROCEDURE
                                  ENTRY
00D6 93F3                               PUSH    @R15,R3
00D8 93F2                               PUSH    @R15,R2
00DA 93F1                               PUSH    @R15,R1
00DC 93F0                               PUSH    @R15,R0
00DE 3A94                               INB     RL1,RR0A        !READ STATUS REG RR0A!
00E0 FE21
00E2 A690                               BITB    RL1,#0          !TEST IF Rx CHAR SET!
00E4 E602                               JR      Z,RESET         !YES CALL RECEIVE ROUTINE!
00E6 5F00                               CALL    RECEIVE
00E8 006C'
00EA C838                       RESET:  LDB     RL0,#%38
00EC 3A86                               OUTB    WR0A,RL0        !RESET HIGHEST IUS!
00EE FE21
00F0 97F0                               POP     R0,@R15
00F2 97F1                               POP     R1,@R15
00F4 97F2                               POP     R2,@R15
00F6 97F3                               POP     R3,@R15
00F8 7B00                               IRET
00FA                            END REC
```

```
                !************* SPECIAL CONDITION INTERRUPT SERVICE ROUTINE ***************!

00FA                    GLOBAL SPCOND PROCEDURE
                        ENTRY

00FA 93F0                       PUSH    @R15,R0
00FC 3A84                       INB     RL0,RR0A+2      !READ ERRORS!
00FE FE23
0100 A687                       BITB    RL0,#7          !END OF FRAME ?!
                        !PROCESS OVERRUN,FRAMING ERRORS IF ANY!
0102 E603                       JR      Z,RESE
0104 C820                       LDB     RL0,#%20
0106 3A86                       OUTB    WR0A,RL0        ! YES,ENABLE INT ON NEXT REC CHAR!
0108 FE21
010A C830               RESE:   LDB     RL0,#%30
010C 3A86                       OUTB    WR0A,RL0        !ERROR RESET!
010E FE21
0110 C808                       LDB     RL0,#%08
0112 3A86                       OUTB    WR0A+2,RL0      !WAIT DISABLE,RxINT ON 1ST OR SP COND.!
0114 FE23
0116 C838                       LDB     RL0,#%38
0118 3A86                       OUTB    WR0A,RL0        !RESET HIGHEST IUS!
011A FE21
011C 97F0                       POP     R0,@R15
011E 7B00                       IRET

