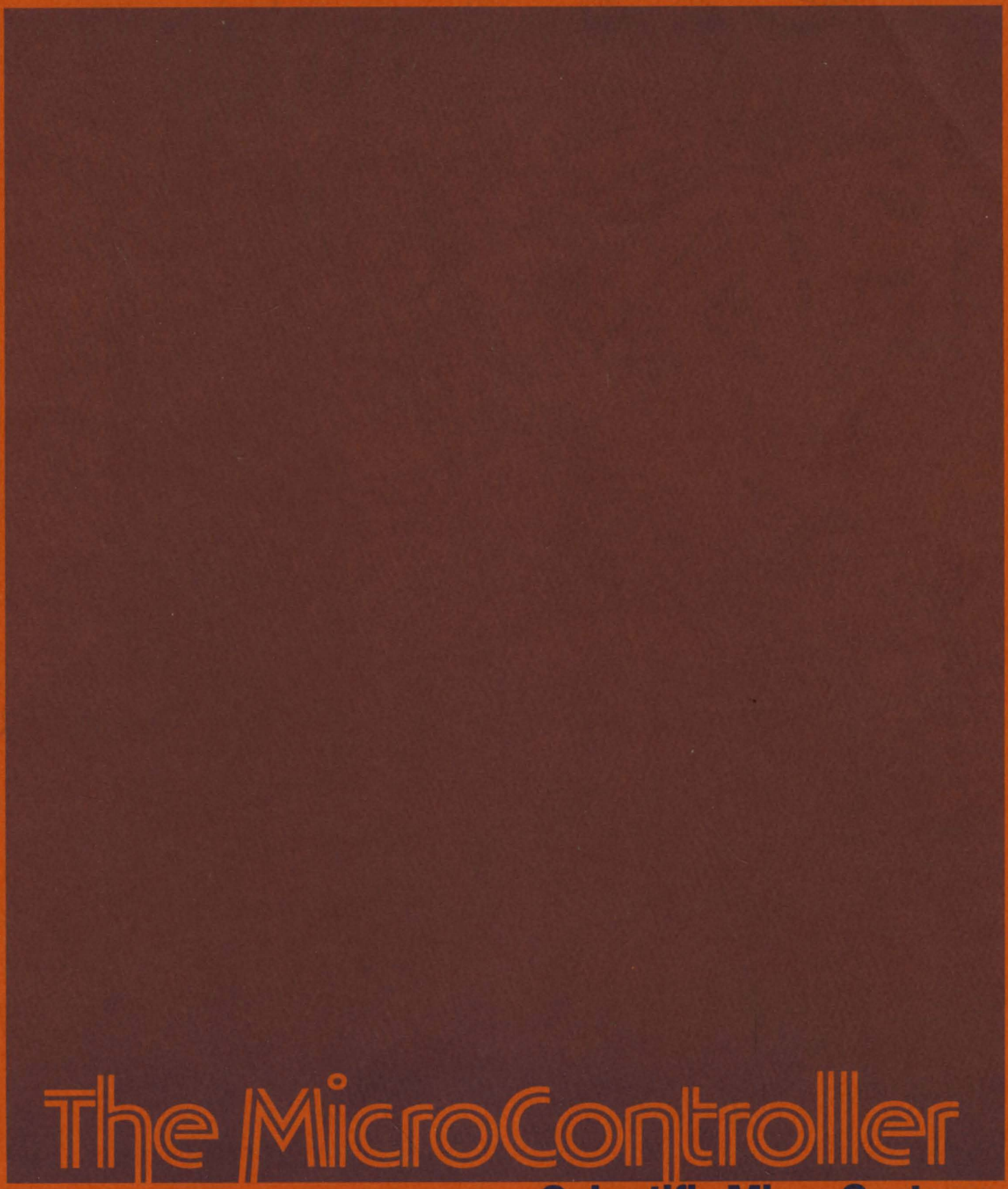
System Description



The MicroController

Scientific Micro Systems

The MicroController Digital System

System Description

Scientific Micro Systems, 520 Clyde Ave., Mountain View, Calif. 94043 415-964-5700

SCIENTIFIC MICRO SYSTEMS

The Name to Watch in High Technology Digital Systems

Scientific Micro Systems designs, developes, and manufactures advanced digital micro-systems. The latest technologies in system design, integrated circuits, and packaging are combined to produce reliable, high performance, cost effective systems.

SMS has a complete professional staff that includes

System Design

Component Design

Software Development

Manufacturing Engineering

Application Support.

SMS, a subsidiary of Corning Glass Works,

- Designed and developed Bipolar LSI memories
- Designed and developed the SMS ROM Simulator
- Designed and developed the SMS Micro-Controller

The material in this manual is for information purposes and is subject to change without notice. Scientific Micro Systems assumes no responsibility for errors which may appear herein.

TABLE OF CONTENTS

INTRODUCTION 2

MICROCONTROLLER ARCHITECTURE 4

System Elements 4

Processor Organization 6

Interface Vector 9

Working Storage 12

PRODUCTION SYSTEM 13

DESIGN SYSTEM 16

SPECIFICATIONS SUMMARY 20

INTRODUCTION

The SMS MicroController is a general-purpose microcomputer using bipolar LSI technology. It is designed to provide an effective and practical cost/performance solution to a wide range of problems. The MicroController provides significant advantages over the present methods of implementing digital system design for control.

COMPARISON OF METHODS OF CONTROL

	MicroController	Minicomputer	Microprocessor	Random Logic
Performance	State-of-the art	4-20 times slower	Current systems 10-50 times slower	Can be faster
Flexibility	High	Very High	Moderate	Very Low
Economics	Low Cost	High Cost	Low Cost	Variable

A control system should be able to respond to external variables by altering the program sequence and generating data for the outside world. A typical control application and its functions are shown in Figure 1.

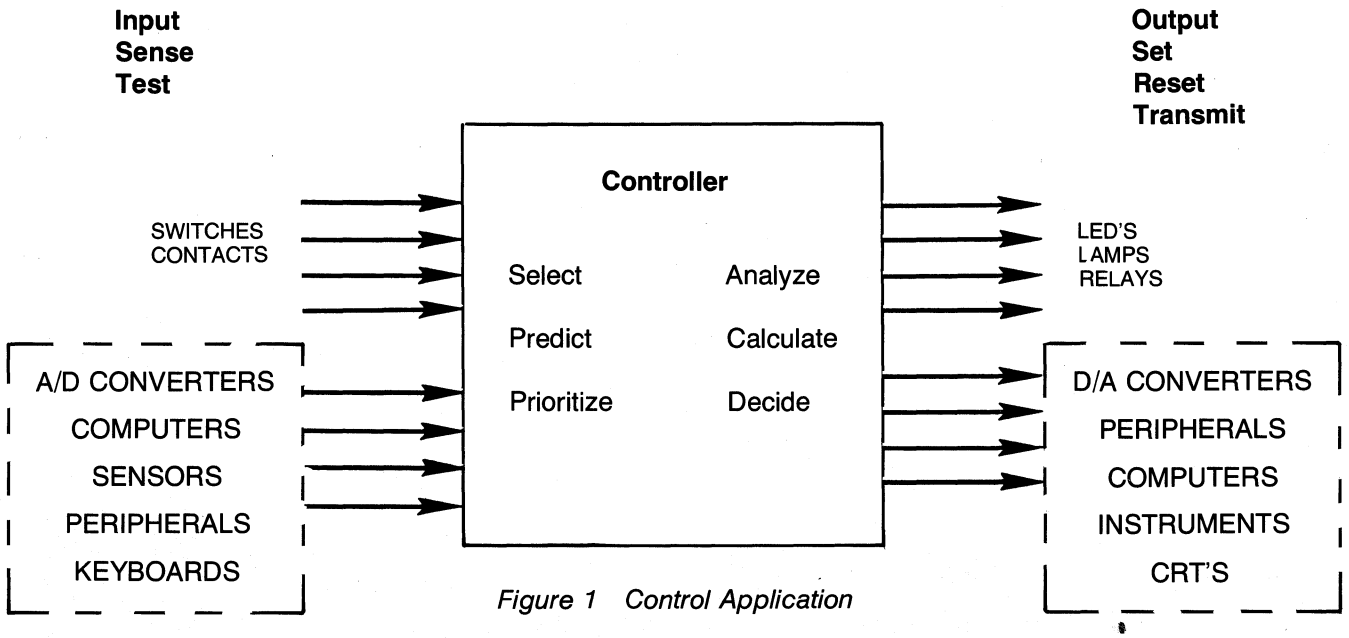


Figure 1 Control Application

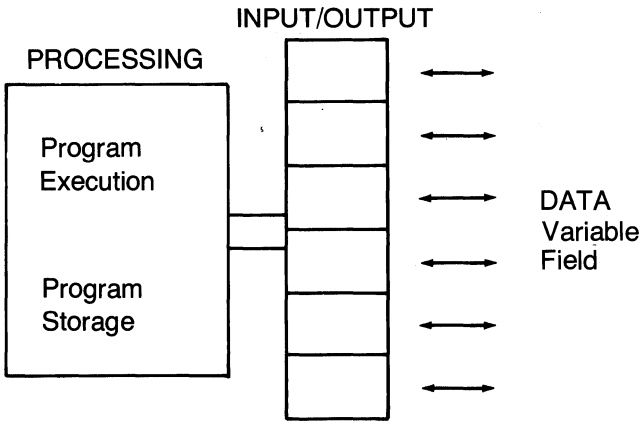


Figure 2

The MicroController becomes a tool for the designer to implement a programmable logic design. The capability exists in the system to store and execute control sequences. (Figure 2)

The interface uses a uniform method of input/output connection that accepts variable field length sizes; additional, expensive, custom circuitry for I/O is not required.

The programmed MicroController becomes the controlling subsystem in the overall specified system.

SYSTEM IMPLEMENTATION

The MicroController System is more than just a collection of components. It is a complete microcomputer system for obtaining programmed solutions. Designing a microcomputer from a microprocessor component is not easy. It may take as many as 55 additional integrated circuits to complete the system. This not only requires extensive knowledge of component interaction, but makes testing of the complete system difficult.
In the design of the MicroController System, the problems of both design and the implementation of the completed production system have been taken into account. The MicroController exists in two forms: a Design System and a Production System.

Design System

The Design System provides resources to aid in design implementation. Emphasis is placed on the designer's application knowledge rather than his knowledge of component design. Software support is provided to facilitate programming and minimize errors during the design. In addition, resources are available to the designer to operate and diagnose a system in real-time at the hardware level.

Production System

The Production System, complete with storage and input/output, is implemented with standard, volume-manufactured modules. The production version provides just those elements needed to execute the design with no superfluous modules: i.e., control panels, excess storage, interconnections, system overhead programs.

Environmental considerations can be minimized — 0-70° C. temperature range, hermetic packaging, single board construction dimensioned to combine with most packaging approaches. Reliability is achieved that is attainable only with today's technology.

APPLICATION

The application areas for the MicroController are broad. They include control in peripherals, data communications, terminals, measurement systems, and industrial applications. The MicroController's economical ease of use will undoubtedly open new applications not previously considered for microelectronics.

MICROCONTROLLER ARCHITECTURE

SYSTEM ELEMENTS

The principal elements of the MicroController System are the Processor, the Interface Vector and the Working Storage. Figure 3 shows the relationships of these elements to each other and to signals external to the system.

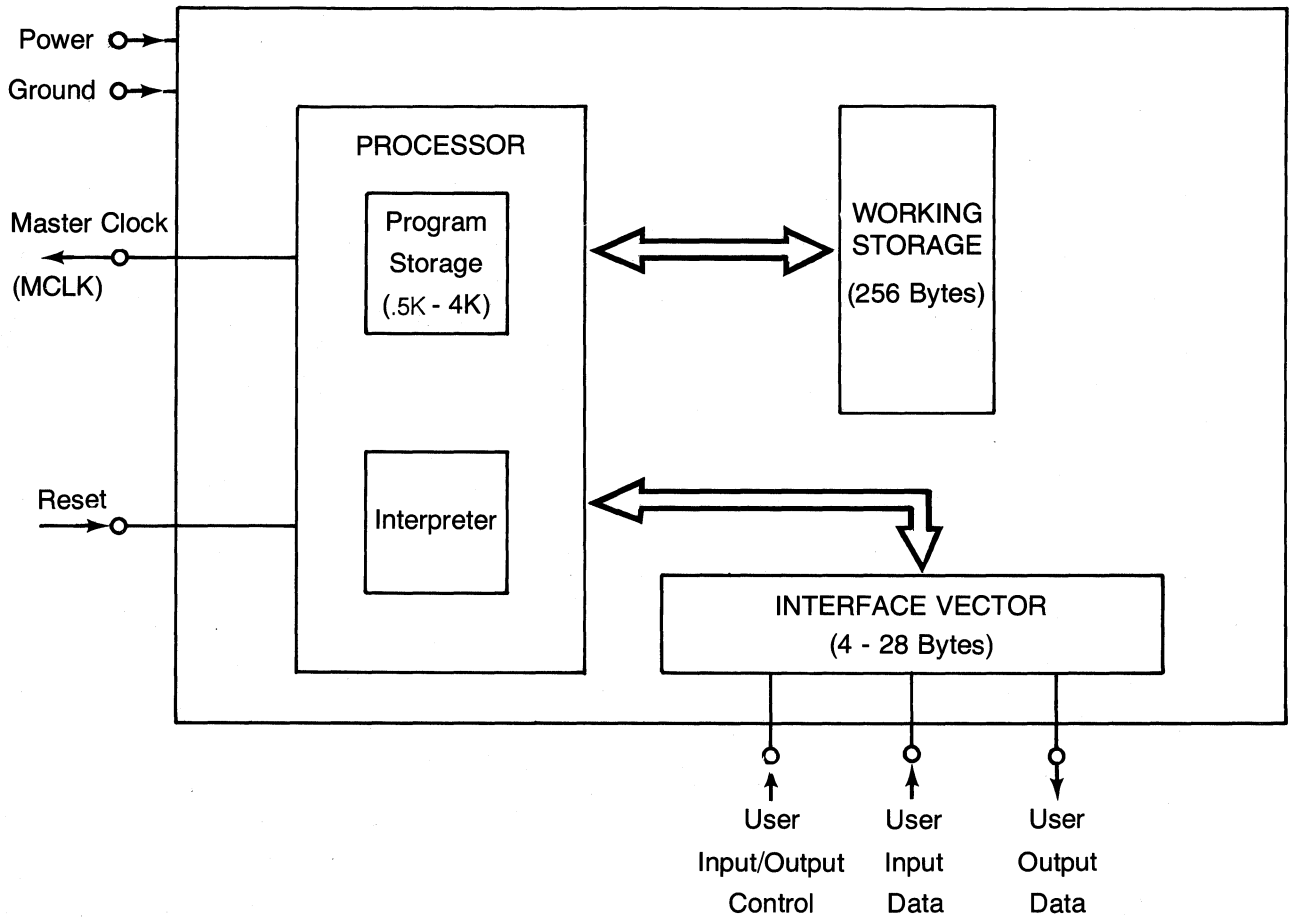


Figure 3 SMS MicroController

The Processor is composed of a programmable Interpreter and a read only Program Storage (4K words maximum). The Interpreter contains an arithmetic-logic unit (ALU), a program counter, 12 registers and operates with a repertoire of 8 instructions.

