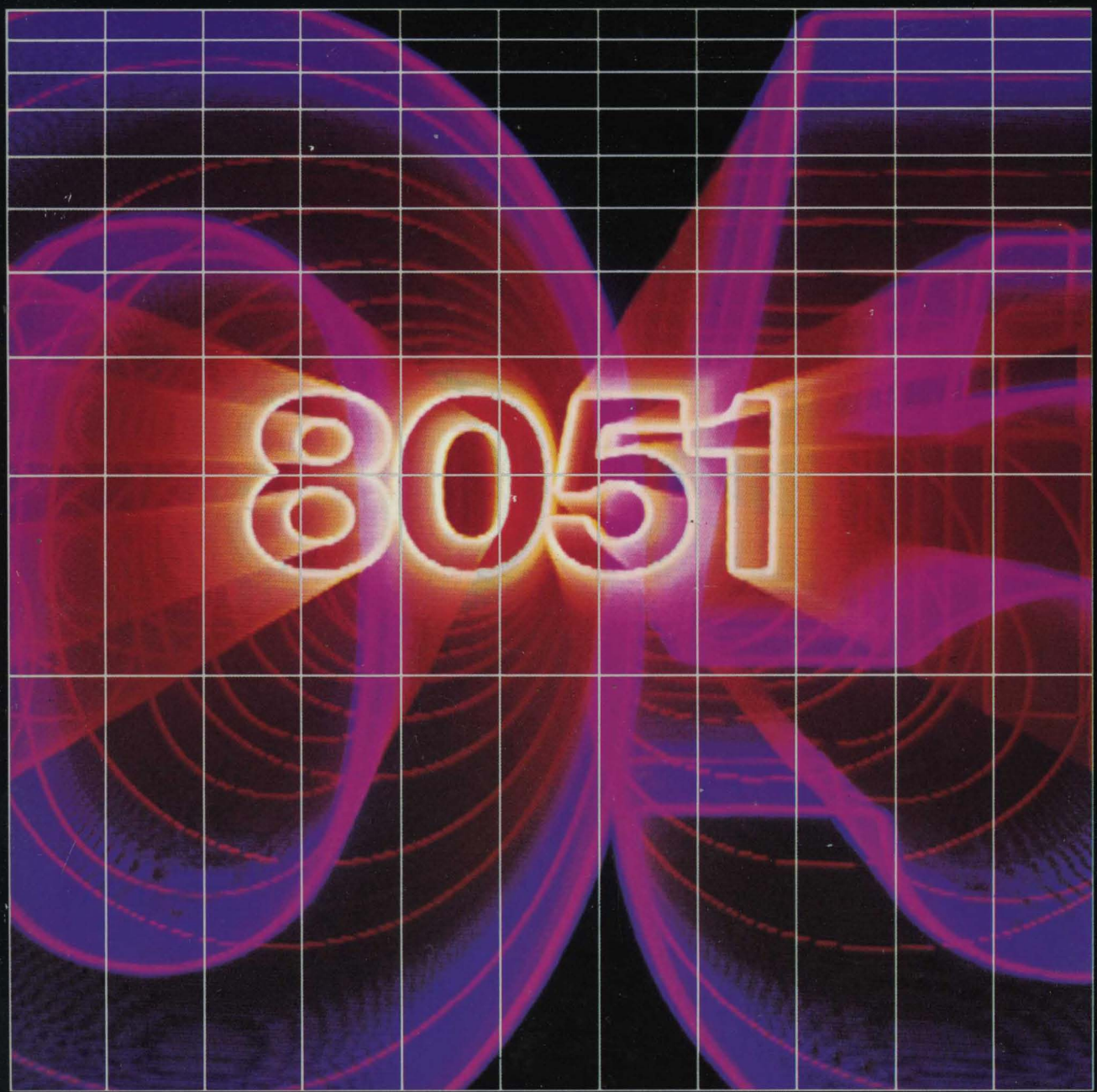
intel®

MCS-51™ Family of Single Chip Microcomputers User's Manual



Part No. 121517-001

intel®

MCS-51® FAMILY OF SINGLE CHIP MICROCOMPUTERS USER'S MANUAL

JANUARY 1981

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Table of Contents

AFN-01739A

CHAPTER 7 (Cont.)