0120                    END SPCOND

                        END SDLC
```

SCC In Binary Synchronous Communication

Zilog

Application Note

October 1982

Zilog's Z8030 Z-SCC Serial Communications Controller is one of a family of components that are Z-BUS™ compatible with the Z8000™ CPU. Combined with a Z8000 CPU (or other existing 8- or 16-bit CPUs with nonmultiplexed buses when using the Z8530 SCC), the Z-SCC forms an integrated data communications controller that is more cost effective and more compact than systems incorporating UARTs, baud rate generators, and phase-locked loops as separate entities.

The approach examined here implements a communications controller in a Binary Synchronous mode of operation, with a Z8002 CPU acting as controller for the Z-SCC.

One channel of the Z-SCC is used to communicate with the remote station in Half Duplex mode at 9600 bits/second. To test this application, two Z8000 Development Modules are used. Both are loaded with the same software routines for initialization and for transmitting and receiving messages. The main program of one module requests the transmit routine to send a message of the length indicated in the 'COUNT' parameter. The other system receives the incoming data stream, storing the message in its resident memory.

DATA TRANSFER MODES

The Z-SCC system interface supports the following data transfer modes:

- **Polled Mode.** The CPU periodically polls the Z-SCC status registers to determine the availability of a received character, if a character is needed for transmission, and if any errors have been detected.
- **Interrupt Mode.** The Z-SCC interrupts the CPU when certain previously defined conditions are met.
- **Block/DMA Mode.** Using the Wait/Request ($\overline{W}/\overline{REQ}$) signal, the Z-SCC introduces extra wait cycles to synchronize data transfer between a CPU or DMA controller and the Z-SCC.

The example given here uses the block mode of data transfer in its transmit and receive routines.

SYNCHRONOUS MODES

Three variations of character-oriented synchronous communications are supported by the Z-SCC: Monosync, Bisync, and External Sync (Figure 1). In Monosync mode, a single sync character is transmitted, which is then compared to an identical sync character in the receiver. When the receiver recognizes this sync character, synchronization is complete; the receiver then transfers subsequent characters into the receiver FIFO in the Z-SCC.

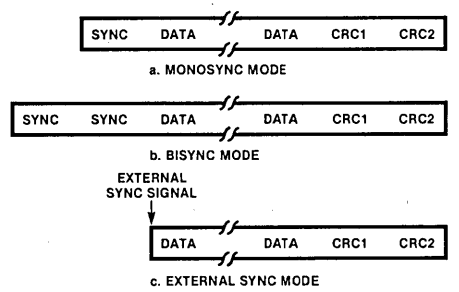


Figure 1. Synchronous Modes of Communication

Bisync mode uses a 16-bit or 12-bit sync character in the same way to obtain synchronization. External Sync mode uses an external signal to mark the beginning of the data field; i.e., an external input pin (SYNC) indicates the start of the information field.

In all synchronous modes, two Cyclic Redundancy Check (CRC) bytes can be concatenated to the message to detect data transmission errors. The CRC bytes inserted in the transmitted message are compared to the CRC bytes computed to the receiver. Any differences found are held in the receive error FIFO.

SYSTEM INTERFACE

The Z8002 Development Module consists of a Z8002 CPU, 16K words of dynamic RAM, 2K words of EPROM

Two Z8000 Development Modules containing Z-SCCs are connected as shown in Figure 3 and Figure 4. The Transmit Data pin of one is connected to the Receive Data pin of the other and vice versa. The Z8002 is used as a host CPU for loading the modules' memories with software routines.

The Z8000 CPU can address either of the two bytes contained in 16-bit words. The CPU uses an even address (16 bits) to access the most-significant byte of a word and an odd address for the least-significant byte of a word.

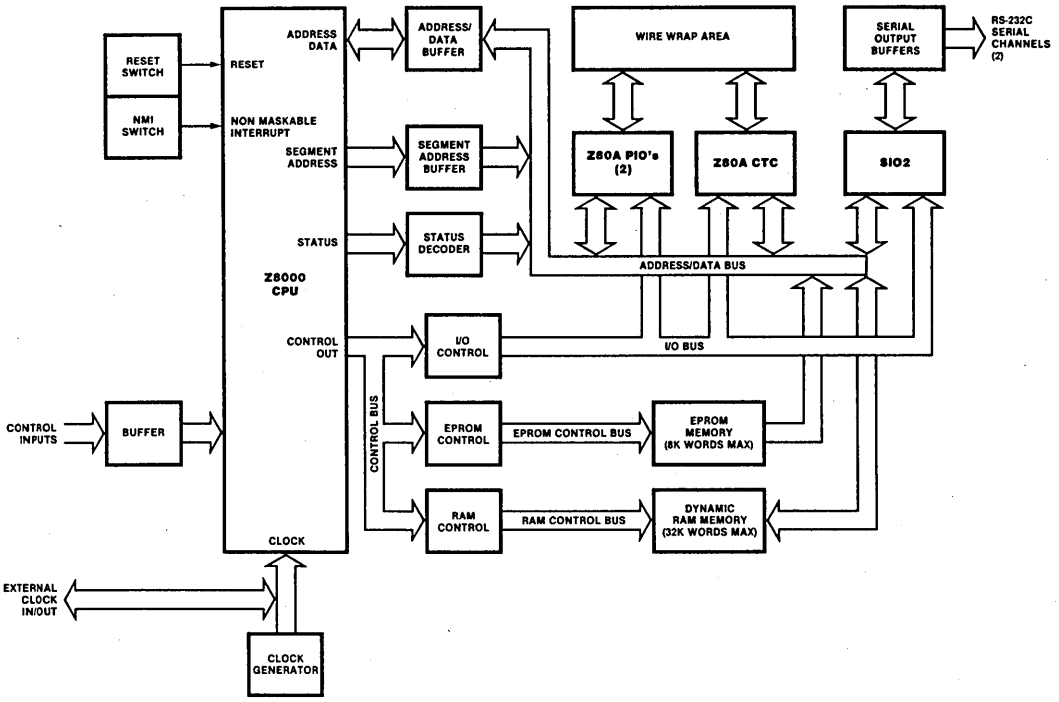


Figure 2. Block Diagram of Z8000 DM

monitor, a Z80A SIO providing dual serial ports, a Z80A CTC peripheral device providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire wrap area for prototyping. The block diagram is depicted in Figure 2. Each of the peripherals in the development module is connected in a prioritized daisy-chain configuration. The Z-SCC is included in this configuration by tying its IEI line to the IEO line of another device, thus making it one step lower in interrupt priority compared to the other device.

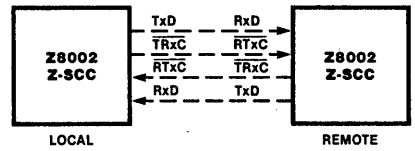


Figure 3. Block Diagram of Two Z8000 Development Modules

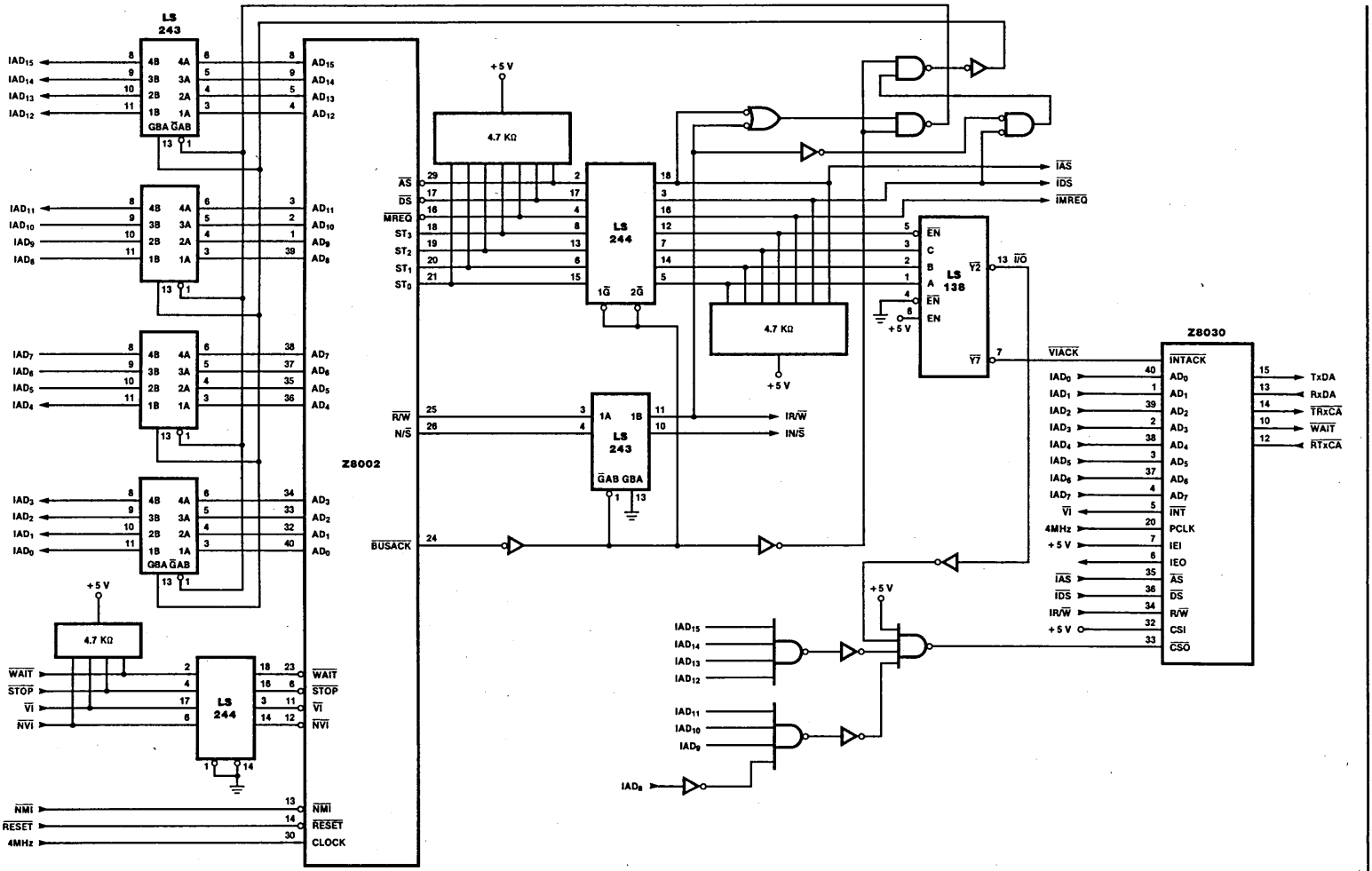


Figure 4. Z8002 with SCC

When the Z8002 CPU uses the lower half of the Address/Data bus (AD_0-AD_7 the least significant byte) for byte read and write transactions during I/O operations, these transactions are performed between the CPU and I/O ports located at odd I/O addresses. Since the Z-SCC is attached to the CPU on the lower half of the A/D bus, its registers must appear to the CPU at odd I/O addresses. To achieve this, the Z-SCC can be programmed to select its internal registers using lines AD_1-AD_5. This is done either automatically with the Force Hardware Reset command in WR9 or by sending a Select Shift Left Mode command to WR0B in channel B of the Z-SCC. For this application, the Z-SCC registers are located at I/O port address 'FExx'. The Chip Select signal ($\overline{CS0}$) is derived by decoding I/O address 'FE' hex from lines AD_8-AD_{15} of the controller. The Read/Write registers are automatically selected by the Z-SCC when internally decoding lines AD_1-AD_5 in Shift Left mode. To select the Read/Write registers automatically, the Z-SCC decodes lines AD_1-AD_5 in Shift Left mode. The register map for the Z-SCC is depicted in Table 1.

INITIALIZATION

The Z-SCC can be initialized for use in different modes by setting various bits in its Write registers. First, a hardware reset must be performed by setting bits 7 and 6 of WR9 to one; the rest of the bits are disabled by writing a logic zero.

Bisync mode is established by selecting a 16-bit sync character, Sync Mode Enable, and a X1 clock in WR4. A data rate of 9600 baud, NRZ encoding, and a data character length of eight bits are among the other options that are selected in this example (Table 2).

Note that WR9 is accessed twice, first to perform a hardware reset and again at the end of the initialization sequence to enable the interrupts. The programming sequence depicted in Table 2 establishes the necessary parameters for the receiver and the transmitter so that, when enabled, they are ready to perform communication tasks. To avoid internal race and false interrupt conditions, it is important to initialize the registers in the sequence depicted in this application note.

Table 1. Register Map

Address (hex)	Write Register	Read Register
FE01	WR0B	RR0B
FE03	WR1B	RR1B
FE05	WR2	RR2B
FE07	WR3B	RR3B
FE09	WR4B	
FE0B	WR5B	
FE0D	WR6B	
FE0F	WR7B	
FE11	B DATA	B DATA
FE13	WR9	
FE15	WR10B	RR10B
FE17	WR11B	
FE19	WR12B	RR12B
FE1B	WR13B	RR13B
FE1D	WR14B	
FE1F	WR15B	RR15B
FE21	WR0A	RR0A
FE23	WR1A	RR1A
FE25	WR2	RR2A
FE27	WR3A	RR3A
FE29	WR4A	
FE2B	WR5A	
FE2D	WR6A	
FE2F	WR7A	
FE31	A DATA	A DATA
FE33	WR9	
FE35	WR10A	RR10A
FE37	WR11A	
FE39	WR12A	RR12A
FE3B	WR13A	RR13A
FE3D	WR14A	
FE3F	WR15A	RR15A

The Z8002 CPU must be operated in System mode in order to execute privileged I/O instructions, so the Flag Control Word (FCW) should be loaded with System/Normal ($S/\overline{N}$), and the Vectored Interrupt Enable (VIE) bits set. The Program Status Area Pointer (PSAP) is loaded with address %4400 using the Load Control instruction (LDCTL). If the Z8000 Development Module is intended to be used, the PSAP need not be loaded by the programmer as the development modules monitor loads it automatically after the NMI button is pressed.

Table 2. Programming Sequence for Initialization

Register	Value (hex)	Effect
WR9	C0	Hardware reset
WR4	10	x1 clock, 16-bit sync, sync mode enable
WR10	0	NRZ, CRC preset to zero
WR6	AB	Any sync character "AB"
WR7	CD	Any sync character "CD"
WR2	20	Interrupt vector "20"
WR11	16	Tx clock from BRG output, TRxC pin = BRG out
WR12	CE	Lower byte of time constant = "CE" for 9600 baud
WR13	0	Upper byte = 0
WR14	03	BRG source bit = 1 for PCLK as input, BRG enable
WR15	00	External interrupt disable
WR5	64	Tx 8 bits/character, CRC-16
WR3	C1	Rx 8 bits/character, Rx enable (Automatic Hunt mode)
WR1	08	RxInt on 1st char & sp. cond., ext. int. disable)
WR9	09	MIE, VIS, Status Low

Since VIS and Status Low are selected in WR9, the vectors listed in Table 3 will be returned during the Interrupt Acknowledge cycle. Of the four interrupts listed, only two, Ch A Receive Character Available and Ch A Special Receive Condition, are used in the example given here.

Table 3. Interrupt Vectors

Vector (hex)	PS Address* (hex)	Interrupt
28	446E	Ch A Transmit Buffer Empty
2A	4472	Ch A External Status Change
2C	4476	Ch A Receive Char. Available
2E	447A	Ch A Special Receive Condition

* "PS Address" refers to the location in the Program Status Area where the service routine address is stored for that particular interrupt, assuming that PSAP has been set to 4400 hex.

TRANSMIT OPERATION

To transmit a block of data, the main program calls up the transmit data routine. With this routine, each message block to be transmitted is stored in memory, beginning with location 'TBUF'. The number of characters contained in each block is determined by the value assigned to the 'COUNT' parameter in the main module.

To prepare for transmission, the routine enables the transmitter and selects the Wait On Transmit function; it then enables the wait function. The Wait On Transmit function indicates to the CPU whether or not the Z-SCC is ready to accept data from the CPU. If the CPU attempts to send data to the Z-SCC when the transmit buffer is full, the Z-SCC asserts its Wait line and keeps it Low until the buffer is empty. In response, the CPU extends its I/O cycles until the Wait line goes inactive, indicating that the Z-SCC is ready to receive data.

The CRC generator is reset and the Transmit CRC bit is enabled before the first character is sent, thus including all the characters sent to the Z-SCC in the CRC calculation, until the Transmit CRC bit is disabled. CRC generation can be disabled for a particular character by resetting the TxCRC bit within the transmit routine. In this application, however, the Transmit CRC bit is not disabled, so that all characters sent to the Z-SCC are included in the CRC calculation.

The Z-SCC's transmit underrun/EOM latch must be reset sometime after the first character is transmitted by writing a Reset Tx Underrun/EOM command to WR0. When this latch is reset, the Z-SCC automatically appends the CRC characters to the end of the message in the case of an underrun condition.

Finally, a five-character delay is introduced at the end of the transmission, which allows the Z-SCC sufficient time to transmit the last data byte, two CRC characters, and two sync characters before disabling the transmitter.

RECEIVE OPERATION

Once the Z-SCC is initialized, it can be prepared to receive data. First, the receiver is enabled, placing the Z-SCC in Hunt mode and thus

setting the Sync/Hunt bit in status register RR0 to 1. In Hunt mode, the receiver is idle except that it searches the incoming data stream for a sync character match. When a match is discovered between the incoming data stream and the sync characters stored in WR6 and WR7, the receiver exits the Hunt mode, resetting the Sync/Hunt bit in status register RR0 and establishing the Receive Interrupt On First Character mode. Upon detection of the receive interrupt, the CPU generates an Interrupt Acknowledge cycle. The Z-SCC sends to the CPU vector %2C, which points to the location in the Program Status Area from which the receive interrupt service routine is accessed.

The receive data routine is called from within the receive interrupt service routine. While expecting a block of data, the Wait On Receive function is enabled. Receive data buffer RR8 is read, and the characters are stored in memory locations starting at RBUF. The Start of Text (%02) character is discarded. After the End of Transmission character (%04) is received, the two CRC bytes are read. The result of the CRC check becomes valid two characters later, at which time, RR1 is read and the CRC error bit is checked. If the bit is zero, the message received can be assumed correct; if the bit is 1, an error in the transmission is indicated.

Before leaving the interrupt service routine, Reset Highest IUS (Interrupt Under Service), Enable Interrupt on Next Recieve Character, and Enter Hunt Mode commands are issued to the Z-SCC.

If a receive overrun error is made, a special condition interrupt occurs. The Z-SCC presents the vector %2E to the CPU, and the service routine located at address %447A is executed. The Special Receive Condition register RR1 is read to determine which error occurred. Appropriate action to correct the error should be taken by the user at this point. Error Reset and Reset Highest IUS commands are given to the Z-SCC before returning to the main program so that the other lower priority interrupts can occur.

SOFTWARE

Software routines are presented in the following pages. These routines can be modified to include various versions of Bisync protocol, such as Transparent and Nontransparent modes. Encoding methods other than NRZ (e.g., NRZI, FM0, FM1) can also be used by modifying WR10.

Appendix

Software Routines