The Interface Vector is a program addressable, buffered, variable field length connection between the controlled elements of the User System and the MicroController. The Interface Vector is organized as 8-bit groups called IV Bytes. User control signals permit each Byte to serve as input, output, or a bi-

directional path. Signals from the User System may occur asynchronously with respect to the Master Clock. A maximum of 28 IV Bytes may be connected to a MicroController System.

The Working Storage provides 256 bytes of temporary storage. It may be viewed as an extension of the 8 data registers in the Processor. The Processor may address a variable length field in Working Storage. Intermediate storage for program data or input/output data buffering are typical uses for the Working Storage.

External input signals direct the operation of the MicroController System. The internal crystal controlled clock sets the operating rate of the system. The Reset signal establishes the initial state. User inputs via the Interface Vector, in combination with the stored program, control the execution sequence of program steps and the consequent value of user output signals available at the Interface Vector. User input/output control signals, Byte Input Control (BIC) and Byte Output Control (BOC), establish the instant at which the external user signals are input to or output from the MicroController System.

The Program Storage capacity, the size of the Interface Vector and the inclusion of Working Storage all are options the user may employ to match the capacity of the MicroController to his application. Additionally, the MicroController System may be used as the source of User System clocking signals by using the crystal controlled Master Clock (MCLK) output. The external system connections and their functions are summarized in Table I.

SYSTEM INTERCONNECTION - TABLE I

User Interface Connection	Function
Power/Ground	+5.0 v. supply to MicroController.
Reset	When low (~0 v.) forces Program Counter and Program Address Register to zero. Internal clock phase set to commence execution cycle when Reset line returns to normal level (+5.0 v.).
Master Clock	MicroController-provided clock (crystal controlled). The leading edge of this clock pulse defines the instant when the MicroController accepts User System data.
IV Data	Data input or output to IV Byte.
Byte Output Control	When active, data stored in an IV Byte is made available to User System.
Byte Input Control	When active, data present at IV Data connections is stored in the IV Byte.

Operating Cycle

The MicroController System is designed to execute a single command during each cycle (300 ns.) of the clock. The principal events during a cycle are:

1. Interpret command code.
2. Read Interface Vector bits or internal registers as specified by command.
3. Perform instructed operation.
4. Write into specified Interface Vector bits or internal registers.
5. Address next program word.

PROCESSOR ORGANIZATION

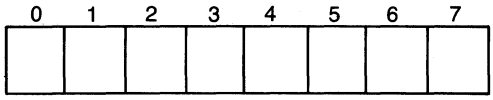
The structure of the MicroController Processor is described in the following sections which outline the registers, input, output and instruction set and is shown in the block diagram (Figure 4). The design has two important aspects that qualify it as a control oriented processor; namely:

1. Integration of input/output and data processing facilities at the instruction level.

2. Single instruction specification of a complete input/output transaction.

Instruction And Data Representation

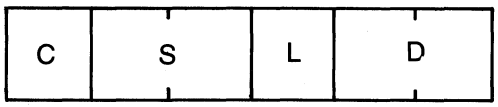
The system data element is an 8-bit Byte. By convention, the bit locations are identified in increasing order to the right, with the least significant bit in position 7.



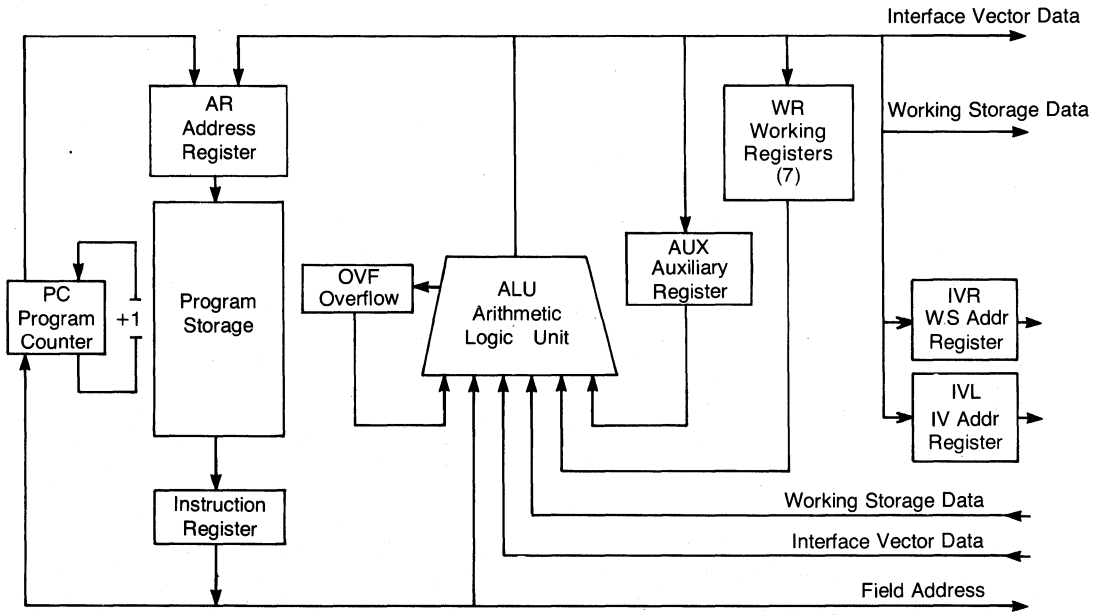
Within the Processor, all operations are performed on fixed length (8-bit) fields. However, data access to the Interface Vector and Working Storage is specified as a variable length field. The desired field within a Byte is identified by its right-most bit and its length. For example, the 3-bit field encompassing bit positions 2, 3, and 4 is specified as 4, 3. These fields are always right justified and unspecified positions filled with zeros when operated upon by the Processor. The MicroController Processor performs 8-bit, unsigned, 2's complement arithmetic.

Throughout this description, octal representation of instructions and addresses will be used. MicroController System names will be used for reference to internal registers and operation codes.

The Interpreter uses a two-address instruction format. The instruction word specifies the operand field read (S, L), the operation performed (C) and the operand field into which the result is written (D, L).



S and D identify registers or Interface Vector/Working Storage bit locations. L specifies the length of the data field. The combination of D and L fields or D alone is used in some instances to permit representation of an immediate operand. The JMP command needs only a single immediate value, thus S, L, D are treated as a single field. These departures in format are discussed more fully in a later section.



Processor Module Block Diagram

Processor Registers, Input/Output And Instructions

Summaries follow of the essential characteristics of the Processor data and control registers plus input signals and output signals. Additional reference to Figure 4 may be helpful. The distinction between register and input/output paths is accomplished by partitioning the address values that appear in an instruction word (S and D fields). Instructions thus may operate on data held in working registers or on input/output quantities with equal facility.

INPUTS

IV DATA—A variable field read from the Interface Vector to the ALU, right justified; addressed by the S, L operand fields and IV address register IVL.

WORKING STORAGE DATA—A variable field read from the Working Storage to the ALU, right justified; addressed by the S, L command fields and Working Storage address register IVR.

RESET—When active (low logic level) causes Program Counter and Address Register to reset to zero. A system input signal.

OUTPUTS

IV DATA—A right justified variable field to be written from the ALU into the IV Byte specified by the D, L Operand fields and IVL.

WORKING STORAGE DATA—A right justified variable field to be written from the ALU into the Working Storage address specified by the D, L operand fields and IVR.

MASTER CLOCK (MCLK)—A duplicate of the internal system clock for external (User System) timing.

CONTROL REGISTERS

INSTRUCTION—A 6-character register containing the current instruction.

PROGRAM STORAGE ADDRESS (AR)—A 5-character register containing the address of the current instruction being accessed from Program Storage. The first character must be 0.

PROGRAM COUNTER (PC)—A 5-character register containing the address of the next instruction to be read from Program Storage. The first character must be 0.

IV BYTE ADDRESS (IVL)—A 3-character register containing the address of the current byte being accessed from the Interface Vector. IVL is under program control.

WORKING STORAGE ADDRESS (IVR)—A 3-character register containing the address of the current byte being accessed from Working Storage. IVR is under program control.

DATA REGISTERS

WORKING REGISTERS (WR)—Seven 8-bit registers for data storage.

OVERFLOW (OVF)—A 1-bit register that retains the most significant bit position carry from ALU. Arithmetically treated as 2^0.

AUXILIARY (AUX)—An 8-bit register. Source of implied operand for arithmetic instructions. May be used as a working register.

ADDRESS PARTITIONS

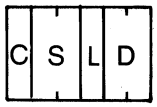
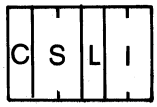
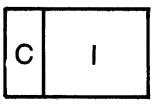
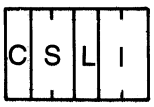
Addresses (octal) used in S and D Fields:

00 - 17	Working Registers, OVF, AUX, IVL, IVR
20 - 27	IV Byte field address (right-most bit)
30 - 37	Working Storage Byte field address (right-most bit)

Instruction Set

The operations performed by each of the eight instructions are shown in Table II.

MICROCONTROLLER INSTRUCTIONS - TABLE II

OPERATION	FORMAT	RESULT	NOTES
MOVE		Content of data field addressed by S, L replaces data in field specified by D, L.	If S and D both are register addresses then L specifies a right rotate of L places applied to the register specified by S.
ADD		Sum of AUX and data specified by S, L replaces data in field specified by D, L.	
AND		Logical AND of AUX and data specified by S, L replaces data in field specified by D, L.	
XOR		Logical exclusive OR of AUX and data specified by S, L replaces data in field specified by D, L.	
XMIT		The literal value I replaces the data in the field specified by S, L.	If S is IV or WS address then I limited to range 00-37. Otherwise I limited to range 000-377.
JMP		The literal value I replaces contents of the Program Counter.	I limited to the range 00000 - 07777.
NZT		If the data in the field specified by S, L equals zero, perform the next instruction in sequence. If the data specified by S, L is not equal to zero, execute the instruction at address determined by using the literal I as an offset to the Program Counter.	If S specifies an IV or WS address then I is limited to the range 00 - 37. I is limited to the range 000 - 377 otherwise.
XEC		Perform the instruction at address determined by applying the sum of the literal I and the data specified by S, L as an offset to the Program Counter. If that instruction does not transfer control, the program sequence will continue from the XEC instruction location.	The offset operation is performed by reducing the value of PC to the nearest multiple of 32 (if I : 00 - 37) or 256 (if I : 000 - 377) and adding the offset.

THE INTERFACE VECTOR

The Interface Vector (IV) is the input/output path between the MicroController System and the user equipment. Each bit in the interface provides a program addressable, buffered, bi-directional path. Both the MicroController and the user have simultaneous access to each bit for read or write operations. IV bits are grouped into 8-bit Interface Vector Bytes to simplify user control of the interface and access by the program.

The Processor treats the Interface Vector as an n-word, variable field, random access storage. Control elements in the Interpreter specify the word (IV Byte) to be accessed as well as the bit position and length of the data field to be read or written. Figure 5 shows the logical organization of the Interface Vector.

The user views the Interface Vector as one or more 8-bit, parallel registers. Each register (IV Byte) has input/output control independent of all others. Further, access conflict between the User System and the MicroController is resolved in favor of the User System to insure that user data is not lost. The user connection to an IV Byte is bi-directional. The IV Byte control signals permit the user to restrict an IV Byte to input or output operation only, or to use an IV Byte to terminate a bi-directional bus. IV Byte controls may be driven by the data output of other IV Bytes to permit full or partial program control of an IV Byte function.