Introduction

1

CHAPTER 1
INTRODUCTION

MANUAL ORGANIZATION

This publication describes Intel's 8051 family of single chip microcomputers. It is written for engineers, technicians and students who understand basic microcomputer operating principles. The manual provides a comprehensive reference guide for the 8051 family.

The manual organization is by chapters and appendices. Chapter 1 provides a brief introduction to the MCS-51® family, its software tools, and its development support in the most general terms.

The second chapter provides a functional description of the 8051 family hardware.

The third chapter describes the 8051's family memory organization, addressing modes, data types and its instruction set.

The fourth chapter is an expanded 8051 family configured system using peripherals and external program and external data memories.

Chapter 5 lists software routines that handle programming applications such as radix conversions and stack manipulations.

The chapter following, the sixth, is the specification section for the 8051 family. A.C. and D.C. characteristics of the 8051 family are located here.

Chapter 7 provides data sheets on Intel products that can be used in conjunction with the 8051.

The next chapter describes the development support available for the 8051 family.

That's the end of the chapters, but there are two appendices. Appendix A is a PL/M-80 description of the 8051 instruction set. Appendix B is the AP Note section where AP-69 and AP-70 are reproduced in their entirety. AP-69 is titled, "An Introduction to the Intel® MCS-51® Single-Chip Microcomputer Family" and AP-70 is titled, "Using the Intel® MCS-51® Boolean Processing Capabilities."

PRODUCT OVERVIEW

Introduced in May, 1980, Intel's 8051 family of single-chip microcomputers is the next generation microcomputer for the controller marketplace. With an expandable and flexible architecture the 8051 family provides high performance for the applications of the future.

8051 Family

The 8051 family has three members: The 8031, 8051, and 8751. The 8031 is a CPU-only device, the 8051 has 4K bytes of factory-masked ROM and the 8751 has 4K bytes of EPROM. The generic term "8051" is used to refer collectively to the 8031, 8051 and 8751.

Supporting the applications of the '80s is the target of the 8051 family. On a single die the 8051 microcomputer combines CPU; non-volatile 4K x 8 read-only program memory (8051 and 8751); volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/event counters; a five-source, two-priority-level, nested interrupt structure; serial I/O channel for either multiprocessor communications, I/O expansion, or full duplex UART; and on-chip oscillator and clock circuits.

The CPU is in a 40-pin package and uses a single 5V power supply. Intel's HMOS process technology is the driving force behind the 8051's ability to bring many peripheral functions on-board a single-chip microcomputer.

Along with 4K program memory on-board, the 8051 can address another 60K of external program memory. If external data memory is needed the 8051 can also address 64K of external RAM. When an 8031 is used, all program memory execution is external and it, too, can address 64K of both external program and data memory. The 8051 contains many features for ease of manipulating variables in Internal Data Memory. The stack may be located anywhere within the internal RAM space. There are 4 banks of registers (eight registers in each bank) that facilitate context switching and provide byte efficiency. Also within the Internal Data Memory are the Special Function Registers. These are memory-mapped locations for the ports, arithmetic registers, control and status registers and the timer/counters.

Within the Internal Data Memory are 256 individually addressable bits. 128 bits are located in the Internal Data RAM and the second 128 bits are located in the Special Function Register. These 256 addressable bits provide a new dimension in controller applications. The programmer/designer may now manipulate individual bits with specific bit instructions! This new feature is the Boolean Processor, which is actually a bit processor with its own accumulator, I/O and instruction set.