```
plzasm    1.3
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1          BISYNC MODULE
                      $LISTON $TTY
                      CONSTANT
                      WR0A     :=  %FE21              !BASE ADDRESS FOR WR0 CHANNEL A!
                      RR0A     :=  %FE21              !BASE ADDRESS FOR RR0 CHANNEL A!
                      RBUF     :=  %5400              !BUFFER AREA FOR RECEIVE CHARACTER!
                      PSAREA   :=  %4400              !START ADDRESS FOR PROGRAM STAT AREA!
                      COUNT    :=   12                !NO. OF CHAR. FOR TRANSMIT ROUTINE!
0000                  GLOBAL MAIN PROCEDURE
                      ENTRY
0000 7601                        LDA     R1,PSAREA
0002 4400
0004 7D1D                        LDCTL   PSAPOFF,R1     !LOAD PSAP!
0006 2100                        LD      R0,#%5000
0008 5000
000A 3310                        LD      R1(#%1C),R0    !FCW VALUE(%5000) AT %441C FOR VECTORED!
000C 001C                                               !INTERRUPTS!
000E 7600                        LDA     R0,REC
0010 00F4'
0012 3310                        LD      R1(#%76),R0    !EXT. STATUS SERVICE ADDR. AT %4476 IN!
0014 0076                                               !PSA!
0016 7600                        LDA     R0,SPCOND
0018 011E'
001A 3310                        LD      R1(#%7A),R0    !SP.COND.SERVICE ADDR AT %447A IN PSA!
001C 007A
001E 5F00                        CALL    INIT
0020 0034'
0022 5F00                        CALL    TRANSMIT
0024 00A6'
0026 E8FF                        JR      $
0028 02                TBUF:     BVAL    %02            !START OF TEXT!
0029 31                          BVAL    '1'            !BVAL MEANS BYTE VALUE. MESSAGE CHAR.!
002A 32                          BVAL    '2'
002B 33                          BVAL    '3'
002C 34                          BVAL    '4'
002D 35                          BVAL    '5'
002E 36                          BVAL    '6'
002F 37                          BVAL    '7'
0030 38                          BVAL    '8'
0031 39                          BVAL    '9'
0032 30                          BVAL    '0'
0033 31                          BVAL    '1'
0034                             END     MAIN
```

```
                !****************** INITIALIZATION ROUTINE FOR Z-SCC ***********************!

0034                    GLOBAL  INIT PROCEDURE
                        ENTRY
0034 2100               LD      R0,#15          !NO.OF PORTS TO WRITE TO!
0036 000F
0038 7602               LDA     R2,SCCTAB       !ADDRESS OF DATA FOR PORTS!
003A 004E'
003C 2101       ALOOP:  LD      R1,#WR0A
003E FE21
0040 0029               ADDB    RL1,@R2
0042 A920               INC     R2
0044 3A22               OUTIB   @R1,@R2,R0      !POINT TO WR0A,WR1A ETC THRO LOOP!
0046 0018
0048 8D04               TEST    R0              !END OF LOOP?!
004A EEF8               JR      NZ,ALOOP        !NO,KEEP LOOPING!
004C 9E08               RET
004E 12         SCCTAB: BVAL    2*9
004F C0                 BVAL    %C0             !WR9=HARDWARE RESET!
0050 08                 BVAL    2*4
0051 10                 BVAL    %10             !WR4=X1 CLK,16 BIT SYNC MODE!
0052 14                 BVAL    2*10
0053 00                 BVAL    0               !WR10=CRC PRESET ZERO,NRZ,16 BIT SYNC!
0054 0C                 BVAL    2*6
0055 AB                 BVAL    %AB             !WR6=ANY SYNC CHAR %AB!
0056 0E                 BVAL    2*7
0057 CD                 BVAL    %CD             !WR7=ANY SYNC CHARR %CD!
0058 04                 BVAL    2*2
0059 20                 BVAL    %20             !WR2=INT VECTOR %20!
005A 16                 BVAL    2*11
005B 16                 BVAL    %16             !WR11=TxCLOCK & TRxC OUT=BRG OUT!
005C 18                 BVAL    2*12
005D CE                 BVAL    %CE             !WR12= LOWER TC=%CE!
005E 1A                 BVAL    2*13
005F 00                 BVAL    0               !WR13= UPPER TC=0!
0060 1C                 BVAL    2*14
0061 03                 BVAL    %03             !WR14=BRG ON, ITS SRC=PCLK!
0062 1E                 BVAL    2*15
0063 00                 BVAL    %00             !WR15=NO EXT INT EN.!
0064 0A                 BVAL    2*5
0065 64                 BVAL    %64             !WR5= TX 8 BITS/CHAR, CRC-16!
0066 06                 BVAL    2*3
0067 C1                 BVAL    %C1             !WR3=RX 8 BITS/CHAR, REC ENABLE!
0068 02                 BVAL    2*1
0069 08                 BVAL    %08             !WR1=RxINT ON 1ST OR SP COND!
                                                !    EXT INT DISABLE!
006A 12                 BVAL    2*9
006B 09                 BVAL    %09             !WR9= MIE,VIS,STATUS LOW!
006C            END INIT


                !****************** RECEIVE ROUTINE ************************************!

                !                   RECEIVE A BLOCK OF MESSAGE                          !
                !                   THE LAST CHARACTER SHOULD BE EOT(%04)               !

006C                    GLOBAL  RECEIVE PROCEDURE
                        ENTRY
006C C828               LDB     RL0,#%28        !WAIT ON RECV.!
006E 3A86               OUTB    WR0A+2,RL0
0070 FE23
0072 6008               LDB     RL0,%A8
0074 00A8
0076 3A86               OUTB    WR0A+2,RL0      !ENABLE WAIT 1ST CHAR,SP.COND. INT!
0078 FE23
007A 2101               LD      R1,#RR0A+16
007C FE31
007E 3C18               INB     RL0,@R1         !READ STX CHARACTER!
0080 C8C9               LDB     RL0,#%C9
0082 3A86               OUTB    WR0A+6,RL0      !Rx CRC ENABLE!
0084 FE27
0086 2103               LD      R3,#RBUF
0088 5400
008A 3C18       READ:   INB     RL0,@R1         !READ MESSAGE!
008C 2E38               LDB     @R3,RL0         !STORE CHARACTER IN RBUF!
008E AB30               DEC     R3,#1
0090 0A08               CPB     RL0,#%04        !IS IT END OF TRANSMISSION ?!
0092 0404
0094 EEFA               JR      NZ,READ
0096 3C18               INB     RL0,@R1         !READ PAD1!
0098 3C18               INB     RL0,@R1         !READ PAD2!
009A 3A84               INB     RL0,RR0A+2      !READ CRC STATUS!
009C FE23
                !       PROCESS CRC ERROR IF ANY, AND GIVE ERROR RESET COMMAND IN WR0A !
009E C800               LDB     RL0,#0
00A0 3A86               OUTB    WR0A+6,RL0      !DISABLE RECEIVER!
00A2 FE27
00A4 9E08               RET
00A6            END RECEIVE
```

```
                        !*****************  TRANSMIT ROUTINE ************************************!
                        !                    SEND A BLOCK OF DATA CHARACTERS                    !
                        !                    THE BLOCK STARTS AT LOCATION TBUF                  !

00A6                            GLOBAL  TRANSMIT PROCEDURE
                                ENTRY
00A6 2102                               LD      R2,#TBUF        !PTR TO START OF BUFFER!
00A8 0028'
00AA C86C                               LDB     RL0,#%6C
00AC 3A86                               OUTB    WR0A+10,RL0     !ENABLE TRANSMITTER!
00AE FE2B
00B0 C800                               LDB     RL0,#%00        !WAIT ON TRANSMIT!
00B2 3A86                               OUTB    WR0A+2,RL0
00B4 FE23
00B6 C888                               LDB     RL0,#%88
00B8 3A86                               OUTB    WR0A+2,RL0      !WAIT ENABLE,INT ON 1ST & SP COND!
00BA FE23
00BC C880                               LDB     RL0,#%80
00BE 3A86                               OUTB    WR0A,RL0        !RESET TxCRC GENERATOR!
00C0 FE21
00C2 2101                               LD      R1,#WR0A+16     !WR8A SELECTED!
00C4 FE31
00C6 C86D                               LDB     RL0,#%6D
00C8 3A86                               OUTB    WR0A+10,RL0     !Tx CRC ENABLE!
00CA FE2B
00CC 2100                               LD      R0,#1
00CE 0001
00D0 3A22                               OTIRB   @R1,@R2,R0      !SEND START OF TEXT!
00D2 0010
00D4 C8C0                               LDB     RL0,#%C0
00D6 3A86                               OUTB    WR0A,RL0        !RESET TxUND/EOM LATCH!
00D8 FE21
00DA 2100                               LD      R0,#COUNT-1
00DC 000B
00DE 3A22                               OTIRB   @R1,@R2,R0      !SEND MESSAGE!
00E0 0010
00E2 C804                               LDB     RL0,#%04
00E4 3E18                               OUTB    @R1,RL0         !SEND END OF TRANSMISSION CHARACTER!
00E6 2100                               LD      R0,#1670        !CREATE DELAY BEFORE DISABLING!
00E8 0686
00EA F081                       DEL:    DJNZ    R0,DEL
00EC C800                               LDB     RL0,#0
00EE 3A86                               OUTB    WR0A+10,RL0     !DISABLE TRANSMITTER!
00F0 FE2B
00F2 9E08                               RET
00F4                            END TRANSMIT


                        !************** RECEIVE INT. SERVICE ROUTINE *************************!