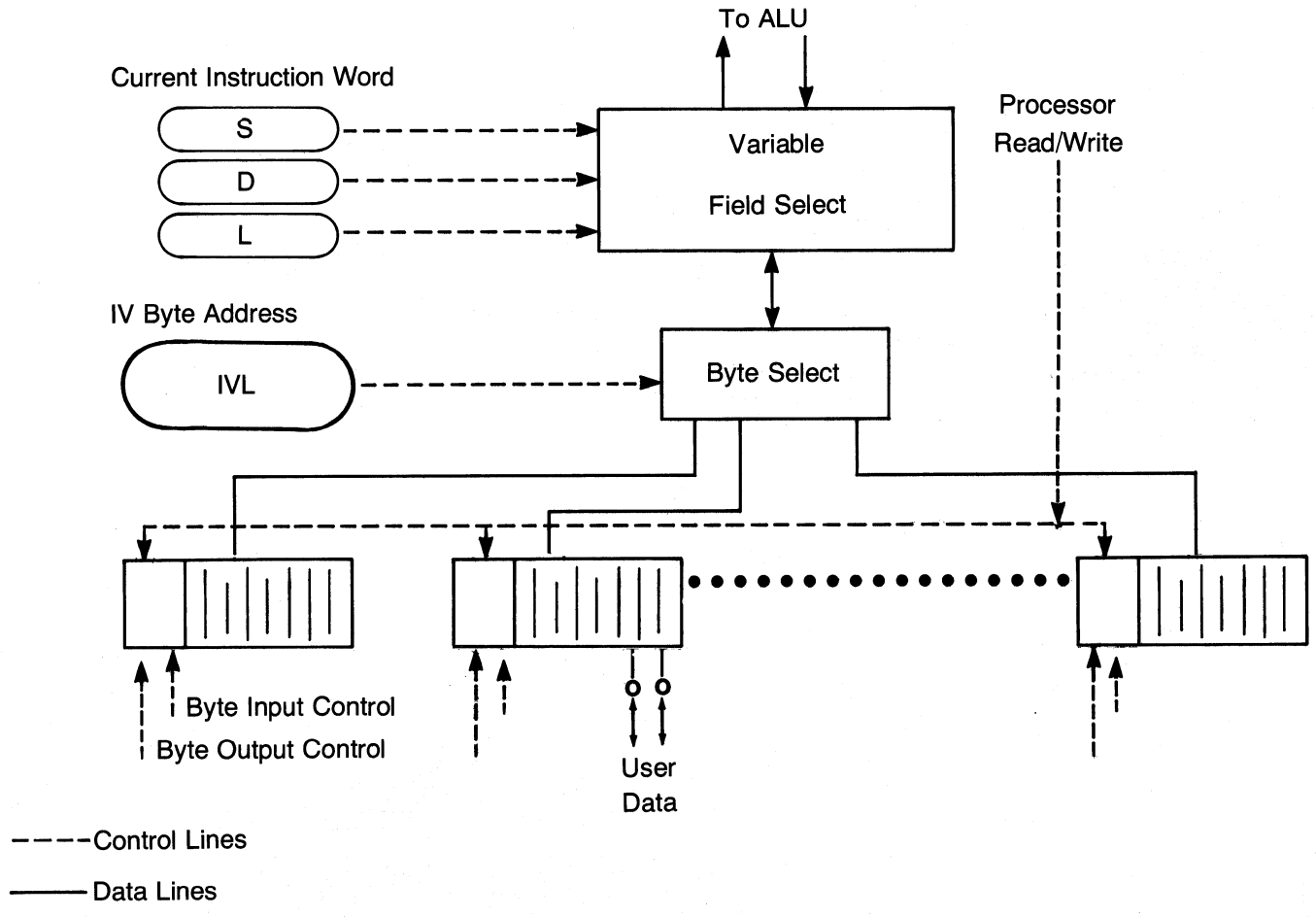


Figure 5 Interface Vector Organization

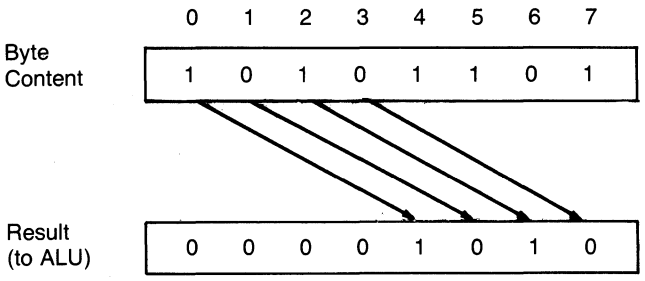
IV/Processor Interaction

The data source or destination for the Interface Vector is the Processor ALU. The register IVL (Figure 5) contains the address of the byte being accessed. The S, L, and D fields of the current instruction specify the bit field being accessed. The S field specifies the right-most bit position of the field being read.

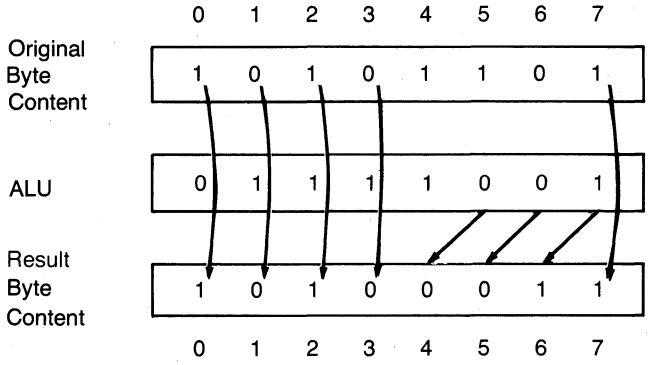
The D field specifies the right-most position into which the result from the ALU is to be written. The L field of the current instruction specifies the length of the field transferred. Bit fields read from the Interface Vector are transmitted to the ALU right justified with leading zeroes inserted.

Examples:

READ: IV FIELD SPECIFICATION S=3, L=4



WRITE: IV FIELD SPECIFICATION D=6, L=3



IV/User Interaction

The state of the two user interface signals, Byte Input Control (BIC) and Byte Output Control (BOC), establishes the direction of information transfer between the MicroController System and the User System. These signals are active when at low voltage logic level (0-0.5 v.). The effect of each of the four possible combinations is shown in Table III.

FUNCTION OF INTERFACE SIGNALS - TABLE III

Control Signals			Function
BOC (L)	BIC (L)	MCLK	
H	H	X	User Data Line Inactive (Hi-Impedance State)
L	H	X	User Data Lines Active. IV Byte Data available to user.
X	L	L	User Data Lines Inactive (Hi-Impedance State), no operation.
X	L	H	User Data Lines Active. User Data will be stored in IV Byte.
L - 0 - 0.5 volt	H - 2.4 - 5 volt		X can be either H or L

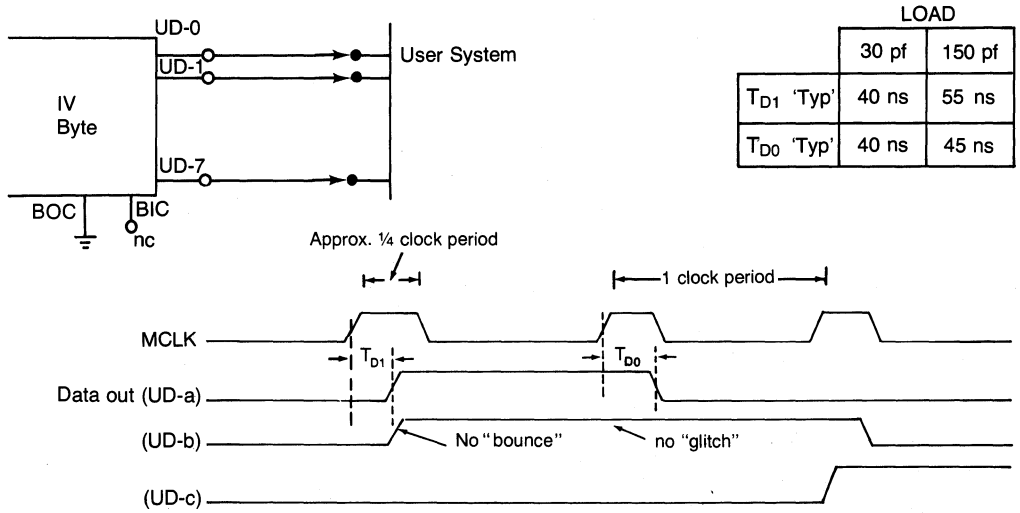
From Table III it is clear that a byte is accessible to the user whether or not the byte is being addressed by the Processor. Further, if both BIC and BOC are simultaneously active, then an input of user data occurs. The IV Byte provides standard TTL input and output levels. Both Tri-state output and open collector output versions are available as specified options. As a further aid to interface implementation, the IV Bytes uniformly assume an internal state of logical 1 when power first is applied to the system. This fact may be used to automatically prevent unwanted circuit or device operation during the power on sequence.

In most cases, the user will find permanent assignment of an IV Byte (or group of IV Bytes) to either input or output to be most advantageous. This is easily accomplished by permanently grounding BIC or BOC as required. The timing relation between data and Master Clock (MCLK) is shown in Figure 6a and 6b. The acceptable duration of input or output signals should be several clock cycles to permit greatest freedom in generating the program for the MicroController. However, the system can be programmed to operate successfully with input pulse durations as short as 1.25 clock periods.

Figure 6a depicts three different outputs from an IV Byte. The three waveforms (UD-a,b,c) typify transitions that might be found during program execution.

Output UD-a and UD-b are from a pair of outputs simultaneously set to H by a MicroController program. UD-a is reset to L by the next MicroController step. Note that there is no effect on the output level created by the Master Clock changing state. There is also no 'bounce' in the leading edge of the waveform.

Figure 6b shows the typical input situation. Waveforms UD-a,b represent extremes in the arrival times of an input relative to the Master Clock. Note that the Processor is guaranteed to have an input available for processing within one clock period of its arrival. The leading edge of 'User Data In' can arrive at any time. However, data will be stored in the IV Byte during the period in which the input signal brackets the clock.



	LOAD	
	30 pf	150 pf
T_{D1} 'Typ'	40 ns	55 ns
T_{D0} 'Typ'	40 ns	45 ns

Figure 6-a Dedicated Output

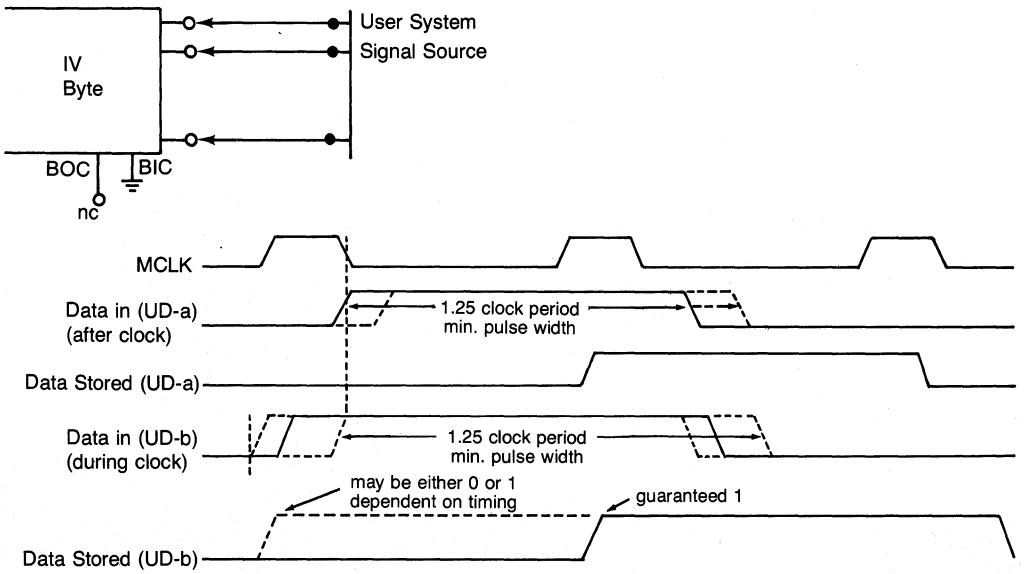


Figure 6-b Dedicated Input

WORKING STORAGE

A 256-byte Working Storage may be included in the MicroController System as an option. Typical uses are as a small buffer storage or as storage for intermediate program data. The Working Storage is viewed by the Processor in much the same manner as is the Interface Vector. Figure 7 shows the logical structure of Working Storage. The Storage is organized as two pages each containing 128 bytes.

A byte is accessed by storing its address in register IVR. The S, L, and D fields control variable field access and data read/write operations in the same manner as for the Interface Vector. The Page Select Register, located within the Working Storage, contains the value which identifies the page to be accessed.