To increase programming ease the 8051 added new addressing modes. Direct Addressing was added to manipulate variables within the 128-byte Data RAM

and the Special Function Registers. Base-Register plus Index-Register Indirect Addressing gives the programmer the ability to easily manipulate constants in program memory for table look-ups. Table look-ups are supported over the entire 64K range by using 16-bit registers.

A more in-depth explanation of the 8051's memory spaces, addressing modes and instruction set is provided in Chapter 3.

The 8051 hardware and on-chip peripherals have been briefly described a few paragraphs earlier. Let's add a little more detail in the following paragraphs. For complete operational and functional explanation of the following features please read Chapter 2.

MCS-51 family has two independent, 16-bit timer/counters. Under software control, each timer/counter may be placed in one of four modes:

0) An 8-bit timer/counter with a divide-by-32 prescaler
1) A 16-bit timer/counter
2) An auto-reload 8-bit timer/counter
3) A mode which provides the programmer/designer with two timers and a counter for added flexibility

There is a five-source interrupt system. The user has software control over enabling/disabling the interrupts and assigning priority levels to the interrupt sources. The two external interrupts, under software control, can be either transition or low-level activated.

Two of the three internal interrupt sources generate an interrupt request when the overflow of the timer/counters occurs. The third interrupt source generates an interrupt request from the serial channel when the receiver register is full or when the transmitter register is empty.

Serial data communication is accomplished by the 8051's serial channel. Operation is flexible by providing user-programmable baud rates, two choices of frame size and multiprocessor communications. There are four modes of operation:

0) An 8-bit frame, synchronous mode which allows I/O expansion using shift registers. A clock output is provided by the 8051.
1) An 8-bit frame, asynchronous mode handling programmable baud rates from 122 to 31,250 bits per second (12MHz operation).
2) A 9-bit frame, asynchronous mode that has a fixed baud rate of 187.5K bits per second (12MHz operation)
3) A programmable baud rate from 122 to 31,250 bits per second (12MHz operation) for a 9-bit frame and asynchronous operation.

Since microcontrollers are real-time oriented, the 8051 has four 8-bit ports for interfacing to the external world. Three of the four ports, under software control, can have additional capabilities. Port 0 is a time multiplexed bus. It outputs the lower 8 bits of the 16-bit address and also receives data and instruction during an external RAM read or external program memory operation. Port 0 also outputs the data when a write operation is performed. Port 2 emits the upper 8-bits of the 16-bit address when external memories are being accessed. Port 3 contains the special control signals such as the read and write strobes, the two external interrupt inputs, the two counter inputs and the transmit and receive pins of the serial channel. Port 1 is strictly an 8-bit quasi-bidirectional port.

Development Support Tools

Intel provides the tools needed for efficient, low risk development of products using the 8051 family. These development tools are based on the Intellec® Series II Microcomputer Development System. The Intellec system runs ISIS-II, a disk-based operating system being used in thousands of customer installations. This same hardware and operating system can also be used to develop systems based on other Intel microprocessor families such as the iAPX 86, iAPX 88, 8085 and 8048.

ASM-51, the 8051 macro assembler provides assembly language programming for the 8051. Symbolic references (i.e., names) to 8051 hardware features are supported.

The Universal PROM Programmer can program any of Intel's PROM memories. To program and verify the 8751, an 8751 personality card adapter should be used. Programming and verification operations are initiated from the Intellec development system console and are controlled by the Universal PROM Mapper program.

The SDK-51, System Design Kit, is an 8031-based prototyping and evaluation kit. It includes the CPU, RAM, I/O ports and a breadboard area for interfacing to customer circuits. A ROM-based monitor program, single-line assembler and disassembler are supplied with the kit. Monitor commands may be entered from an on-board keypad or from a terminal. Monitor commands allow programs to be entered, run, stopped and single-stepped; memory contents can be altered as well as displayed.