00F4                              GLOBAL REC PROCEDURE
                                  ENTRY
00F4 93F0                               PUSH    @R15,R0
00F6 3A84                               INB     RL0,RR0A        !READ STATUS FROM RR0A!
00F8 FE21
00FA A684                               BITB    RL0,#4          !TEST IF SYNC HUNT RESET!
00FC EE02                               JR      NZ,RESET        !YES CALL RECEIVE ROUTINE!
00FE 5F00                               CALL    RECEIVE
0100 006C'
0102 C808                       RESET:  LDB     RL0,#%08
0104 3A86                               OUTB    WR0A+2,RL0      !WAIT DISABLE!
0106 FE23
0108 C8D1                               LDB     RL0,#%D1
010A 3A86                               OUTB    WR0A+6,RL0      !ENTER HUNT MODE!
010C FE27
010E C820                               LDB     RL0,#%20
0110 3A86                               OUTB    WR0A,RL0        !ENABLE INT ON NEXT CHAR!
0112 FE21
0114 C838                               LDB     RL0,#%38
0116 3A86                               OUTB    WR0A,RL0        !RESET HIGHEST IUS!
0118 FE21
011A 97F0                               POP     R0,@R15
011C 7B00                               IRET
011E                            END REC
```

```
                !************* SPECIAL CONDITION INTERRUPT SERVICE ROUTINE ***************!

011E                    GLOBAL SPCOND PROCEDURE
                        ENTRY

011E 93F0                       PUSH    @R15,R0
0120 3A84                       INB     RL0,RR0A+2      !READ ERRORS!
0122 FE23
                                !PROCESS ERRORS!
0124 C830                       LDB     RL0,#%30
0126 3A86                       OUTB    WR0A,RL0        !ERROR RESET!
0128 FE21
012A C808                       LDB     RL0,#%08
012C 3A86                       OUTB    WR0A+2,RL0      !WAIT DISABLE,RxINT ON 1ST OR SP COND.!
012E FE23
0130 C8D1                       LDB     RL0,#%D1
0132 3A86                       OUTB    WR0A+6,RL0      !HUNT MODE,REC. ENABLE!
0134 FE27
0136 C838                       LDB     RL0,#%38
0138 3A86                       OUTB    WR0A,RL0        !RESET HIGHEST IUS!
013A FE21
013C 97F0                       POP     R0,@R15
013E 7B00                       IRET