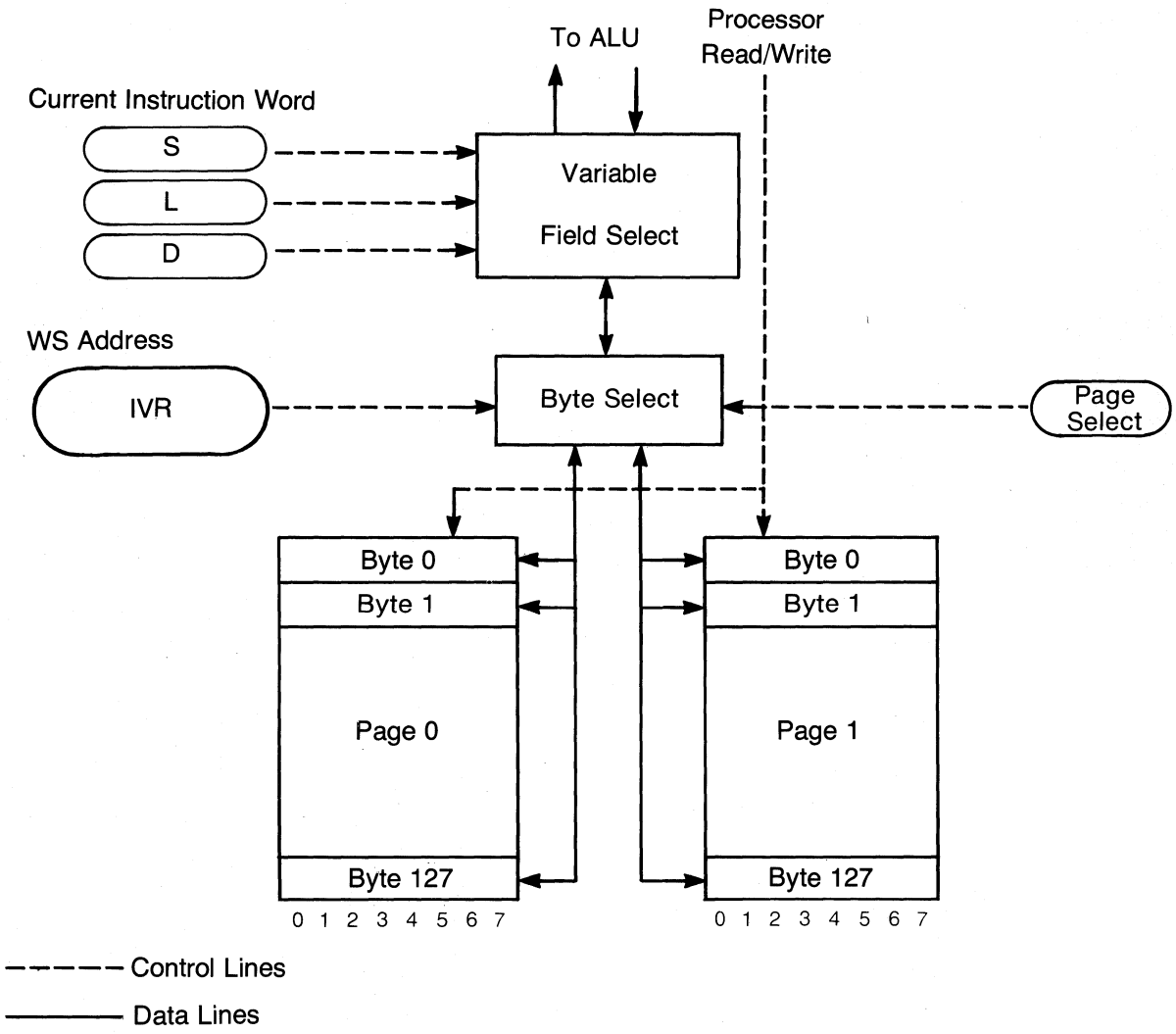


Figure 7 Working Storage

PRODUCTION SYSTEM

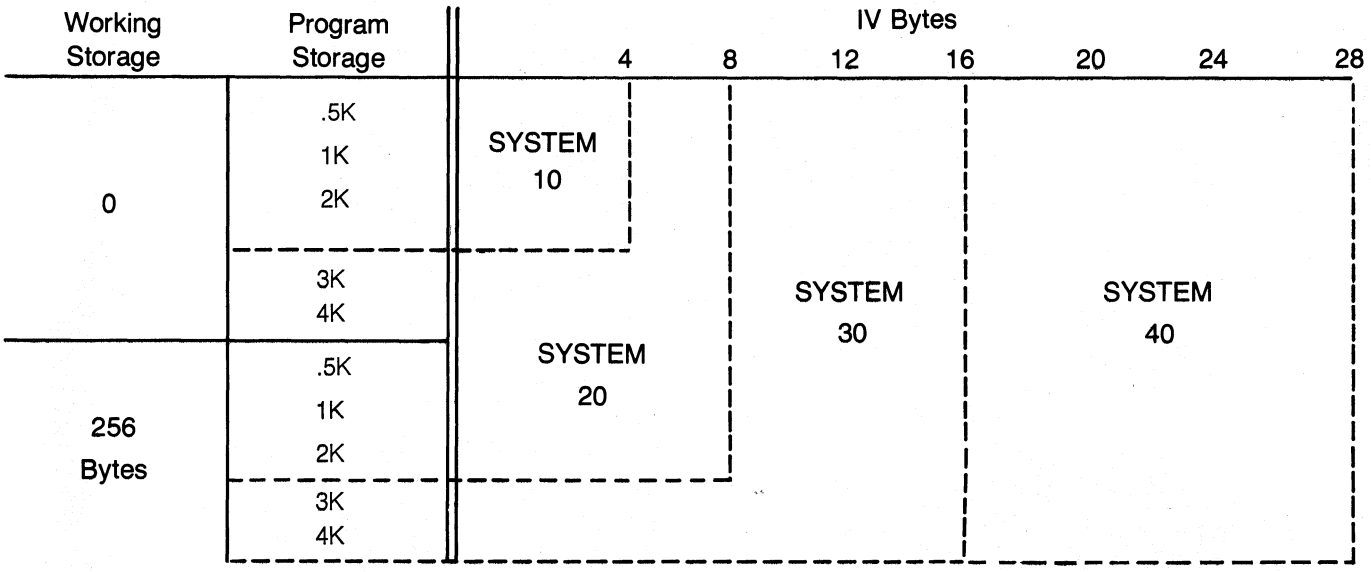
The range of logical and physical variation obtainable within the framework of the MicroController System is described in this section. The system modularity maximizes the latitude for matching performance, size and cost and minimizes the interconnect and design problems that are visible to the user at this level of the system.

Four MicroController System package sizes are available. The major variants in each of the configurations are (1) Program Storage capacity, (2) Number of IV Bytes, and (3) Presence or absence of Working Storage. Table IV summarizes the configurations and their capacities. Panel dimensions for each system are shown in the table. Note that Program Storage may be 0.5K to 4K words. IV Byte capacity is incremented in 4 Byte units or 32 1/0 points.

System size may vary as follows:

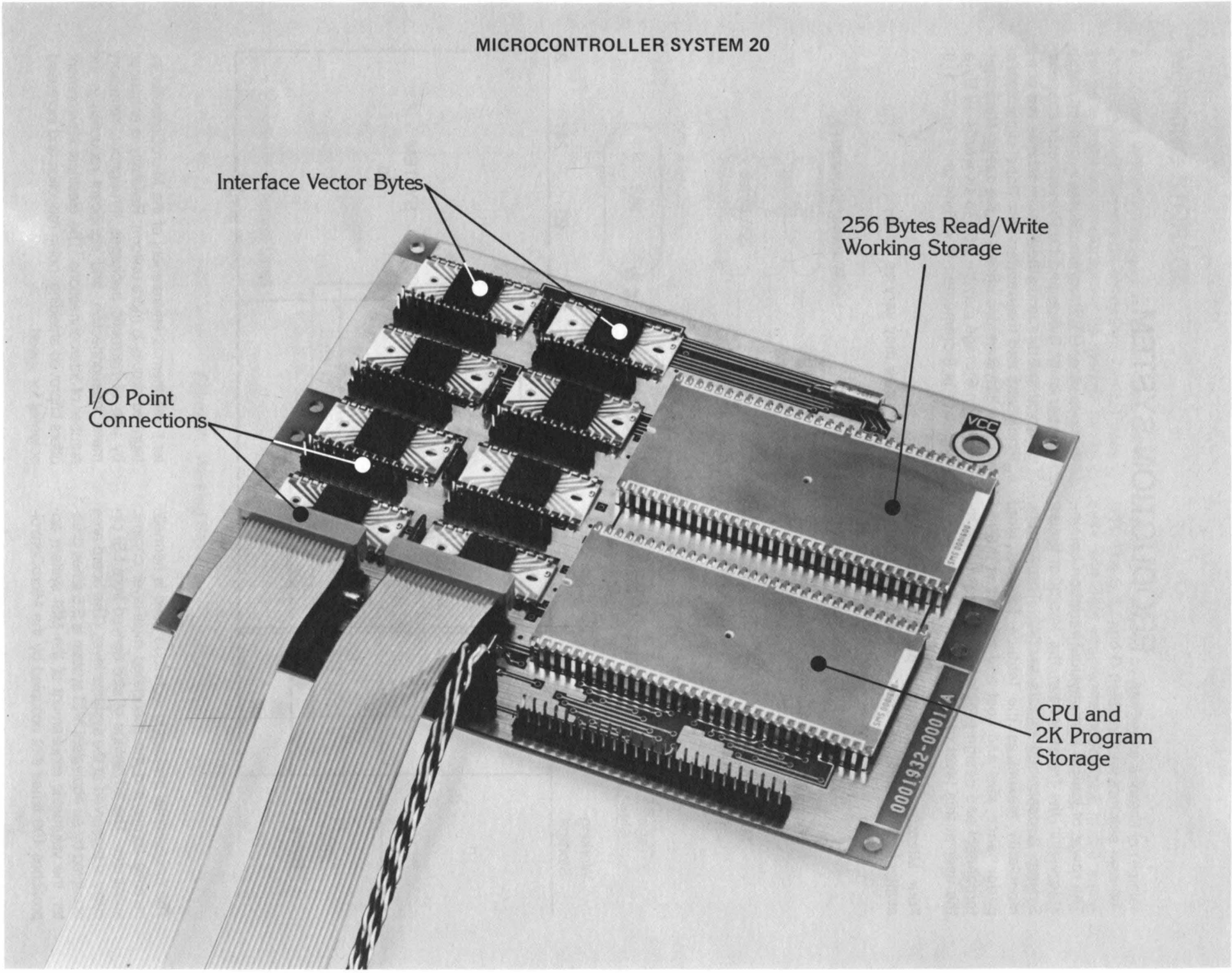
	Program Storage	I/O	Working Storage
Minimum System (System 10)	.5K Words (8K Bits)	4 IV Bytes (32 I/O Points)	0-
Maximum System (System 40)	4K Words (64K Bits)	28 IV Bytes (224 I/O Points)	256 Bytes

MICROCONTROLLER CONFIGURATIONS - TABLE IV



The fundamental packaging technique used is hermetic ceramic carrier and the associated multi-layer ceramic substrates. The advantage of size derived from LSI circuitry is preserved at the systems level. (The board area occupied by an equivalent MSI system is 3.5 times greater; the volumetric requirement of the MSI system approaches 100 times that occupied by the MicroController.) The thermal environment of the MicroController integrated circuits is more uniform. Reliability is enhanced by use of hermetic packages throughout, refractory metal interconnection, and significant reduction in the number of interconnections. The electrical environment offers improved shielding, noise reduction and increased potential for speed.

MICROCONTROLLER SYSTEM 20



All systems consist of standard, volume manufactured subassemblies. Each system is completely asembled and tested prior to shipment. The system arrives ready for use.

Panel dimensions and power requirements for all four systems are shown in Table V.

SYSTEM PANEL DIMENSIONS AND POWER REQUIREMENTS
TABLE V

System	**Dimension**	**Power Watts**
10	6.875 in. x 2.675 in.	11
20	6.875 in. x 5.375 in.	21.3
30	6.875 in. x 8.075 in.	34
40	6.875 in. x 16.150 in.	40

DESIGN SYSTEM

The problems faced by a designer in developing his system are different from those of the manufacturer in implementing production of the system. The manufacturer is concerned that a system element economically meets the advertised specification. The designer must have a means to plan, document, and test the performance of his design against product requirements. The MicroController Design System provides the resources necessary to assist the designer in these tasks. The MicroController Simulator (MCSIM) and the MicroController Machine Compiler (MCMAC) are the principal elements of the Design System.

MICROCONTROLLER MACHINE COMPILER (MCMAC)

MCMAC supplies the user with a statement oriented, symbolic programming system. Its design development features permit the description and documentation of a MicroController program. Using MCMAC provides:

Use of Symbolic Instructions	Symbolic-to-machine language translation.
Resource Allocation	Automatic assignment for the Interface Vector and Working Storage.
Program Development Aids	Extensive file maintenance and editing capabilities.
Design Documentation	Output files of Program Storage allocation, error listings, Interface Vector and Working Storage data allocation.
Input for MCSIM	Creates paper tape of MicroController program for input to MCSIM.
Manufacturing Data	Automatic creation and maintenance of a manufacturing file that describes completely the production implementation for the system design.
Low Cost Availability	The MCMAC System is accessible through a nationwide time-sharing service.

MCMAC simplifies the program description and bookkeeping by using symbolic representations for both the operation code and operand part of the MicroController instruction. A set of mnemonics is used for the operation codes (MOVE, ADD, AND, XOR, XEC, NZT, XMIT, and JMP) as well as the data registers (RI, R2, ··· R6, R11, AUX, and OVF) and control registers (IVL, IVR). Applying the rules governing MCMAC constructs, the user may assign suitable names to Interface Vector and Working Storage bit fields and Program Storage addresses. The general format of the MicroController instruction is maintained in MCMAC.

The following MCMAC programming example shows a method of implementing a typical function encountered in logic design:

Programming example: Test an input field for a specific bit pattern XXX.

XMIT INPADR, IVL	Store IV Byte address named 'INPADR' in IVL.
XMIT XXX, AUX	Store the literal bit pattern 'XXX' in register AUX.
XOR INPFIELD, AUX	Compare AUX with specified field named 'INPFIELD.'
NZT AUX, ALPHA	Test result for '0.' Transfer if not '0' to Program Storage address ALPHA.
• • •	
ALPHA: continue	Continuation of program.