The ICE-51™ In-Circuit Emulator provides real-time

symbolic debugging support for the 8051 microcomputer. A 40-pin probe replaces the 8051 in the system under test. This probe is connected to a specifically configured 8051 "bond-out" chip which makes internal 8051 hardware available to ICE-51 circuitry. The ICE-51 module emulates the 8051 in the system under test in response to commands entered through the Intellec console. These commands allow the user to debug the system by setting breakpoints, tracing the flow of execution, single-stepping, examining and altering memory and I/O, etc.

All references to program variables and labels are symbolic (i.e., their ASM-51 names). Software testing can also map "system under test" memory into the full speed ICE-51 memory to permit software testing to begin before prototype hardware has been developed.

Functional Description

2

CHAPTER 2
FUNCTIONAL DESCRIPTION

This chapter explains the functions of the 8051. The first part begins with a brief description of the memory spaces and addressing modes. (For more in-depth information, please see Chapter 3.) The chapter then explains the hardware registers, the ALU, and Boolean Processor.

The second part of Chapter 2 explains in detail the workings of the on-chip peripherals: the interrupt system, the I/O pins, the timer/counters and the serial channel.

8051 CPU ARCHITECTURE

The 8051 CPU manipulates operands in four memory spaces. These are the 64K-byte Program Memory, 64K-byte External Data Memory, 384-byte Internal Data Memory and 16-bit Program Counter spaces. The Internal Data Memory address space is further divided into the 256-byte Internal Data RAM and 128-byte Special Function Register (SFR) address spaces shown in Figure 2-1. Four Register Banks (each bank has eight registers), 128 addressable bits, and the stack reside in the Internal Data RAM. The stack depth is limited only by the available Internal Data RAM. Its location is determined by the 8-bit Stack Pointer. All registers except the Program Counter and the four 8-Register Banks reside in the Special Function Register address space. These memory mapped registers include arithmetic registers, pointers, I/O ports, and registers for the interrupt system, timers and serial channel. 128 bit locations in the SFR address space are addressable as bits. The 8051 currently contains 128 bytes of Internal Data RAM and 20 Special Function Registers.

Conditional branches are performed relative to the Program Counter. The register-indirect jump permits branching relative to a 16-bit base register with an offset provided by an 8-bit index register. Sixteen-bit jumps and calls permit branching to any location in the contiguous 64K Program Memory address space.

The 8051 has five methods for addressing source operands: Register, Direct, Register-Indirect, Immediate, and Base-Register-plus Index Register-Indirect Addressing.

The first three methods can be used for addressing destination operands. Most instructions have a "destination, source" field that specifies the data type, addressing methods and operands involved. For operations other than moves, the destination operand is also a source operand.

Registers in the four 8-Register Banks can be accessed through Register, Direct, or Register-Indirect Addressing; the 128 bytes of Internal Data RAM through Direct or Register-Indirect Addressing; and the Special Function Registers through Direct Addressing. External Data Memory is accessed through Register-Indirect Addressing. Look-up Tables resident in Program memory can be accessed through Base Register-plus Index Register-Indirect Addressing.

The 8051 is classified as an 8-bit machine since the internal ROM, RAM, Special Function Registers, Arithmetic/Logic Unit and external data bus are each 8-bits wide. The 8051 performs operations on bit, nibble, byte and double-byte data types.