0140                    END SPCOND

                        END BISYNC

   0 errors
Assembly complete
```

October 1988

Military Products

Zilog offers a high reliability version of many of our Z8000 logic circuits. Zilog military microcircuits are fabricated, assembled, and tested in accordance with the latest requirements of MIL-STD-883 using the highest quality and reliability standards.

Zilog's multi-million dollar fabrication facility in Nampa, Idaho incorporates the highest quality and latest technological equipment available for semiconductor wafer processing. Our Nampa wafer fabrication line is JAN Mil-M-38510 certified and meets stringent government requirements for process control, facility cleanliness, documentation, and equipment calibration.

The assembly, end-of-line processing, and final test/ship areas have also gained JAN certification.

Zilog has implemented the latest test procedures to ensure maximum performance and reliability in addition to full compliance with the military specifications.

Zilog has extensive operator training programs, carefully monitored internal process specification controls, strict equipment maintenance procedures, ongoing research and development, and continuous data management to ensure that Zilog military products represent the industry standard for excellence.

MILITARY SOFTWARE

Ada

Ada, a high-level programming language developed and specified by the U.S. Department of Defense, is designed for use in embedded applications.

Zilog currently has a validated Ada Compiler available which implements the ANSI/Mil-STD-1815A requirements. This Compiler, developed by Meridian Software and called Advantage, features high compile speed and efficient code generation.

The compiler generates Z8001 segmented memory object code. Full implementation and validation was approved in July 1987.

JAN MIL-M-38510

Zilog Military Products has a clear and ongoing commitment to the qualification and production of high-reliability JAN Mil-M-38510 QPL components.

Zilog's strong JAN commitment is exemplified by the qualification of ourZ8002 Military microprocessor.

Zilog Military Products has proven its ongoing commitment to the military community and will continue to dedicate resources toward that goal.

MIL-STD-883 MILITARY PROCESSED PRODUCT

Mil-Std-883 establishes uniform methods and procedures for testing microelectronic devices to insure the electrical, mechanical, and environmental integrity and reliability that is required for military applications.

Mil-Std-883 Class B, Revision C is the industry standard product assurance level for military ground and aircraft application. All Zilog military products fully meet Revision 'C' of this standard.

The total reliability of a system depends upon tests that are designed to stress specific quality and reliability concerns that affect the microelectronic products.

The following tables detail the 100% screening and electrical tests, sample electrical tests, and Qualification/Quality Conformance testing required.

Z8000 MILITARY PRODUCTS

The following table contains a listing of the Z8000 Military Products. Contact your local Zilog sales office for the Product Specification of any of these devices.

Device	Description	883		DESC		JAN	
		Speed	Package	Speed	Package	Speed	Package
Z08001	Segmented CPU, 16-Bit	4, 6, 10	C, L[1]	4, 6, 10	C, L		
Z08002	Nonsegmented CPU, 16-Bit	4, 6, 10	C, L	4, 10	C, L	4, 6	C
Z08030	Z-SCC, Z-Bus, NMOS	4, 6	C, L	4, 6	C, L		
Z80C30	Z-SCC, Z-Bus, CMOS	6, 8	C				
Z08530	SCC, Universal Bus, NMOS	4, 6	C, L	4, 6	C, L		
Z85C30	SCC, Universal Bus, CMOS	6, 8	C				
Z08036	Z-CIO, Z-Bus	4, 6	C, L	4, 6	C, L		
Z08536	CIO, Universal Bus	4, 6	C, L	4, 6	C		
Z08038	Z-FIO, Z-Bus	4, 6	C, L[2]				
Z08581	Z8000 Clock Chip	6, 10	C, L	6, 10	C, L		
Z80000[3]	CPU, 32-Bit	8	G				

C = Ceramic, L = LCC, G = Pin Grid Array

1) 6 and 10 MHz only
2) 6 MHz only
3) Future product, contact your local sales office for current availability.

October 1988

Z8000 Development Support

Zilog's policy is to provide easy to use, basic support tools to allow customers to develop hardware designs and write software application programs quickly and efficiently.

Zilog encourages and supports third party companies to provide a comprehensive set of support tools to help Zilog customers. A list of known companies that provide support for the Zilog Z8000 and Z80,000 microprocessors follows.

ZILOG DEVELOPMENT SUPPORT SHORTFORM DIRECTORY

	Z8	Z8	Z80	Z180	Z280	Z8000	Z80,000
ICE	Microtek Orion Sophia	Microtek Sophia	Ap.Micro HP Kontron Microtek Orion Sophia Tektronix	HP Microtek Orion Sophia	HP* Microtek* Teksel* Softaid Orion*	HP Kontron Orion Applied Micro	
Eval Board	JK Eng	Zilog	Zilog*	Zilog*	SBS	SBS*	
CrossAssembler	Zilog 2500 AD Microtek	Zilog 2500 AD Microtek	Zilog 2500 AD Unidot Microtek	2500 AD Microtek Am.Automa'n Uniware	Zilog 2500 AD	Zilog HP Tektronix 2500 AD Unidot	Zilog 2500 AD
Pascal			HP Kontron Microtek	Microtek		HP Kontron Meridian	
RTOS			Ready VRTX		JMI*	Ready VRTX	JMI*
Forth		Inner Access	Inner Access			Inner Access	
ADA Compiler						Meridian TLD*	
Jovial Compiler						PSS PEA ACT TLD*	
Fortran Compiler			Unidot Supersoft			Unidot	

* Planned

Disclaimer

Zilog's policy is to be the catalyst by which independently supported software tools are provided for customer use in design with Zilog products. Zilog in no way warrants any products not manufactured by Zilog in the above list. Zilog support products are provided to assist in designing with Zilog integrated circuits, and are only intended to assist users and software support vendors during product introduction. Zilog products are available under an unsupported license agreement only. Please contact Zilog marketing if the above information is incorrect or incomplete.

Zilog initially developed efficient cross-assemblers including utilities such as linkers and loaders and board level debug tools to allow efficient programming with new microprocessors. Zilog also develops 'C' compilers, 'ADA' compilers and operating system support to allow fast implementation of first designs.

Designers wishing to develop their own tools from the Zilog base of tools can apply for copies of Zilog Software that is available under end use license for a minor charge.

16 BIT SUPPORT

The Z8001 and Z8002 are fully supported with cross-assemblers, C and Ada compilers, and the VRTX operating system to allow fast system design.

Hardware support tools with fully real-time in-circuit emulation (ICE) is available from low cost PC toolsets to HP 64000 systems depending on your development needs.

32 BIT SUPPORT

The Z320 comprehensive third party support program is in development to allow easy compilation of programs and effective hardware debug.

The Z80320 has a superset of the Z8000 instruction set and can run the Z8000 binary op code already generated for 16-bit Z8000 applications. The Z8000 support therefore allows the ability to run "compact mode" in the Z320 without use of cache or the Z320 MMU. Efficient "C" compilers and assembler linkers are available now for the Z8000 which can be recompiled when the Z80320 "C" compiler becomes available.

Support planned for the first quarter of 1988 includes a: "C" compiler, "ADA" compiler for embedded control, IBM PC/AT single-board computer and evaluation board ATZ80K, debug monitor for the ATZ80K, and a realtime operating system.

Z328 In-Circuit Emulation Chip

The Z80328 In-Circuit Emulator chip is a support chip that gives users the hardware control of the internal instruction execution microcycles performed by the pipelined structure.

Access to breakpoint request and breakpoint acknowledge signals allows real-time execution with external control of the breakpoint trap feature. Continual exertion of a breakpoint request will result in single-step operation where access to read or modify CPU registers and memory in the CPU and the target offers the ability to trace.

During single-step operation, the internal CPU cache is disabled allowing control of operand source and destination addresses.