MICROCONTROLLER SIMULATOR (MCSIM)

MCSIM is a programming and design test instrument. It provides the designer with a MicroController with which he can interconnect and run his system in real-time. MCSIM is a MicroController with the addition of:

1. Modifiable Program Storage.
2. User oriented display and control panel for system operation and diagnosis.
3. Input/Output connector that is physically and electrically equivalent to the production MicroController.

The Control Panel contains the features to enter, run, and change programs and consists of a 22-key keyboard, a 36-character display and a paper tape reader on the front of the instrument. (See Figure 9.) A Teletype interface and synch output connector are at the rear. The display presents the following information:

- Contents of Program Storage
- Contents of Registers
- Contents of Interface Vector
- Verification of adherence to format rules.

The operator's options include:

- Modification of the program and data
- Specification of breakpoints
- Starting, stopping, or single-stepping the program
- Instruction insertion.

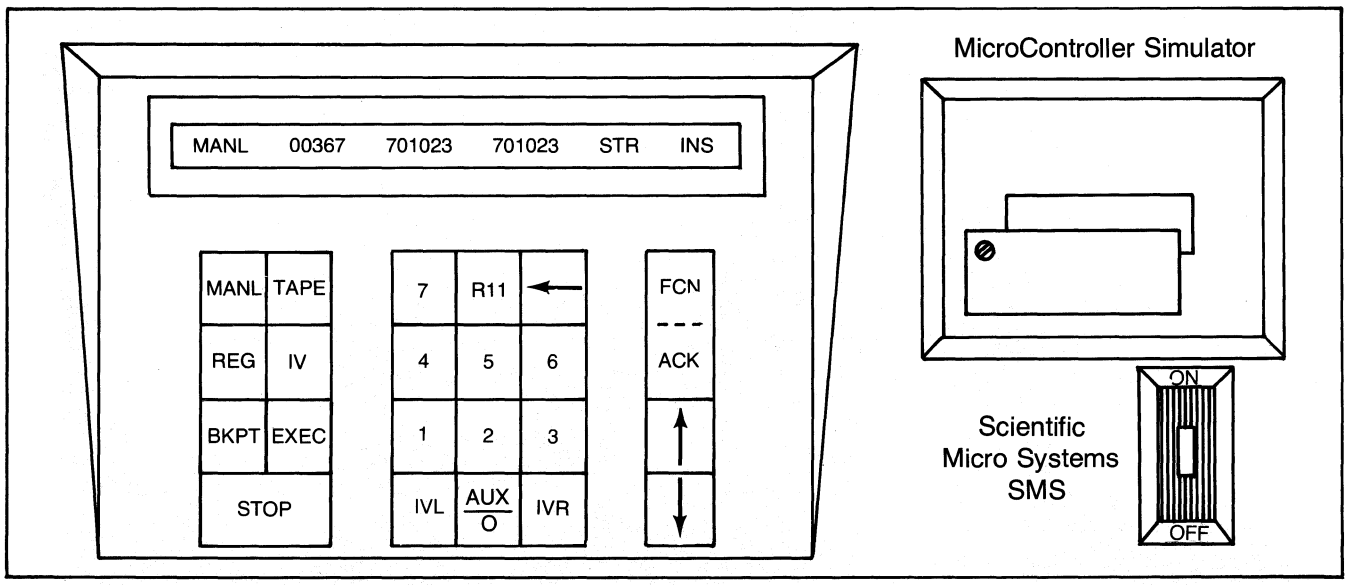
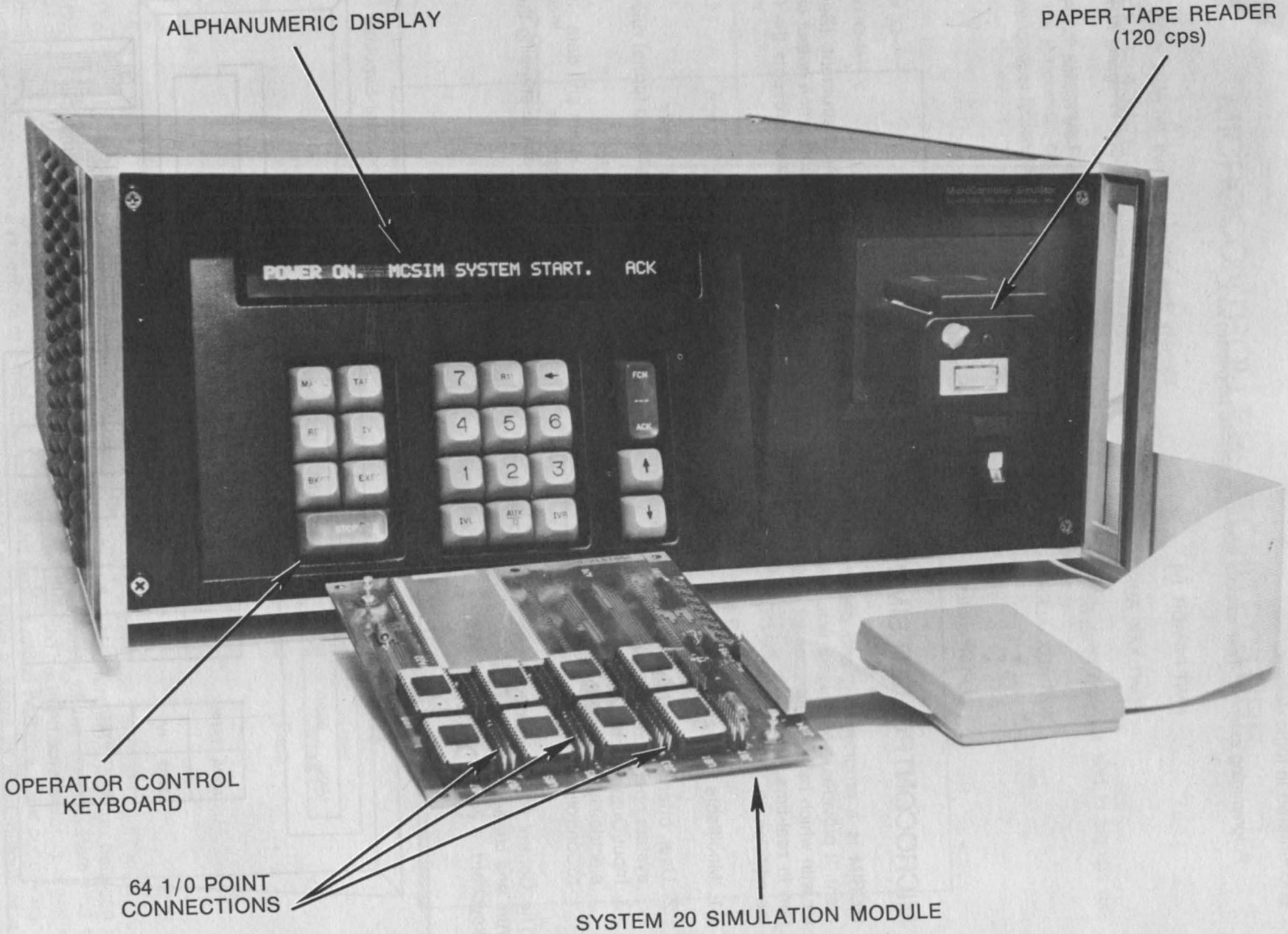


Figure 8 MCSIM Control Panel

MICROCONTROLLER SIMULATOR (MCSIM)



The Mode Select Keys and the Function Select Keys used in combination give the operator twenty-six (26) possible functions to accomplish complete loading and testing of the program. (See Table VI.) Each Mode Select Key accesses a set of functions. The Function Select Keys (↑ ↓) position a roll forward or backward to make the required function available.

MCSIM OPERATOR CONTROLLED FUNCTIONS - TABLE VI

Mode		Functions/Displays	
Manual	MANL	Change Address	CHG ADR
		Store Instruction	STR INS
		Program Identification	PROG ID
Tape	TAPE	Load	LOAD
		Verify	VERIFY
		Begin Punch Address	BEG PUN
		End Punch Address	END PUN
		Punch	PUNCH
Register	REG	Change Address	CHG ADR
		Store Octal Data	STR REG
		Store Binary Data	STR BIN
		Complement Overflow	CPL OVF
Interface Vector	IV	Display Current Enabled Bytes	ENABLED
		Change IV Byte Address	CHG ADR
		Store Binary Data in IV Byte	STR IV
Breakpoint	BKPT	Display Currently Active Breakpoint	CURRENT
		Set Compare Address	NORMAL
		Stop at Compare Address	STOP
		Replace Instruction at Compare Address	REPLACE
Execute	EXEC	Halt, Not in XEC Range	HALT
		Halt, Display PC in XEC Range	HALT PC
		Halt, Display RA in XEC Range	HALT XR
		Run Program	RUN
		Insert Instruction	INSERT
		Single Step	STEP
		Single Step in XEC Range	STEP XR

The input tape and documentation for debugging on MCSIM are supplied as output from MCMAC. The operator loads the program from the paper tape prepared by MCMAC. The MCMAC output listings assist in monitoring the program, detecting errors and recording changes. Should program errors be detected, the system development cycle (correct program - compile - debug) is repeated until the user is satisfied that the system is error free and meets his specifications. At that time, SMS obtains the necessary manufacturing and program information automatically through MCMAC and quickly delivers the production MicroController systems.

SPECIFICATIONS SUMMARY

Type	General purpose byte oriented LSI MicroController.
Program Storage	Up to 4096 words ROM/PROM.
Internal Storage	8-8 bit general registers. 2-8 bit I/O address registers. 1-1bit overflow register. 256-bytes Working Storage (optional).
Input/Output	Up to 224 connection points. Variable field size and bit addressable. Expandable in increments of 32. TTL (Tristate) bi-directional I/O points.
Instructions	16 bit instruction word. 8 instruction types including register transfer, arithmetic/logical, control, execute.
Speed	300 ns instruction cycle (fetch, decode and execute). Each instruction performed in single instruction cycle.
Packaging	Maximum 4096 word Program Storage, 224 I/O points, 256 byte Working Storage, Interpreter with crystal clock packaged in 6.9 in. x 16.2 in. x 1 in. volume.
Operating Temperature	0-70° C.
Power	Single 5 volt supply. Maximum current (maximum system) - 8 Amps. Typical current (maximum system) - 3 Amps.

Scientific Micro Systems

sms A Subsidiary of Corning Glass Works

MC555D
Printed in USA

Scientific Micro Systems, Inc., 520 Clyde Ave., Mountain View, CA 94043 · (415) 964-5700

MSI packages and would have required a huge, 64-pin DIP-type package if standard techniques had been used. Such packages are fragile, expensive and basically intended for low-volume applications.

The large-size packages can degrade the electrical performance of the chip they house because of the high resistance and capacitance of the lines between the chip and the package pins. Thermal characteristics of large ceramic DIP-type packages tend to be poor. The heat from the chip, buried at the bottom of three, 15-mil ceramic layers, has difficulty getting out to the top surface to be dissipated. And, if lead mounted, the DIP cannot dissipate effectively against the board.

So SMS, to solve both the packing density and heat problems at the same time, arrived at the concept of using hermetic, leadless, ceramic chip carriers mounted directly to a multilayer, cold-fired, ceramic substrate to form a system module.

If this sounds straightforward, it, in fact, wasn't. Because, according to Heinz Reusser, SMS manager of manufacturing development, even the best and most knowledgeable ceramics' suppliers don't fully understand the electrical requirements demanded of their packages; their primary concern is simply the mechanical integrity of their products.

So SMS had to adjust the design of the leadless carriers to optimize the electrical performance of the carriers. Interestingly, the cost of the redesign was so small that the SMS staff is surprised that more users are not doing it.

The adjustments were possible only because the carriers are square, with the chip-to-carrier-edge lines arranged radially about the central chip area. SMS equalized the line resistances and reduced them by an order of magnitude (from two ohms worst-case in a standard DIP layout to 0.2 ohms), reduced the line capacitances and so on. None of these adjustments would be possible in a standard, large DIP because of the line pattern geometry necessitated by the package shape.

The 56-pin leadless carriers used by SMS are custom modifications of standard products from American Lava (3M) and Kyocera. The carriers, with line spacing of 50 mils on-center about the edges, occupy only about 25 percent of the area needed by a DIP carrying a chip of comparable size. So, even though ceramic is costlier than PC materials, the saving in material-area needed balances the cost differential.

So far we've seen how SMS reduced the I/O area problem, eliminated chip line length problems and kept costs in line. But what about the thermal problem?

Again, SMS had to put another modification onto the carrier. For this, SMS added a heat-sink pad to the back of the leadless carrier. The pad serves two functions: used as the (reflow soldered) bonding point to a similar pad on the substrate, it firmly anchors the chip carrier assembly, thus eliminating the usual dependence on lead connections for the mechanical hold-down; and the pad provides a metallic joint to the substrate so that the heat from the chip flows directly through the thin alumina carrier to the pad on its back, through the solder joint, right into the heat-conductive alumina substrate —a feat impossible in conventional packaging systems.

In effect, SMS turns its packages upside down to mount them on the motherboard. In fact, William Ide, SMS technical staff member, claims that everyone would be a lot better off if they simply turned all their packages upside down, picking up the leads and easing the heat-sink problem at the same time.

Since each chip is hermetically sealed in its carrier, and the carriers are simply reflow soldered to the individual module's substrate, the resulting system module is repairable. Further, the temperature profile of a conventionally packaged system module exhibits hot and cold spots; SMS's modules do not. And the module substrate has sufficient area so that the system modules can operate in a 70°C, still-air environment, mounted on a conventional PC card.