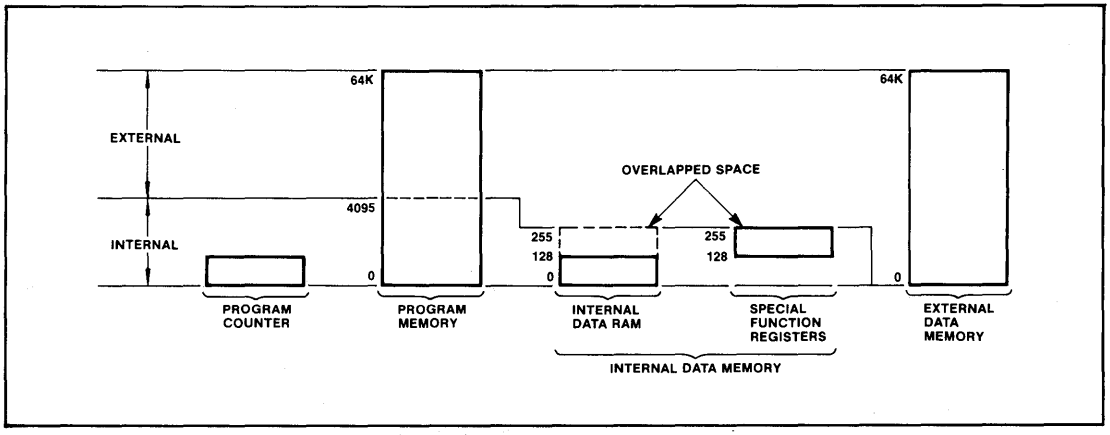


Figure 2-1. 8051 Family Memory Organization

The 8051 has extensive facilities for byte transfer, logic, and integer arithmetic operations. It excels at bit handling since data transfer, logic and conditional branch operations can be performed directly on Boolean variables.

The 8051's instruction set is an enhancement of the instruction set familiar to MCS-48 users. It is enhanced to allow expansion of on-chip CPU peripherals and to optimize byte efficiency and execution speed. Op codes were reassigned to add new high-power operations and to permit new addressing modes which make the old operations more orthogonal. Efficient use of program memory results from an instruction set consisting of 49 single-byte, 45 two-byte and 17 three-byte instructions. When using a 12MHz crystal, 64 instructions execute in 1 μs and 45 instructions execute in 2 μs. The remaining instructions (multiply and divide) require only 4 μs. The number of bytes in each instruction and the number of oscillator periods required for execution are listed in the Instruction Set in Chapter 3 in Table 3-2.

CPU HARDWARE

This section describes the hardware architecture of the 8051's CPU in detail. The interrupt system and on-chip functions peripheral to the CPU are described in subsequent sections. A detailed 8051 Functional Block Diagram is displayed in Figure 2-2.

Instruction Decoder

Each program instruction is decoded by the instruction decoder. This unit generates the internal signals that control the functions of each unit within the CPU section. These signals control the sources and destination of data, as well as the function of the Arithmetic/Logic Unit (ALU).

Program Counter

The 16-bit Program Counter (PC) controls the sequence in which the instructions stored in program memory are executed. It is manipulated with the Control Transfer instructions listed in Chapter 3.

Internal Program Memory

The 8051/8751 have 4K bytes of program memory resident on-chip.

Internal Data Memory

The 8051 contains a 128-byte Internal Data RAM (which includes registers R7-R0 in each of four Banks), and twenty memory-mapped Special Function Register.

INTERNAL DATA RAM

The Internal Data RAM provides a convenient 128-byte scratch pad memory.

Register Banks

There are four Register Banks within the Internal Data RAM. Each Register Bank contains registers R7-R0.

128 Addressable Bits

There are 128 addressable software flags in the Internal Data RAM. They are located in the 16 byte locations starting at byte address 32 and ending with byte location 47 of the RAM address space.

Stack

The stack may be located anywhere within the Internal Data RAM address space. The stack may be as large as 128 bytes on the 8051.

SPECIAL FUNCTION REGISTERS

The Special Function Registers include arithmetic registers (A, B, PSW), pointers (SP, DPH, DPL) and registers that provide an interface between the CPU and the on-chip peripheral functions. There are also 128 addressable bits within the Special Function Registers. The memory-mapped locations of these registers and bits are discussed in Chapter 3.

A Register

The A register is the accumulator.

B Register

The B register is dedicated during multiply and divide and serves as both a source and a destination. During all other operations the B register is simply another location of the Special Function Register space.

Program Status Word Register

The carry (CY), auxiliary carry (AC), user flag 0 (F0), register bank select (RS0 and RS1), overflow (OV) and parity (P) flags reside in the Program Status Word (PSW) Register. These flags are bit-memory-mapped within the byte-memory-mapped PSW. The PSW flags record processor status information and control the operation of the processor.