The Z328 is an easy solution to system debugging that is not intended to replace full in-circuit emulation tools. The Z328 is available in the 84-pin ceramic grid-array package. The Z328 price is $150.00 in 1-to-10 piece volume orders.

ATZ80-K, IBM PC-AT Z320 Development Board

The ATZ80K board design is a stand-alone Z80320 microprocessor board with a PC-AT bus interface. the board is designed to run from 10 to 25 MHz clock speeds and interface with compatible IBM PC-AT systems as an add-in co-processor board, or as a passive AT bus master or slave.

A set of tools, including an on-board debugger and an independently available "C" compiler and "ADA" compiler will allow this board to provide an effective software development environment in 1Q88. Zilog will be supplying this board to customers who would like to evaluate Z80320 performance. Manufacturers will be able to purchase OEM quantities of this board from the supplier of this product, Single Board Solutions, Inc., in Sunnyvale, California.

SBS is but one of the Z8000 multibus board manufacturers who will be upgrading to use Z80320 microprocessors for cost-effective small system performance upgrades.

Features

The ATZ80K board contains the following on-board features.

1 MB of X9 parity dynamic RAM (expandable to 64 Mbyte) with zero wait states.

128 KB of EPROM or EEPROM with zero wait states (expandable to 2 Mbyte).

2 serial RS232 ports by Z80C30 CMOS SCC.

2 8-bit ports, one in an SCSI configuration driven by a Z8036 CIO.

1 SBX connector for add-on SBX boards.

1 Z-BUS connector.

Configurable clock speed from 10 to 25 MHz.

1 Z80320 10 MHz microprocessor.

Burst mode memory access with all memory.

8 vectored interrupts, 2 non-vectored interrupts.

Applications

The configuration of the ATZ80K board allows the user to build systems and use the development environment also as a high-performance host. Two primary applications are targeted.

Development Environments

An on-board memory debugger, the "DM80K" allows users to access Z80320 registers and perform operations such as Load, Input, Output, Display, Alter, Fill, Go, Send, and Mapping control commands.

Development tools resident on the IBM PC-AT host, such as Zilog's assembler or "C" compiler, will allow programs to be generated, executed in the host, then down-loaded into the Z80320 board for target execution and debug.

Block move operations allows host memory to target memory transfers at over 1 MB/second for effective memory-to-memory communication.

This configuration will lend itself well to easy benchmark comparison by users running their individual code on the Zilog Z80320, the PC-AT processor, the intel 80286 or 80386.

Passive "AT" Bus Master or Slave Environment

The control of all functions on the "AT" bus are provided by this single-board computer. All refresh and control signals are provided by the "ATZ80K" board.

The SBX connector allows additions of many features, including disk controllers, although software drivers will be needed to interface either to the "AT" bus or "SBX" resident controllers.

Arbitration for bus control is available through the Z8036 CIO that does interrupt vector generation for the Z80320.

Either the "AT" bus or Z-BUS extension can be used for system addition of memory or I/O.

Z8000 SUPPORT

Hardware

Applied Micro (206) 882-2000	ES-1800, 10MHz ICE, RS232 Compatible with others.
Boston Office Systems (617) 894-7800	Universal Microprocessor Dev Sys. VAX/VMS.
Hewlett-Packard (415) 857-1501 (800) 752-0900	HP9000 and HP64000. Host to HP64000 and VAX
Kontron (415) 965-7020 (800) 227-8834	KSE Series. Host to any. Have interface Card , 6 MHz.
Orion (415) 361-8883	UDL (Universal Dev Lab) Host to any PC.
Single Board Solution (408) 253-0250	Multibus Board/Development Module.
Tektronix (800) 835-9433	TEK 8500, 10 MHz, Host to MS DOS, PS DOS UNIX and VMS

Cross Assemblers

2500 AD (719) 395-8683	Crossassembler that operates on CP/M, MS DOS, UNIX PC DOS and CP/M86
Boston Office Systems (617) 894-7800	Crossassembler that operates on PDP, and VAX.
Hewlett Packard (415) 857-1501	Crossassembler that operates on HP9000 HP46000, VAX (UNIX).
Kontron (415) 965-7020	Crossassembler that operates on CP/M Also a card for IBM PC, 6 MHz (KPI Card).
Microtec (408) 733-2919	Crossassembler that operates on MS DOS,
Tektronix (800) 835-9433	Crossassembler that operates on MS DOS PSDOS, UNIX, VMS, and Tektronix Systems.
Unidot (303) 526-9263	Crossassembler that operates on MS DOS MS UNIX, XENIX and VENIX.

Western Wares (303) 327-4898	Cross assembler that operates on CP/M Intel OS, and MS DOS

Cross Compilers

Hewlett-Packard (415) 857-1501 (800) 4HP-DATA	C and Pascal cross compiler that Operates on HP9000, HP46000, and VAX.
Inner Access (415) 591-8295	Forth Cross Compiler that CP/M, and PDP-11
Kontron (415) 965-7020	Pascal Cross Compiler that operates on KIDS, and CP/M 80.
Meridian (714) 380-9800	C, Pascal Cross Compiler and Ada that operate UNIX, MS DOS and VMS.
MIPS (313) 661-5000	APL interpreter that operates on UNIX
Tektronix (800) 835-9433	Pascal Cross Compiler that operates on CP/M, CP/M86, MS DOS and PC DOS.
Unidot (303) 526-9263	C Cross Compiler that operates under UNIX, XENIX, and VENIX.
Jovial	C Compiler, Public Domain Source available for the Z8002, contact Zilog.

Real Time Operating Systems

Ready Systems (800) 228-1249	VRTX (Z8002)

Z80,000 Hardware and Software Support Products

Single Board Solution (408) 253-0250	PC AT/SBX compatible board ($1995.00) with Monitor/Debugger
Zilog	C compiler, operates on PC DOS, VAX/VMS Systems
Zilog	Cross Assembler/Linker that runs on VAX/VMS and PC DOS systems.

NOTE: Z8000 software is also compatible with the Z80,000.

August 1988

Zilog Quality and Reliability

ZILOG'S QUALITY AND RELIABILITY PROGRAM

Introduction

The Zilog Corporation is fortunate to have an excellent reputation for quality and reliability in its products. We recognize that the expectations of our customers are continually rising, and only the best in performance is acceptable.

Zilog's Quality and Reliability Program is based on careful study of the principles laid down by such pioneers as W.E. Seming and J.M. Jurna and, perhaps even more important, observation of the practical implementation of those principles in Japanese, European and American manufacturing facilities.

The Zilog program begins with employee involvement. Whether the judgement of our performance is based on perfection in incoming inspection, trouble-free service in the field or timely and accurate customer service, we recognize that our employees ultimately control these factors. Hence, our Quality Program is broadly shared throughout the organization.

Harmony Between Design and Process

High product quality and reliability in VLSI products is possible only if there is structural harmony between product design and the manufacturing process. Using a technique which we call Process Templating, the technology file is periodically updated to assure that product design parameters fall within statistical control limits with which the process is actually operated.

Process Template

In simple terms, the Process Template is the pattern displayed each day by the process evaluation parameters which are automatically recorded from the test patterns on wafers. This information is fed back into the design software in such a way that the product design bears a key and lock relationship with the process.

Training

Product design and processing are people-dependent. Zilog emphasizes the fundamentals involved in design for quality and reliability. Customer Service, which is another important aspect of Zilog's quality performance as a vendor, also depends upon our people clearly understanding their jobs and our obligations to our customers. This too is part of the curriculum which is administered by Zilog.

Order Acknowledgement Policy

Reliability and quality warranties can be met only if Zilog and the customer are in agreement on the product and delivery specification. Zilog makes an extra effort in this regard by providing to the customer a series of documents which very clearly state what Zilog understands the specifications to be.