The multilayer substrate itself is interesting. SMS starts out with a 15-mil thick, unfired ceramic tape, printing the various layers one after another along the length of the tape. The layers are stamped out, stacked and fired together to form a laminated, multilayer substrate.

However, the SMS substrate ends up with 15-mil-thick ceramic (as compared with the one or two mils of glass common in hard-fired substrates) between the interconnect layers. There are four layers of metal and three of ceramic, and the back side is a full ground plane; capacitance between the lines is one-seventh that of conventional thick-film processes.

The SMS packaging concept is a simple one, basically no more complex than that of PC assembly. But, for their particular product, it eliminates about four PC boards, as well as the associated board interconnects and backplane wiring. It also does away with forced-air cooling, yet increases the packing density.

SMS also feels that the concept preserves the advantages of LSI by reducing the number of interconnect levels. One level holds the module to the PC motherboard; a second lets the user get his signals in and out. •

Reprinted from ELECTRONIC PACKAGING and PRODUCTION, February, 1975, pages 15, 16, 18.
Copyright © 1975 by Milton S. Kiver Publications, Inc.

PACKAGING CONCEPT INCREASES PACKING DENSITY, DECREASES THERMAL PROBLEMS

Scientific Micro Systems recently announced a high-speed microcomputer, the SMS MicroController, which makes use of a unique, SMS-developed packaging scheme.

According to Glenn Oliver, SMS engineering vice president, the packaging scheme is unique for two reasons. First, it preserves the circuit-level advantages of LSI, carrying them to a high system level. (Which means that conventional packaging methods waste a great deal of the prime advantage of LSI, that of lots of circuitry on a single chip.)

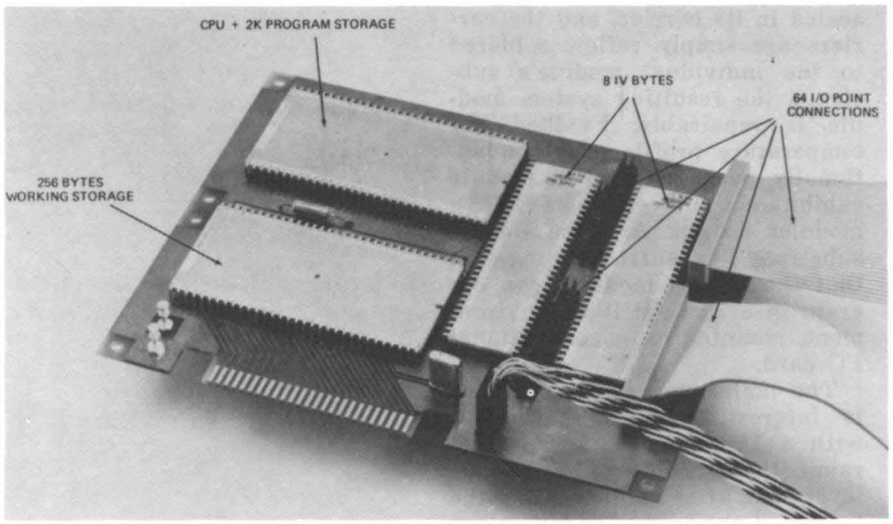
Second, the SMS packaging allows its microcomputer system to be specified as though it were a component. Typically, for example, a commercially rated, 0-70°C part has to be derated to a lower temperature in its working environment. But the thermal characteristics of the SMS system module are such that it requires no derating; it will operate in still air at a 70°C ambient. The designer's problems of fit, cooling, etc., are minimized at the beginning of the design.

SMS has radically reduced the number of packaging and interconnect strata that usually exist between the LSI chip level and the system level. Using established basic processes, but on a larger scale than attempted before, the Mountain View, Calif. firm has succeeded in packaging a system by packaging semiconductors rather than packaging semiconductors and building a system from those pieces.

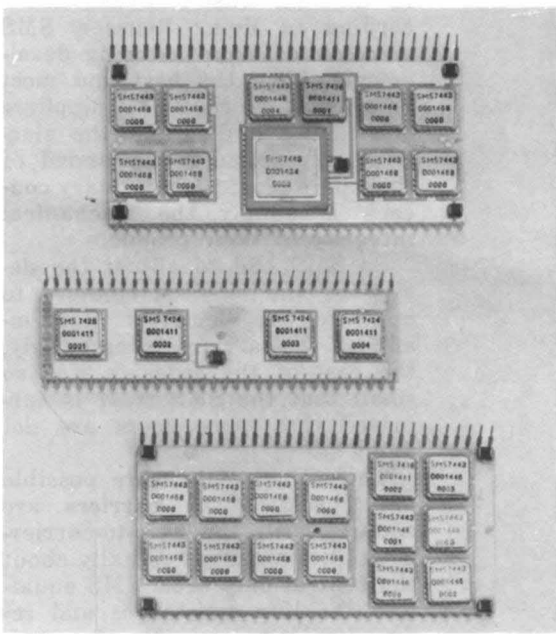
The SMS approach is not an ordinary multichip hybrid method. Such methods, in which chips are bonded directly to a substrate, allow a high component density but suffer from several drawbacks: where there is a great deal of input/output (as in microprocessors, for example), the number of pins needed at the packaging level, or the substrate area that they require, is excessive; for technologies other than CMOS, getting the heat out is a problem; repair of hermetic hybrids is extremely difficult and, in fact, may not be possible; and conventional hybrid techniques can be an expensive way to package a system.

Because SMS uses high-speed, high-power Schottky technology in its MicroController, the thermal characteristics of the packaging scheme are particularly important. Direct chip assembly methods (beam leads, solder bumps, etc.) would not have been good enough since SMS wanted to eliminate the need for forced-air cooling.

Further, the SMS bipolar CPU chip, at 250-mils square, is probably the largest chip in volume production. It replaces about 116



This medium-sized microcomputer in the SMS MicroController line shows the rugged and compact LSI packaging. The unit measures 5.4 x 6.9 in. and provides 2K 16-bit words of program storage, 64 I/O connection points and 256 bytes of working storage.



The SMS packaging concept makes use of chips mounted in hermetically sealed, leadless, ceramic carriers. These carriers are reflow soldered, by means of a heat-sink/mounting pad on their back, to a ceramic substrate. The carriers mounted to the substrate form a system module, which is simply inverted and mounted to a PC motherboard. The thermal characteristics of these modules is such that they operate in a 70°C, still-air environment.

Scientific Micro Systems
sms A Subsidiary of Corning Glass Works

SMS 300

INTERPRETER

The SMS 300 Interpreter is a monolithic, high-speed microprocessor implementing bipolar Schottky technology. It serves as the central processing unit, CPU, for the SMS MicroController, allowing 16-bit instructions to be fetched, decoded and executed in 300 nanoseconds. A 300 nanosecond instruction cycle requires maximum memory access of 85 nanoseconds, and maximum I/O device access of 40 nanoseconds.

Interpreter instructions operate on 8-bit, parallel data. Logic is distributed along the data path within the Interpreter. Input data can be rotated and masked before being subject to an arithmetic or logical operation; and output data can be shifted and merged with the input data, before being output to external logic. This allows 1- to 8-bit I/O and data memory fields to be accessed and processed in a single instruction cycle.

FEATURES

- 225 ns instruction decode and execute delay (with SMS 360 IV Byte).
- Eight 8-bit working registers.
- Single instruction access to 1-bit, 2-bit, 3-bit . . . or 8-bit field on I/O bus.
- Separate instruction address, instruction, and I/O data busses.
- On-chip oscillator.
- Bipolar Schottky technology.
- TTL inputs and outputs.
- Tri-state output on I/O data bus.
- +5 volt operation from 0° to 70°C.

The MicroController

Scientific Micro Systems

520 Clyde Avenue • Mountain View, California 94043 • (415) 964-5700 TWX: 910-379-6577

INTERPRETER INTERFACE

Signals described in Table 1 are used by the Interpreter to communicate with program storage and I/O devices. The four types of signals are:

1. Instruction address (A0—A12) outputs.
2. Instruction data (I0—I15) inputs.
3. Interface Vector (IV) Bus data ($\overline{IVB0}$—$\overline{IVB7}$) input/outputs.
4. Timing and control inputs/outputs.

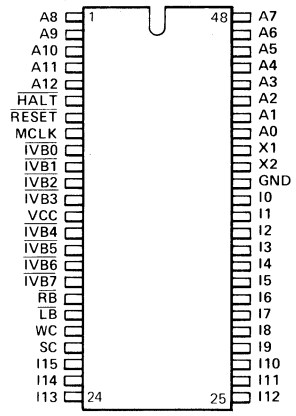


Figure 1a
Interpreter Pin-out

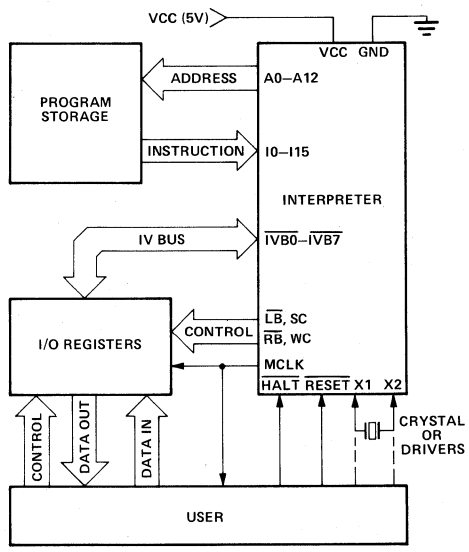


Figure 1b
Typical System Organization

Table 1
Pin Definition

A0-A12:	Instruction address lines. A high level equals "1." These outputs directly address up to 8192 words of program storage. A12 is least significant bit.
I0-I15:	Instruction lines. A high level equals "1." Receives instructions from Program Storage.
$\overline{IVB0}$-$\overline{IVB7}$:	Interface Vector (IV) Bus. A low level equals "1." Bidirectional tri-state lines to communicate with I/O devices.
MCLK:	Master Clock. Output to clock I/O devices, and/or provide synchronization for external logic.
WC:	Write Command. High level indicates data is being output on the IV Bus.
SC:	Select Command. High level indicates that an address is being output on the IV Bus.
$\overline{LB}$:	Left Bank. Low level output to enable one of two sets of I/O devices (LB is the complement of RB).
$\overline{RB}$:	Right Bank. Low level output to enable one of two sets of I/O devices (RB is the complement of LB).
$\overline{HALT}$:	Low level is input to stop the Interpreter.
$\overline{RESET}$:	Low level is input to initialize the Interpreter.
X1,X2:	Inputs for an external frequency determining crystal. May also be interfaced to logic or test equipment.
VCC:	5V power connection.
GND:	Ground.

Program Storage Interface

Program Storage is typically connected to the A0-A12 (A12 is least significant bit) and I0-I15 signal lines. An address output on A0-A12 identifies one 16-bit instruction word in program storage. The instruction word is subsequently input on I0-I15 and defines the interpreter operations which are to follow.

The SMS 331 Program Storage Module, or any TTL compatible memory, may be used for program storage.

I/O Devices Interface

An 8-bit I/O bus, called the Interface Vector (IV) data bus, is used by the Interpreter to communicate with two sets of I/O devices. The complementary $\overline{LB}$ and $\overline{RB}$ signals identify which set of I/O devices is selected.

Both I/O and I/O address information can be output on the IV bus. The SC and WC signals are typically used to distinguish between I/O data and I/O address information as follows:

SC	WC	
1	0	I/O address is being output on IV bus
0	1	I/O data is being output on IV bus
0	0	I/O data is expected on the IV bus, as input to the Interpreter
1	1	Not generated by the Interpreter

The SMS 322-02W Working Storage Module, and the SMS 360 IV byte, may be attached to the IV bus without any additional circuitry.

INTERPRETER ARCHITECTURE AND OPERATION SUMMARY

Figure 2 provides a diagram of Interpreter internal architecture, and Table 2 summarizes Interpreter registers.