The CY, AC, and OV flags generally reflect the status of the latest arithmetic operations. The P flag always reflects the parity of the A register. The carry flag is also the Boolean accumulator for bit operations. Specific details on these flags are provided in the Arithmetic Flags section of Chapter 3. F0 is a general purpose flag which is pushed onto the stack as part of a PSW save. The two Register Bank select bits (RS1 and RS0) determine which one of the four Register Banks is selected.

Stack Pointer

The 8-bit Stack Pointer (SP) contains the address at which

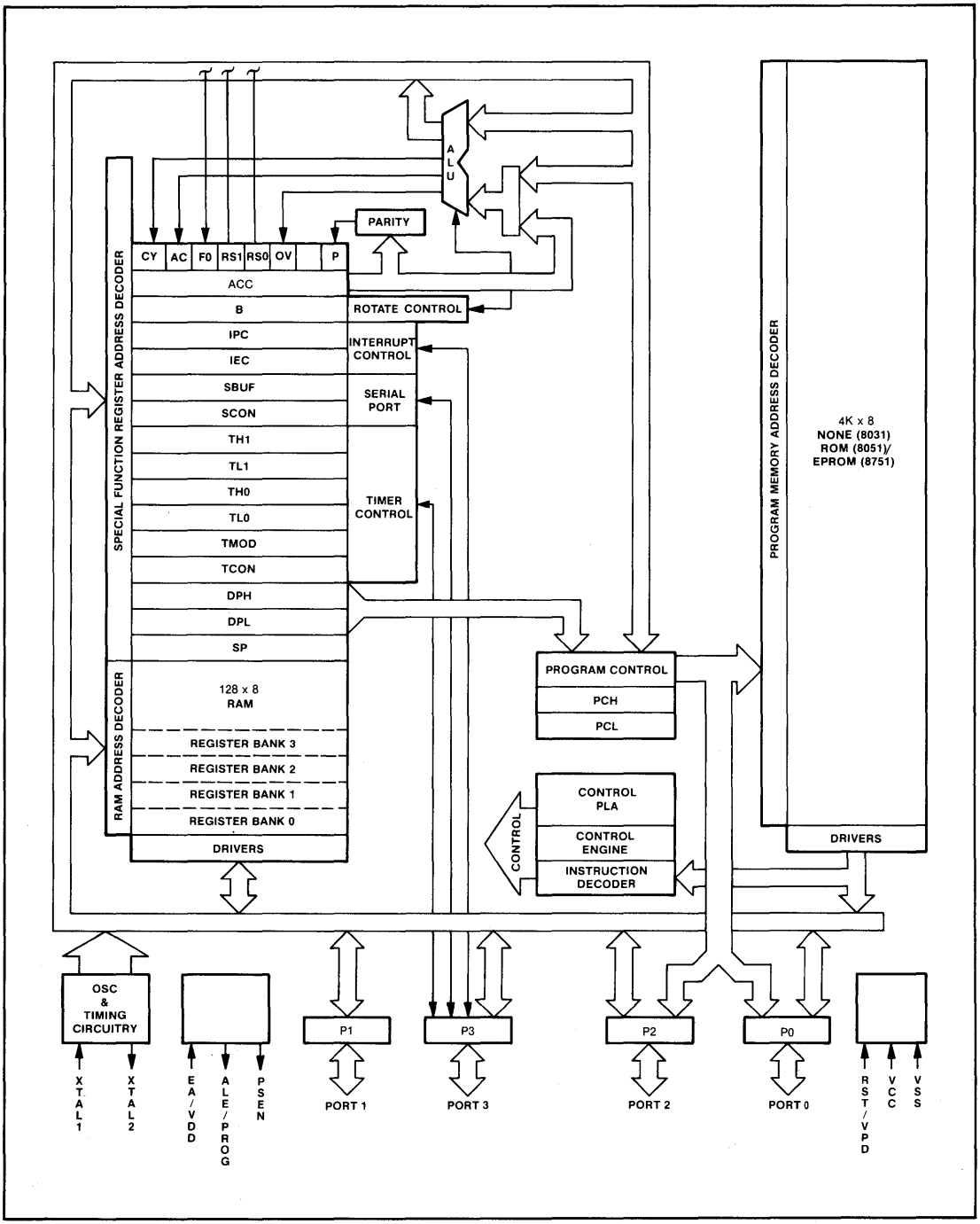


Figure 2-2. 8051 Family Functional Block Diagram

AFN-01739A

the last byte was pushed onto the stack. This is also the address of the next byte that will be popped. The SP is incremented during a push. SP can be read or written to under software control.

Data Pointer (High) and Data Pointer (Low)

The 16-bit Data Pointer (DPTR) register is the concatenation of registers DPH (data pointer's high-order byte) and DPL (data pointer's low-order byte). The DPTR is used in Register-Indirect Addressing to move Program Memory constants, to move External Data Memory variables, and to branch over the 64K Program Memory address space.

Port 3, Port 2, Port 1, Port 0

The four ports provide 32 I/O lines to interface to the external world. All four ports are both byte and bit addressable. The 8051 also allows memory expansion using Port 0 (P0) and Port 2 (P2) while Port 3 (P3) contains special control signals such as the read and write strobes. Port 1 (P1) is used for I/O only.

Interrupt Priority Register

The Interrupt Priority (IPC) register contains the control bits to set an interrupt to a desired level. A bit set to a one gives the particular interrupt a high priority listing.

Interrupt Enable Register

The Interrupt Enable (IEC) register stores the enable bits for each of the five interrupt sources. Also included is a global enable/disable bit of the interrupt system.

Timer/Counter Mode Register

Within the Timer Mode (TMOD) register are the bits that select which operations each timer/counter will do.

Timer/Counter Control Register