Test Guardbanding

In order to assure that every Zilog product performs to full customer expectations, Zilog uses a waterfall methodology in its testing. Each step in the test process is guardbanded within the subsequent steps. This technique of multiple guardbanding assures that circuits which may be marginal to the customers' expectations are eliminated in the manufacturing process long before they get to the shipping container.

Probe at Temperature

Semiconductor devices tend to exhibit their most limited performance at the highest operating temperature. Therefore, it is Zilog's policy that all chips are tested-at-room temperature the very first time they are electrically screened at the wafer probe station. The circuits are tested again at their upper operating temperature limit in the 100% final test operation. Every new Zilog product is evaluated over extremes of operating temperature, supply voltage and clock frequency, prior to release to production. This information permits the proper guardbanding of the test program waterfall and identification of any marginal "corners" in design tolerances.

Product Characterization

A product characterization report is available to Zilog's customers which summarizes the more important tolerances that are identified in the process of this exhaustive product design evaluation. In addition to characterization, every new Zilog product design is fully qualified by a comprehensive series of life, electrical and environmental tests before release to production.

Product Qualification
Again, A qualification report is available to our customers which summarizes certain key life and environmental data taken in the course of these evaluation. Wherever possible, industry standard environmental and life tests are employed.

Product Characterization
Before it is released to production, every new Zilog process is thoroughly characterized by an exhaustive series of pilot production runs and tests which identify the statistical electrical and mechanical limits which that particular process is capable. This documentation is maintained as the historical record for that particular regime. Process re-characterization is done any time there is a major process or manufacturing site change, and added to the characterization history. The daily test site evaluation work recorded in the process template noted earlier, demonstrates that the process remains in specification during time of formal characterization.

Process Qualification
Just as Zilog measures the robustness and reliability of its products by a product qualification process, Zilog also qualifies every process prior to production by an exhaustive stress sequence performed on representative products. Once process qualification is complete, a process re-qualification is performed any time there is a major process change, or the process template statistical quality limits are significantly exceeded or adjusted.

PPM Measurement Direct
It is frequently said that if you want to improve something you need to put a measurement on it. Therefore, Zilog measures its outgoing quality "parts per million" by the maintenance of careful records on the statistical sampling of production lots prepared for shipment. This information is then translated by our statisticians to a statement of our parts per million (or someday parts per billion) outgoing quality performance.

PPM Measurement Indirect
It is one thing for Zilog to do a good job in outgoing product quality and it is another for a customer to agree. Therefore, Zilog asks certain key customers to provide us with their incoming inspection data which helps us calibrate our own outgoing performance in terms of the actual results in the field.

FIT Measurement Direct
Just as Zilog records its outgoing quality in terms of parts per million, it also measures its outgoing product reliability in terms of "fits" or failures-per-billion-device-hours, using the results of daily operating life test measurements on the circuits, performed in accordance with the production process.

FIT Measurement Indirect
Zilog calibrates its own data on reliability or failure rate using information received from certain customers who share their board assembly and burn-in data with Zilog. It is also frequently said that "The customer is always right." If the customer has an application, quality or reliability problem while using a Zilog product, whether it is Zilog-caused or not, we believe that we have a responsibility to resolve it.

Field Quality Engineer
Therefore, Zilog maintains a force of skilled Applications Engineers who double as field quality engineers and are available on immediate call to consult at the customers' locations on any problems he may be experiencing with Zilog product performance.

Document Control
Skilled quality professionals maintain careful and up-to-date specifications on all aspects of Zilog's products and processes in an elaborate document control system. Specification changes and updates are electronically transmitted to the factory floor in order to assure that processing operations are being performed to the most current specifications.

Failure Analysis
Zilog believes that a customer problem is a Zilog problem. Accordingly, Failure Analysis facilities staffed by experienced professionals exist at each Zilog site to provide rapid evaluation of in-process and infield rejects and failures to determine the casualty and provide corrective action through a feedback loop into the Production, Design or Applications process.

We think you will agree that the Zilog Quality and Reliability program is comprehensive and well-disciplined. However, the program can be fully effective only if Zilog employees believe in it. Zilog shares more data with the customer than is frequently the case with the average components vendor because we, as an employee group, are very proud of our reputation in this regard. You satisfaction is our pleasure and your displeasure is our concern. We look forward to your feedback.

LITERATURE GUIDE

General Literature	Part No.	Unit Cost
Component Short Form Catalog	00-5472-03	N/C
Reliability Handbook	00-2475-01	N/C
Corp Profile	00-3124-00	N/C
Z8/Super8 Microcomputer Family		
Z8 Design Handbook	03-8275-02	12.50
Z8 PLZ Assembler User's Guide	03-3048-04	12.50
Z8 Programmer's Pocket Guide	03-9000-02	3.50
Z8 Basic/Debug Software Manual	03-3149-02	12.50
Z8600 Z8 MCU Product Spec.	00-2430-01	N/C
Z8601/03/11/13 Z8 MCU Product Spec.	00-2440-01	N/C
Z86C08 Product Spec.	00-2457-01	N/C
Z86C10 CMOS Z8 MCU Product Spec.	00-2466-01	N/C
Z86C21/Z86E21 CMOS MCU Prod. Spec.	00-2464-01	N/C
Z8681 Z8 MCU Product Spec.	00-2194-05	N/C
Z8691 Z8 ROMless Product Spec.	00-2455-01	N/C
Super8 MCU ROMless Product Spec.	00-2426-02	N/C
Super8 Article Reprint	00-5470-01	.50
Univ. Obj. File Utilities User's Manual	00-8236-03	
ASM 8 Cross Assembler User Guide	00-8267-03	
Z80/Z280 Microprocessor Family		
Z80 CPU Technical Manual	03-0029-02	15.00
Z80 CPU Programmer's Reference Gde.	03-0012-03	7.00
Z80 DMA Technical Manual	00-2013-A0	6.00
Z80 PIO Technical Manual	03-0008-01	9.00
Z80 CTC Technical Manual	03-0036-02	5.00
Z80 SIO Technical Manual	03-3033-01	7.50
Z280 MPU Preliminary Product Spec.	00-2259-05	N/C
Z280 Technical Manual	03-8224-02	15.00
Z180/Z64180 Technical Manual	03-8276-01	12.00
Z180/Z64180 CMOS MPU Product Spec.	00-2429-01	N/C
Z84C00 CMOS Z80 CPU Prelim. Prod. Spec.	00-2360-03	N/C
Z84C10 CMOS Z80 DMA Product Spec.	00-2408-03	N/C
Z84C20 CMOS Z80 PIO Product Spec.	00-2362-03	N/C
Z84C30 CMOS Z80 CTC Product Spec.	00-2363-03	N/C
Z84C40 CMOS Z80 SIO Product Spec.	00-2421-02	N/C
Z84C80/81 Product Spec.	00-2474-01	N/C
Z84C90 CMOS Z80 KIO Pre. Prod. Spec.	00-2469-01	N/C
Interfacing the Z80 CPUs to the Z8500 Peripherals Application Note	00-2296-01	2.00
Z80180 Z180 MPU	03-8276-01	
Z84C01 Z80 CPU	00-2479-01	
Z8000 Microprocessor Family		
Z8000 CPU Technical Manual	00-2010-06	12.00
Z8001/Z8002 CPU Product Spec.	00-2045-05	N/C
Z160 CPU Product Spec.	00-2045-05	N/C
Z8000 Programmer's Pocket Guide	03-0122-03	7.00
Z8000 PLZ Assembly Language Prog. Mnl.	03-3055-03	20.00