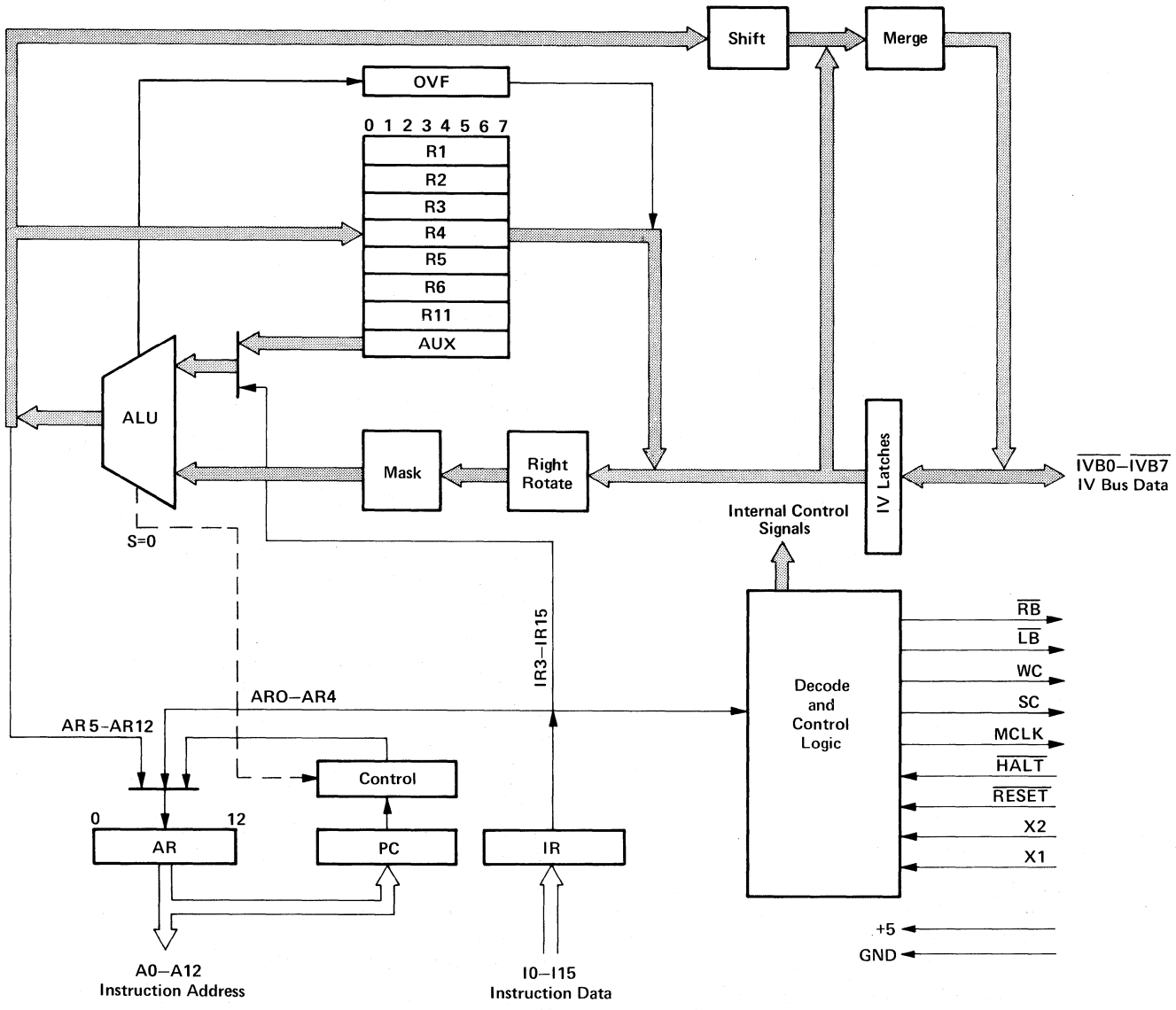


Figure 2
Interpreter Architecture

Table 2
Interpreter Internal Registers

Programmable Registers (all 8 bits):

AUX	–	General working register. Contains second term for arithmetic or logical operations.
R1	–	General working register.
R2	–	General working register.
R3	–	General working register.
R4	–	General working register.
R5	–	General working register.
R6	–	General working register.
R11	–	General working register.
OVF	–	The least-significant bit of this register is used to reflect overflow status resulting from the ADD operation (see Instruction Set Summary).

Other Registers:

Address Register (AR)	– A 13-bit register containing the address of the current instruction.
Program Counter (PC)	– Normally contains the address of the current instruction and is incremented to obtain the next instruction address.
Instruction Register (IR)	– Holds the 16-bit instruction word currently being executed.

Instruction Cycle

Each interpreter operation is executed in one instruction cycle, which may be as short as 300 ns, when other SMS components are used in the system. The Interpreter generates MCLK to synchronize external logic to the instruction cycle. Instruction cycles are subdivided into quarter cycles. MCLK is an output during the last quarter cycle.

During the third quarter cycle of an instruction, an address is output on A0-A12, identifying the location in program storage of the next instruction word. This instruction word defines the next instruction, and must be input on I0-I15 during the first quarter cycle of the next instruction cycle (see Figure 3).

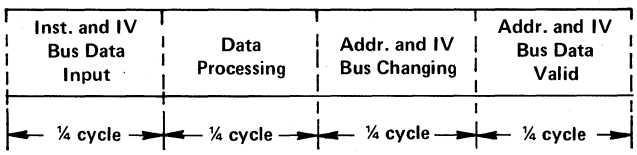
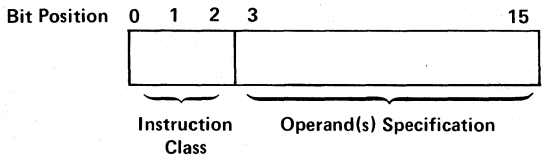


Figure 3
Instruction Cycle

Instruction Set Summary

The 16-bit instruction word input on I0-I15 is decoded by the instruction decode logic to implement events that are to occur during the remainder of the instruction cycle. Generally the 16-bit instruction word is decoded as follows:



A detailed usage of the 13 "operand(s) specification" bits is given in following sections.

Three operation code bits allow for eight instruction classes. The eight instruction classes are summarized in Table 3. Each entry is referred to as an "instruction class" because the unique architecture of the Interpreter allows a number of powerful variations to be specified by the thirteen operand(s) specification bits. A complete description of instruction formats and some instruction examples are provided in Appendix A.

Data Processing

The Interpreter architecture includes eight 8-bit working registers, an arithmetic logic unit (ALU), an overflow register, and the 8-bit IV data bus. Internal 8-bit data paths connect the registers and IV data bus to the ALU inputs, and the ALU output to the registers and IV data bus. Data processing logic is distributed along these internal 8-bit data paths. Rotate and mask logic precedes the ALU on the data entry path. Shift and merge logic follows the ALU on the data output path. All four sets of logic can operate on eight data bits in a single instruction cycle. (See Figure 2)

When less than eight bits of data are specified for output to the IV bus by the ALU, the data field (shifted if necessary) is inserted into the prior contents of the IV bus latches. The IV bus latches contain data input at the start of an instruction. This data in the IV bus latches will be specified in the instruction as a) IV bus source data or b) data from an automatic read when the IV bus is specified as a destination. Therefore, IV bus bit positions outside an inserted bit field are unmodified.

Data Addressing

Sources and destinations of data are specified using a 5-bit octal number, as shown in Table 4. The source and/or destination of data to be operated upon is specified in a single instruction word.

Referring to Table 2, the Auxiliary register (address 00) is the implied source of the second argument for ADD, AND or XOR operations.

IVL and IVR are write-only registers used only as a destination. They have addresses and are treated as registers, but in reality they do not exist. When IVL is specified as a destination of the D field = $20\text{-}27_8$, then $\overline{LB}$ = 'low', $\overline{RB}$ = 'high' are generated; when IVR is specified as a destination or the D field = $30\text{-}37_8$, then $\overline{RB}$ = 'low', $\overline{LB}$ = 'high' are generated.

When IVL or IVR is specified as the destination in an instruction, SC is also activated and data is placed on the IV bus. If IVL or IVR is specified as a source of data, the source data is all zeros.

Instruction Sequence Control

The Address Register and Program Counter are used to generate addresses for accessing an instruction. The Address Register is used to form the instruction address, and in all but three instructions (XEC, NZT, and JMP) the address is copied into the Program Counter. The instruction address is formed in one of three ways:

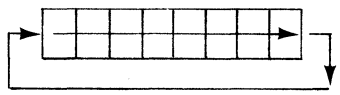
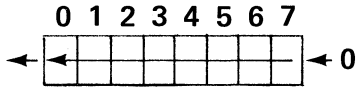
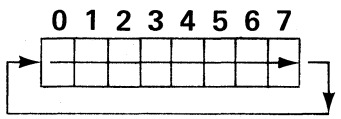
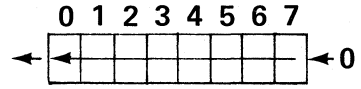
1. For all instructions but the JMP, XEC, and a satisfied NZT, the Program Counter is incremented by one and placed in the Address Register.

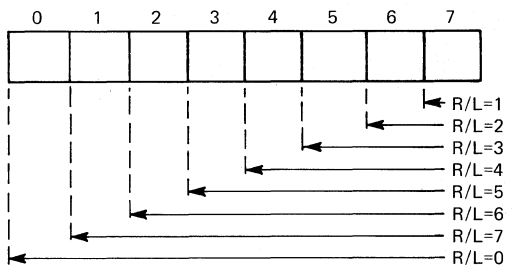
Table 3
Instruction Set Summary

INSTRUCTION MNEMONIC	OP CODE	FORMATS	DESCRIPTION	I/O CONTROL SIGNALS	INSTRUCTION CYCLE: INSTRUCTION INPUT AND DATA PROCESSING	INSTRUCTION CYCLE: ADDRESS/IV BUS OUTPUT
MOVE	0	**Register to Register** 0 2 3 7 8 10 11 15 0 | S | R/L | D S≠07,17,20-37$_8$ D≠10,20-37$_8$	(S) → D Move contents of register specified by S to register specified by D. Right rotate contents of register S by R/L places before operation.	SC= WC= LB/RB=	0 0 1 if D=17	1 if D=07,17 0 1 if D=17
		IV Bus to Register: 0 2 3 7 8 10 11 15 0 | S | R/L | D S=20-37$_8$ D≠10,20-37$_8$	Move right rotated IV bus (source) data specified by S to register specified by D. R/L specifies the length of source data with most significant bits set to zero.	SC= WC= LB/RB= LB/RB=	0 0 0 if S=20–27 1 if S=30–37	1 if D=07,17 0 1 if D=17 1 if D=17
		Register to IV Bus: 0 2 3 7 8 10 11 15 0 | S | R/L | D S≠07,17,20-37$_8$ D=20-37$_8$	Move contents of register specified by S to the IV bus. Before placement on IV bus, data is shifted as specified by D, and R/L bits merged with existing IV bus data.	SC= WC= LB/RB= LB/RB=	0 0 X X	0 1 0 if D=20–27 1 if D=30–37
		IV Bus to IV Bus: 0 2 3 7 8 10 11 15 0 | S | R/L | D S=20-37$_8$ D=20-37$_8$	Move right rotated IV bus data (source) specified by S to the IV bus. Before placement on IV bus, data is shifted or specified by D and R/L specifies the length of source data and of destination data merged with existing IV bus data.	SC= WC= LB/RB= LB/RB=	0 0 0 if S=20–27 1 if S=30–37	0 1 0 if D=30–37 1 if D=30–37
ADD	1	**SAME AS MOVE**	(S) plus (AUX) → D Same as MOVE but contents of AUX ADDed to the source data. If carry from most significant bit then OVF=1, otherwise OVF=0.		SAME AS MOVE	
AND	2	**SAME AS MOVE**	(S) △ (AUX) → D Same as MOVE but contents of AUX ADDed with source data.		SAME AS MOVE	
XOR	3	**SAME AS MOVE**	(S) + (AUX) → D Same as MOVE but contents of AUX exclusive ORed with source data.		SAME AS MOVE	
XEC	4	**Register Immediate:** 0 2 3 7 8 15 4 | S | I S≠07,17,20-37$_8$ I=000-377$_8$	Execute instruction at current page address offset by I + (S). EXECute the instruction at the address determined by catenating 5 high order bits of PC with the 8 bit sum of I and register specified by S. PC is not incremented.	SC= WC= LB/RB=	0 0 X	0 0 X
		IV Bus Immediate: 0 2 3 7 8 10 11 15 4 | S | R/L | I S=20-37$_8$ I=00-37$_8$	EXECute the instruction at the address determined by catenating 8 high order bits of PC with the 5 bit sum of I and rotated IV bus data (source) specified by S. R/L specifies length of source data with most significant bits set to zero. PC is not incremented.	SC= WC= LB/RB= LB/RB=	0 0 0 if S=20–27 1 if S=30–37	0 0 X X
NZT	5	**Register Immediate:** 0 2 3 7 8 15 5 | S | I S≠07,17,20-37$_8$ I=000-377$_8$	If (S) ≠ 0, jump to current page address offset by I; otherwise PC + 1 → PC. If contents of register specified by S is Non Zero then Transfer to address determined by catenating 5 high order bits of PC with I; otherwise increment PC.	SC= WC= LB/RB=	0 0 X	0 0 X
		IV Bus Immediate: 0 2 3 7 8 10 11 15 5 | S | R/L | I S=20-37$_8$ I=00-37$_8$	If right rotated IV bus data (source) is Non Zero then Transfer to address determined by catenating 8 high order bits of PC with I; otherwise increment PC.	SC= WC= LB/RB= LB/RB=	0 0 0 if S=20–27 1 if S=30–37	0 0 X X
XMIT	6	**Register Immediate:** 0 2 3 7 8 15 6 | D | I S≠20-37$_8$ I=000-377$_8$	Transmit I → S TRANSMIT and store 8 bit binary pattern I to register specified by D.	SC= WC= LB/RB=	0 0 X	1 if D=07,17 0 1 if D=17
		IV Bus Immediate: 0 2 3 7 8 10 11 15 6 | D | R/L | I S=20-37$_8$ I=00-37$_8$	TRANSMIT binary pattern I to IV bus. Before placement on IV bus, literal I is shifted as specified by D and R/L bits merged with existing IV bus data.	SC= WC= LB/RB= LB/RB=	0 0 X X	0 1 0 if S=20–27 1 if S=30–37
JMP	7	**Address Immediate:** 0 2 3 15 7 | A A=00000-17777$_8$	Jump to Program Address A. JUMP to program storage address A. A is stored in the address register (AR).	SC= WC= LB/RB=	0 0 X	0 0 X

2. For the JMP instruction, the full 13-bit address field from the JMP instruction is placed into the Address Register and copied into the Program Counter.
3. For the XEC and NZT instructions, the high order 5- or 8-bits of the Program Counter are combined with 8- or 5-lower-order bits of ALU output (XEC or NZT) and placed in the Address Register. For the NZT instruction, it is also copied into the Program Counter.