Z8010 MMU Technical Manual	00-2015-A0	4.00
Z8010 MMU Product Spec.	00-2046-03	N/C

Peripherals	Part No.	Unit Cost
Z765A FDC Product Spec.	00-2357-02	N/C
Z8030/Z8530 SCC Technical Manual	00-2057-04	6.00
Z8030 Z-FIO FIFO Technical Manual	00-2051-01	8.50
Z8036/Z8536 CIO Technical Manual	00-2091-02	8.50
Z8581 Clock Generator Product Spec.	00-2315-03	N/C
Interfacing the Z-BUS Peripherals Art. Rpt.	00-5396-01	.50
Initializing the CIO Application Note	00-2256-01	1.00
Using Z8581 Clock Stretches in Z80 CPU Applications App. Note	00-2807-01	1.00
Design Considerations Using Quartz Crystals Application Note	00-2802-01	.50
Interfacing the Z8500 Periph. to the 68000	00-2267-01	1.00
Interfacing the Z-BUS Peripherals to 8086/8088/V20/V30 Application Note	00-2255-01	1.00
Z5380 CMOS SCSI Product Spec.	00-2477-01	N/C
Z8536 CIO Product Spec.	00-2021-02	N/C
Z7220A HPGD Product Spec.	00-2407-02	N/C
Z8016 Z-DTC Product Spec.	00-2129-01	N/C
Z8068 Z-DCP Product Spec.	00-2422-01	N/C
Z8060/8560 FIFO Product Spec.	00-2450-01	N/C
Z8038/8538 FIO FIFO Product Spec.	00-2451-01	N/C
Z8516 DMA (DTC) Product Spec.	00-2449-01	N/C
Z8030/8530 SCC Product Spec.	00-2439-02	N/C
Z80C30/Z85C30 CMOS SCC Product Spec.	00-2442-03	N/C
Z80.000 Microprocessor Family		
Z80,000 CPU Product Spec.	00-2071-06	N/C
Z80,000 Technical Manual	03-8225-01	17.50
Z320 CPU Product Spec.	00-2470-01	N/C
Memory Management w/Z80,000 App. Note	00-2324-01	1.00
Development Systems		
Ada Programming Language Manual	03-8220-01	10.00
Components Military Literature		
Z8681 ROMless Military Spec.	00-2392-01	N/C
Z8001/2 CPU Military Spec.	00-2342-02	N/C
Z8581 CGC Military Spec.	00-2346-01	N/C
Z8030 Z-SCC Military Spec.	00-2388-01	N/C
Z8530 SCC Military Spec.	00-2397-01	N/C
Z8036 Z-CIO Military Spec.	00-2389-01	N/C
Z8038/8538 FIO FIFO Mil. Spec.	00-2463-01	N/C
Z8536 CIO Military Spec.	00-2396-01	N/C
Z8400 Z80 CPU Military Spec.	00-2351-02	N/C
Z84C00 Military Spec.	00-2441-02	N/C
Z8420 PIO Military Spec.	00-2384-01	N/C
Z8430 CTC Military Spec.	00-2385-01	N/C
Z8440/1/2/4 Military Spec.	00-2386-01	N/C
Z80C30/85C30 Military Product Spec.	00-2478-01	N/C

ORDERING INFORMATION:

Z5380, 1.5 MB/sec

40-pin DIP

Z0538010PSC

44-pin Plastic Leaded Chip Carrier

Z0538010VSC

40-Pin DIP

6 MHz	8 MHz
Z7220A06CSE	Z7220A08CSE

44-Pin PLCC

6 MHz	8 MHz
Z7220A06VSC	Z7220A08VSC

Z0765A FDC, 8.0 MHz

40-pin Plastic Dip	44-pin Plastic Leaded Chip Carrier
Z0765A08PSC	Z0765A08VSC

Z8001 Segmented CPU, 6.0 MHz
48-pin DIP
Z0800106PSC
Z0800106CSE
Z0800106PEC
Z0800106CEE

Z8001 Segmented CPU, 10.0 MHz
48-pin DIP
Z0800110PSC
Z0800110CSE
Z0800110PEC
Z0800110CEE

Z8002 Nonsegmented CPU, 6.0 MHz
40-pin DIP
Z0800206PSC
Z0800206DSE
Z0800206PEC
Z0800206DEE

44-pin PCC
Z0800206VSC

Z8002 Nonsegmented CPU, 10.0 MHz
40-pin DIP
Z0800210PSC
Z0800210DSE
Z0800210PEC
Z0800210DEE

44-pin PCC
Z0800210VSC

Z8010 MMU, 6.0 MHz
48-pin DIP
Z0801006PSC
Z0801006CSE

Z8010 MMU, 10.0 MHz
48-pin DIP
Z0801010PSC
Z0801010CSE

Z8016 Z-DTC, 6.0 MHz
48-pin DIP
Z0801606PSC
Z0801606CSE

Z8038 Z-FIO

40-pin DIP	44-pin PCC
6 MHz	
Z0803806PSC	Z0803806VSC
Z0803806DSC	

Z8538 FIO

40-pin DIP	44-pin PCC
6 MHz	
Z0853806PSC	Z0853806VSC
Z0853806DSE	

Z8060 FIFO, 1 Mbyte/sec	Z8560 FIFO, 1 Mbyte/sec
28-pin DIP	28-pin DIP
Z0806000PSC	Z0856000PSC
Z0806000CSE	

Z8068, Z-DCP, 4 MHz
40-pin DIP
Z0806804PSC

ORDERING INFORMATION (cont):

Z80,000 CPU, 10 MHz
84-pin PGA
Z8000010GSE

Z160 CPU, 10.0 MHz
44-pin PLCC
Z0816010VSC

Z80320 CPU, 8 MHz
68-pin PLCC
Z8032008VSC

Z80320 CPU, 10 MHz
68-pin PLCC
Z8032010VSC

Z8516 DTC, 4 MHz
68-pin PCC
Z0851604VSC

Z8516 DTC, 6 MHz
48-pin DIP
Z0851606PSC
Z0851606CSE

Z8516 DTC, 6 MHz
68-pin PCC
Z0851606VSC

Z8030

6 MHz

40-pin DIP	44-pin PCC
Z0803006PSC	Z0803006VSC
Z0803006DSE	

8 MHz

Z0803008PSC	Z0803008VSC
Z0803008DSE	

4MHz

40-pin DIP	44-pin PCC
Z0853004PSC	

Z8530

6MHz

Z0853006PSC	Z0853006VSC
Z0853006PEC	
Z0853006DSC	
Z0853006DEE	

8MHz

Z0853008PSC	Z0853008VSC
Z0853008DSC	

Z85C30

6 MHz

40-pin DIP	44-pin PCC
Z85C3006PSC	Z85C3006VSC

8 MHz

Z85C3008PSC	Z85C3008VSC

10 MHz

Z85C3010PSC	Z85C3010VSC

Z80C30

6 MHz

40-pin DIP	44-pin PCC
Z80C3006PSC	Z80C3006VSC

8 MHz

Z80C3008PSC	Z80C3008VSC

10 MHz

Z80C3010PSC	Z80C3010VSC

Z8036 Z-CIO, 4.0 MHz

40-pin DIP	44-pin LCC
Z0803604CMB	Z0803604LMB

Z8036 Z-CIO, 6.0 MHz

40-pin DIP	44-pin LCC
Z0803606PSC	Z0803606LME
Z0803606DSE	Z0803606LMB
Z0803606DEA	
Z0803606CME	
Z0803606CMB	44-pin PCC
	Z0803606VSC

ORDERING INFORMATION (cont):

Z8536 CIO, 4.0 MHz

40-pin DIP	44-pin PCC
Z0853604CMB	Z0853604LMB

Z8536 CIO, 6.0 MHz

40-pin DIP	44-pin LCC
Z0853606PSC	Z0853606LME
Z0853606DSE	Z0853606LMB
Z0853606DEA	
Z0853606CME	**44-pin PCC**
Z0853606CMB	Z0853606VSC

Z8581 CGC, 6.0 MHz
18-pin DIP

Z0858106PSC	Z0858106CMB
Z0858106CSE	Z0858106LME
Z0858106CEA	Z0858106LMB
Z0858106CME	

Z8581 CGC, 10 MHz
18-in DIP

Z0858110PSC	Z0858110CMB
Z0858110CSE	Z0858110LME
Z0858110CEA	Z0858110LMB
Z0858110CME	

ORDERING INFORMATION

Codes

PACKAGE
Preferred
D = Cerdip
P = Plastic
V = Plastic Chip Carrier

Longer Lead Time
C = Ceramic
F = Plastic Quad Flat Pack
G = Ceramic PGA (Pin Grid Array)
L = Ceramic LCC
Q = Ceramic Quad-in-Line
R = Protopack
T = Low Profile Protopack

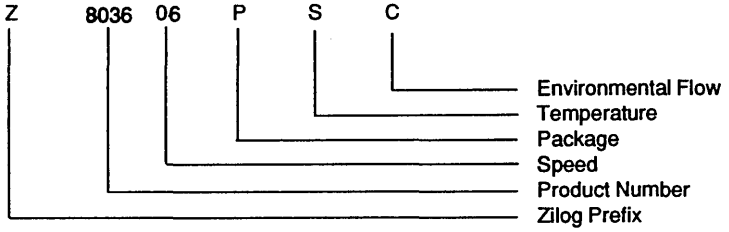
TEMPERATURE
Preferred
S = 0°C to +70°C

Longer Lead Time
E = -40°C to +100°C
M = -55°C to +125°C

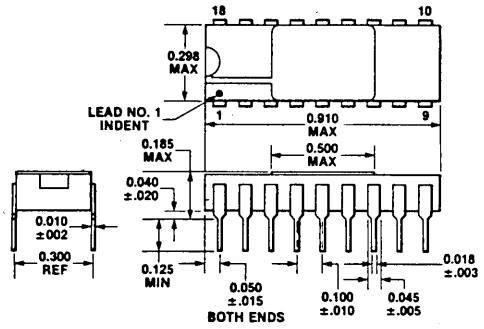
ENVIRONMENTAL
Preferred
C = Plastic Standard
E = Hermetic Standard
F = Protopack Standard

Longer Lead Time
A = Hermetic Stressed
B = 833 Class B Military
D = Plastic Stressed
J = JAN 38510 Military

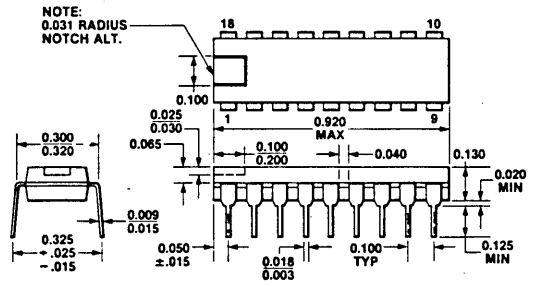
Example:
Z0803606PSC is an 8036, 6 MHz, Plastic, 0° C to +70° C, Plastic Standard Flow.



PACKAGE INFORMATION

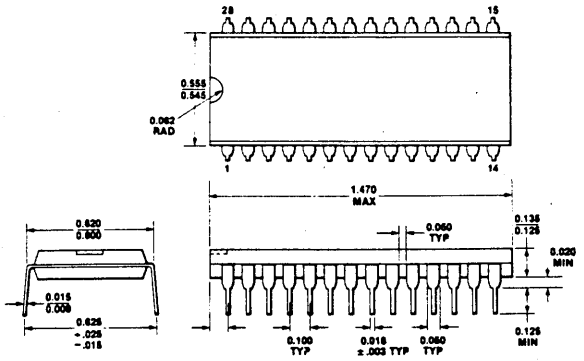


18-Pin Ceramic Package



18-Pin Plastic Package

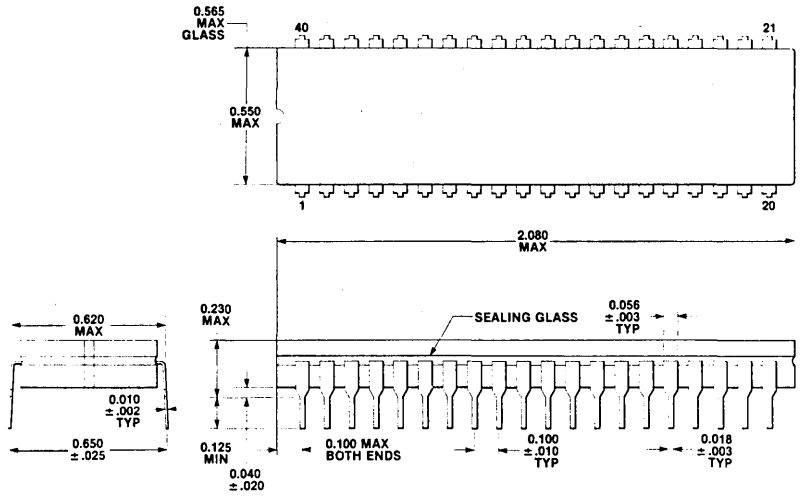
NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4



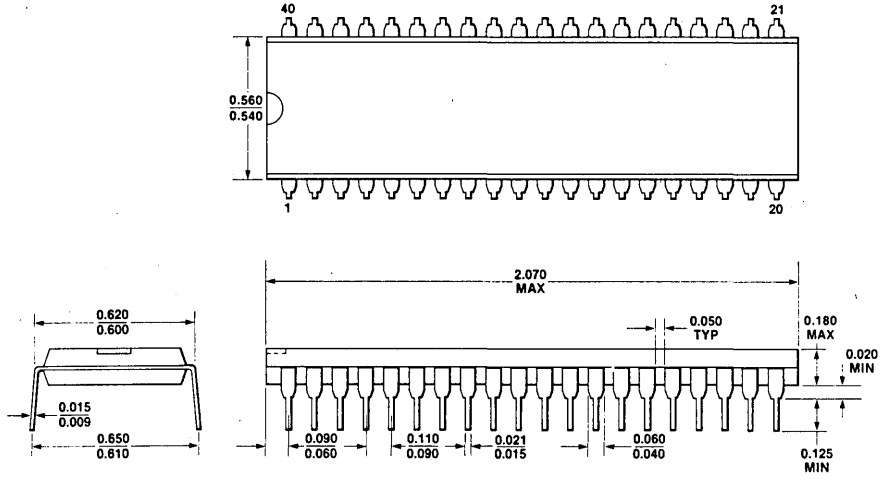
28-Pin Plastic Package

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

PACKAGE INFORMATION (Continued)



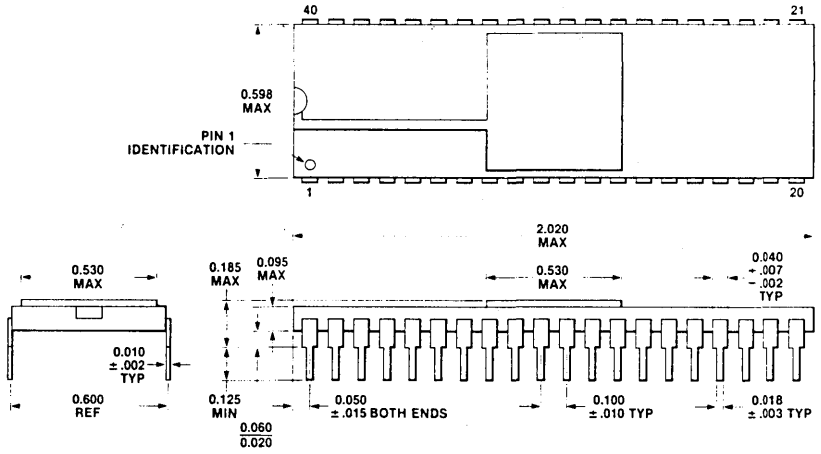
40-Pin Dual-in-Line Package (DIP), Cerdip



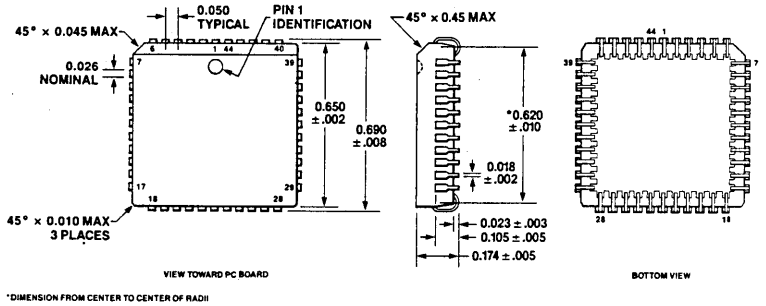
40-Pin Dual-in-Line Package (DIP), Plastic

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

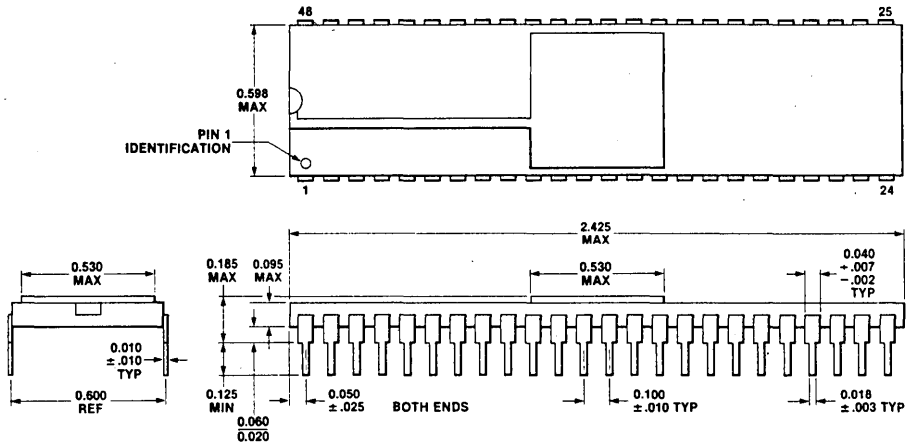
PACKAGE INFORMATION (Continued)



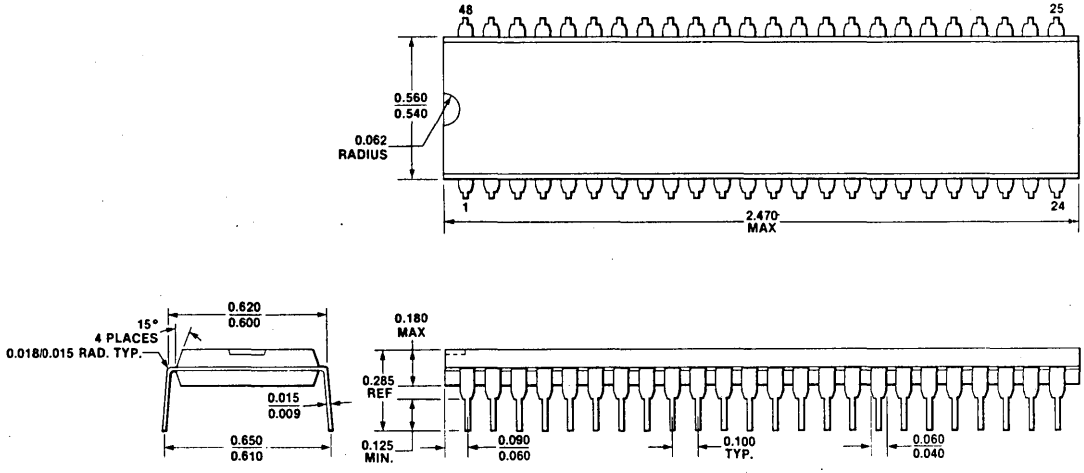
40-Pin Dual-in-Line Package (DIP), Ceramic



44-Pin Plastic Chip Carrier (PCC)

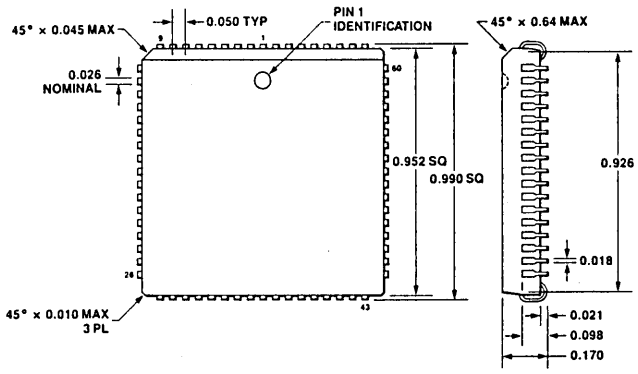


48-Pin Dual-in-Line Package (DIP), Ceramic

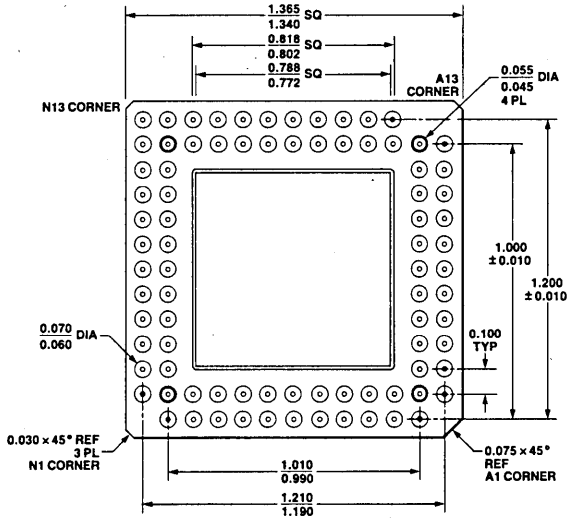


48-Pin Dual-in-Line Package (DIP), Plastic

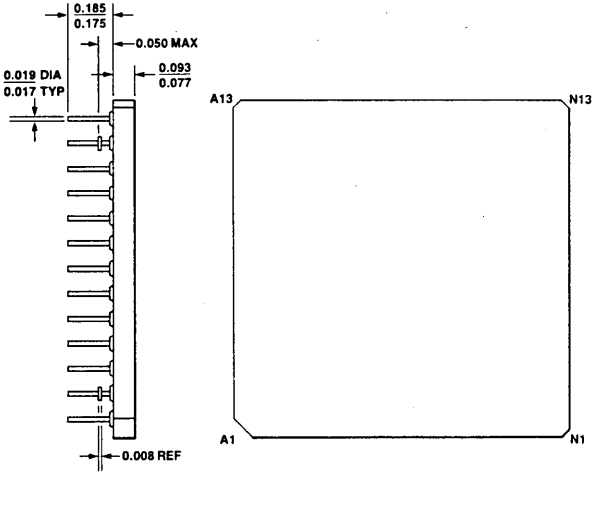
PACKAGE INFORMATION (Continued)



68-Pin Plastic Chip Carrier (PCC)



84-Pin Grid Array (PGA), Bottom View



View toward PC Board

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

Notes

Notes

Notes

Notes

Notes

Notes

Notes

ZILOG DOMESTIC SALES OFFICES AND TECHNICAL CENTERS

CALIFORNIA
Agoura 818-707-2160
Campbell 408-370-8120
Tustin 714-838-7800

COLORADO
Boulder 303-494-2905

FLORIDA
Largo 813-585-2533

GEORGIA
Norcross 404-923-8500

ILLINOIS
Schaumburg 312-517-8080

NEW HAMPSHIRE
Nashua 603-888-8590

MINNESOTA
Edina 612-831-7611

NEW JERSEY
Hasbrouck Hts. 201-288-3737

OHIO
Seven Hills 216-447-1480

PENNSYLVANIA
Ambler 215-653-0230

TEXAS
Dallas 214-987-9987

VIRGINIA
Arlington 703-525-5700

INTERNATIONAL SALES OFFICES

CANADA
Toronto 416-673-0634

GERMANY
Munich 49-89-672-045

JAPAN
Tokyo 81-3-587-0528

HONG KONG
Kowloon 852-3-723-8979

TAIWAN
Taipei 1-886-2-741-3125

ZILOG ASIA LTD.
Singapore 1-65-235-7155

UNITED KINGDOM
Maidenhead 44-628-39200

The information contained herein is subject to change without notice. Zilog assumes no responsibility for the use of any circuitry embodied in a Zilog product. No other circuit patent licenses are implied.

All specifications (parameters) are subject to change without notice. The applicable Zilog test documentation will specify which parameters are tested.

Zilog, Inc. 210 Hacienda Ave., Campbell, CA 95008-6609
Telephone (408) 370-8000 TWX 910-338-7621