Table 4
Data Source/Destination Addresses

S and/or D Field Specification (octal)	Source/Destination
00	Auxiliary Register (AUX)
01 to 06	Working registers (R1 to R6) respectively
07	IVL write-only register (destination only)
10	Overflow status (OVF) – source only
11	Working register (R11)
17	IVR write-only register (destination only)
2N (N=0,1,2, 3,4,5,6,7)	a. If a source, IV bus data right rotated (7 – N) bits and masked (specified by R/L). LB='low' and RB='high' generated. **IV Bus Source Data** 0 1 2 3 4 5 6 7  b. If a destination, IV bus data left shifted (7 – N) bits and merged (specified by R/L). LB='low' and RB='high' generated. **IV Bus Destination Data** 0 1 2 3 4 5 6 7  0
3N (N=0,1,2, 3,4,5,6,7)	a. If a source, IV bus data right rotated (7 – N) bits and masked (specified by R/L). LB='high' and RB='low' generated. **IV Bus Source Data** 0 1 2 3 4 5 6 7  b. If a destination, IV bus data left shifted (7 – N) bits and merged (specified by R/L). LB='high' and RB='low' generated. **IV Bus Destination Data** 0 1 2 3 4 5 6 7  0



IV BUS DATA
LENGTH SPECIFICATION

SYSTEM DESIGN USING THE INTERPRETER

Designing hardware around the SMS 300 Interpreter reduces to selecting a program storage device (ROM, PROM, etc.), selecting I/O devices (IV BYTE, MULTIPLEXORS, RAM, etc.), selecting clock mode (system driven or crystal controlled) and interfacing the Interpreter to these components, as shown in Figure 4.

System Clock

The Interpreter has an integrated oscillator which generates all necessary clock signals. The oscillator is designed to connect directly to a series resonant quartz crystal via pins X1 and X2. The crystal resonant frequency, f, is related to the desired cycle time, T, by the relationship f=2/T. For a 300 ns system, f=6.667 MHz.

In lower speed applications where the cycle time need not be precisely controlled, a capacitor may be connected between X1 and X2 to drive the oscillator. If cycle time is to be varied, X1 and X2 should be driven from complementary outputs of a pulse generator. Figure 5 shows a typical configuration. For systems where the Interpreter is to be driven from a master clock, the X1 and X2 lines may be interfaced to TTL logic as shown in Figure 6.

Crystal characteristics

Type:	Fundamental mode, series resonant
Impedance at Fundamental:	35 ohms maximum
Impedance at harmonics and spurs:	50 ohms minimum

Halt, reset signals

$\overline{\text{HALT}}$:

A low level at the $\overline{\text{HALT}}$ input causes the Interpreter to stop processing after completion of the current instruction (end of quarter cycle when MCLK is high). $\overline{\text{HALT}}$ does not inhibit MCLK or affect any internal registers. Normal operation begins with the next complete instruction cycle after the $\overline{\text{HALT}}$ input goes high.

$\overline{\text{RESET}}$:

A low level at the $\overline{\text{RESET}}$ input sets the program counter and address register to zero and inhibits MCLK. $\overline{\text{RESET}}$ must be applied for at least one full instruction cycle to insure both registers are cleared. MCLK occurs on first instruction cycle and normal operation begins with the second instruction cycle after the $\overline{\text{RESET}}$ input goes high.

Example:

A specific example of a MicroController system, using the SMS 300 Interpreter — four SMS 360 IV Bytes, and two SMS 8205 ROMs is shown in Figure 4. Only eight components are required to build this MicroController system which contains 512 words of program storage, 32 TTL I/O connection points, and operates at a 300-ns instruction cycle time.

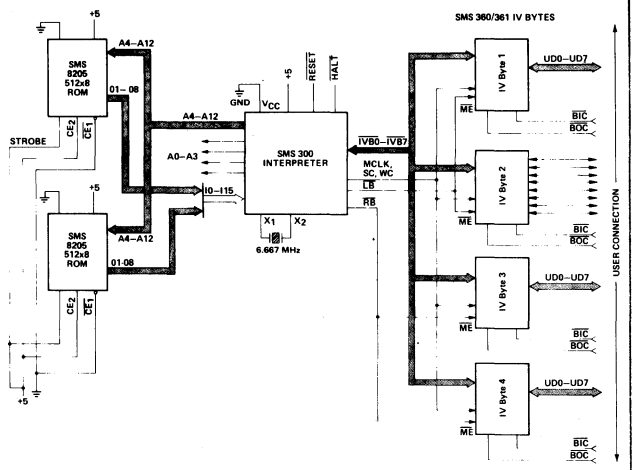


Figure 4
Example of MicroController System

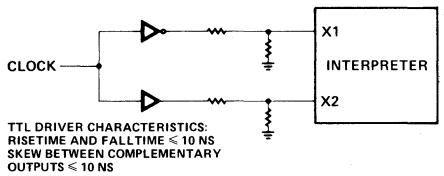


Figure 6
Clocking with TTL

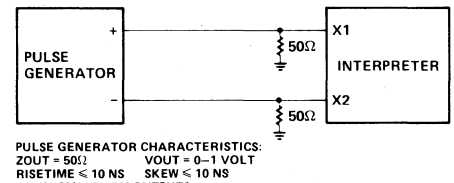


Figure 5
Clocking with a Pulse Generator

System Timing

The system instruction cycle time is determined by program storage access time, I/O register data/control delays, and Interpreter propagation delays. Instruction cycle time is normally constrained by two major propagation delay paths:

I. Program storage access time + instruction to address stable delay **or**

   Program storage access time + I/O control input delay + I/O device access time, + IV Bus to address stable delay.

II. I/O control input delay + I/O device access time.

These propagation path delay times must be consistent with the Interpreter internal clock times.

Interpreter internal clock intervals occur every quarter cycle of a complete instruction cycle. The Interpreter output MCLK is high during the last quarter cycle of every instruction cycle. Interpreter input operations (instruction data, IV Bus data) occur during the first two quarter cycles (INPUT PHASE). Interpreter output operations (address, IV Bus data) occur during the last two quarter cycles (OUTPUT PHASE). Figure 7 illustrates typical timing waveforms for an instruction cycle. Interpreter propagation delays are shown in Figure 7 and quantified in Table 8.

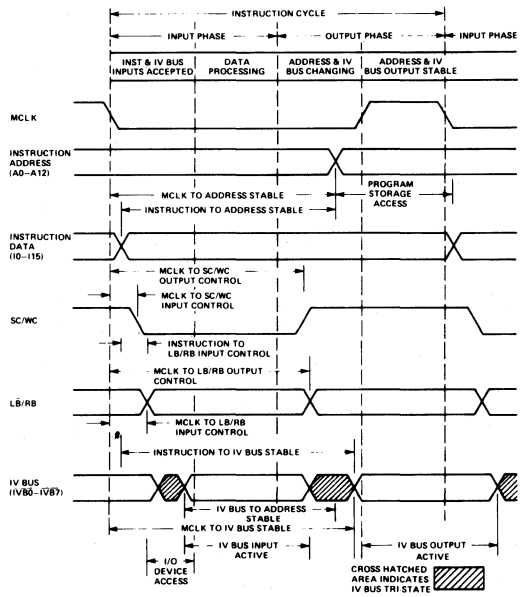


Figure 7
Typical Instruction Cycle Timing

Propagation path II delay time must be less than one quarter cycle. Interpreter delay times shown in Table 8 which are applicable are: MCLK TO SC/WC INPUT CONTROL and MCLK TO LB/RB INPUT CONTROL. These delays occur during the first part of the first quarter cycle as shown in Figure 7 and correspond to I/O control input delays.

The maximum I/O device access time is the difference between one quarter cycle time and the I/O control input delay. Using the values from Table 8, the required I/O device access time is determined by the following equations:

35ns + I/O device access $\leqslant$ ¼(cycle time) EQ1

25ns + I/O device access $\leqslant$ ¼(cycle time) EQ2

For a 300-ns instruction cycle time, I/O device access times must be less than 40ns and 50ns respectively. The SMS 360/361 IV Byte is an I/O register which satisfies the I/O device access time constraints.

Propagation path I determines the allowable program storage access time for a given instruction cycle time. The program storage access time is the smaller of these two equations:

Program storage access $\leqslant$ cycle time − instruction to address stable EQ3

Program storage access $\leqslant$ cycle time − (I/O device access + I/O control input delay + IV Bus to address stable delay) EQ4

Therefore, a cycle time of 300ns requires a program storage access time of 85ns or less.

Tradeoffs can be made between I/O device access time and program storage access time. If the I/O device access and program storage access times are less than the limits determined in equations EQ1, EQ2 and EQ3, then EQ4 can be used to trade I/O device and program storage access times.

Propagation delays during the OUTPUT PHASE usually do not limit instruction cycle times. MCLK is normally used to control data entry into I/O devices on the IV Bus during the last two quarter cycles. The user must insure that data set-up time requirements of I/O devices are satisfied. Data output on the IV Bus will be stable for the duration of MCLK if the I/O device access time and instruction cycle time satisfy the following equation:

I/O device access + I/O control input delay + ½(cycle time) $\leqslant$ MCLK to IV Bus stable delay.

If the above inequality is not satisfied, the IV Bus data may be changing during MCLK.

SPECIFICATIONS

Interpreter specifications are given in Tables 5 and 6, and Figure 8.

Absolute Maximum Ratings

Supply Voltage V_{CC} 7V
Logic Input Voltage. 5.5V
Crystal Input Voltage. 2V

Table 5
AC Electrical Characteristics

Delay Description	Propagation Delay Limit	Cycle Time Limit
X2 falling edge to MCLK falling edge		
MCLK to SC/WC input control	25ns	
MCLK to SC/WC output control *		.5 (CYC) + 25ns
INSTRUCTION to LB/RB input control	35ns	
MCLK to LB/RB input control	35ns	
MCLK to LB/RB output control		.5 (CYC) + 35ns
INSTRUCTION to IV BUS stable	225ns	
MCLK to IV BUS stable	225ns	
IV BUS input stable to IV BUS output stable	150ns	
INSTRUCTION to ADDRESS stable	215ns	
MCLK to ADDRESS stable	215ns	.5 (CYC) + 40ns
IV BUS to ADDRESS stable	140ns	

MCLK falling edge to HALT falling edge	¼(CYC) − 40ns (max)
MCLK falling edge to HALT rising edge	¼(CYC) − 40ns
MCLK falling edge to RESET falling edge	½(CYC) (max)
RESET rising edge to first MCLK	0 to 1 CYC (max)

Table 6
DC Electrical Characteristics

Parameter	Symbol	Conditions	Limits Min.	Limits Typ.	Limits Max.	Units
High-level Input Voltage	V_{IH}					
X1, X2			.6			V
All others			2			V
Low-level Input Voltage	V_{IL}					
X1, X2					.4	V
All others					.8	V
Input Clamp Voltage (Note 1)	V_{CL}	V_{CC} = 4.75V, I_I = −10mA			−1.5	V
High-level Input Current	I_{IH}					
X1, X2		V_{CC} = 5.25V, V_{IH} = .6V		2700		μA
All others		V_{CC} = 5.25V, V_{IH} = 4.5V		<1	50	μA
Low-level Input Current	I_{IL}					
X1, X2		V_{CC} = 5.25V, V_{IL} = .4V		−2500		μA
$\overline{IVB0}$–7		V_{CC} = 5.25V, V_{IL} = .4V		−140	−200	μA
I0–I15		V_{CC} = 5.25V, V_{IL} = .4V		−880	−1600	μA
$\overline{HALT}$, $\overline{RESET}$		V_{CC} = 5.25V, V_{IL} = .4V		−230	−400	μA
Low-level Output Voltage	V_{OL}					
A0–A12		V_{CC} = 4.75V, I_{OL} = 4.25mA		.35	.55	V
All others		V_{CC} = 4.75V, I_{OL} = 16mA		.35	.55	V
High-level Output Voltage	V_{OH}	V_{CC} = 4.75V, I_{OH} = 3mA	2.4			V
Short Circuit Output Current (Note 2)	I_{OS}	V_{CC} = 5.25V	−30		−140	mA
Supply Voltage	V_{CC}		4.75	5	5.25	V
Supply Current	I_{CC}	V_{CC} = 5.25V		300	450	mA

NOTES:

1. Crystal inputs X1 and X2 do not have clamp diodes.
2. Only one output may be grounded at a time.

(Limits apply for V_{CC} = 5V ± 5% and $0^oC < T_A < 70^oC$ unless specificed otherwise.)

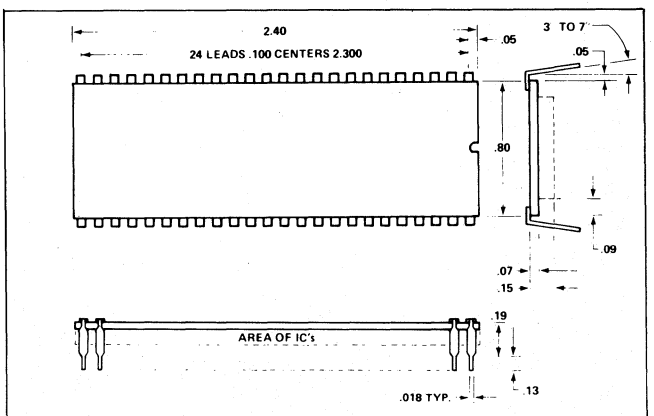


Figure 8
Interpreter Package Dimensions