The timer/counters are controlled by the Timer/Counter Control (TCON) register bits. The start/stop bits for the timer/counters along with the overflow and interrupt request flags are mapped in TCON.

Timer/Counter 1 (High), Timer/Counter 1 (Low), Timer/Counter 0 (High), Timer/Counter 0 (Low)

There are four register locations for the two 16-bit timer/counters. These registers can be read or written to, to give the programmer easy access to the timer/counters. TH1 and TH0 refer to the 8 high-order bits of timer/counter 1 and 0, respectively. TL1 and TL0 refer to the low-order bits of both timer/counter 1 and 0.

Serial Control Register

This control register (SCON) has bits that enable reception of the serial port. Selecting the operating mode of the serial port is accomplished with bits in this register also.

Serial Data Buffer

The Serial Data Buffer (SBUF) register is used to hold serial port input or output data depending on whether the serial port is receiving or transmitting data.

ARITHMETIC SECTION

The arithmetic section of the processor performs many data manipulation functions and is comprised of the Arithmetic/Logic Unit (ALU), A register, B register and PSW register. The ALU accepts 8-bit data words from one or two sources and generates an 8-bit result under the control of the instruction decoder. The ALU performs the arithmetic operations of add, subtract, multiply, divide, increment, decrement, BCD-decimal-add-adjust and compare, and the logic operations of and, or, exclusive-or, complement and rotate (right, left, or nibble swap (left four)).

PROGRAM CONTROL SECTION

The program control section controls the sequence in which the instructions stored in program memory are executed. The conditional branch logic enables conditions internal and external to the processor to cause a change in the sequence of program execution.

OSCILLATOR AND TIMING CIRCUITRY

Timing generation for the 8051 is completely self-contained, except for the frequency reference which can be a crystal or external clock source. The on-board oscillator is a parallel anti-resonant circuit with a frequency range of 1.2MHz to 12MHz. There is a divide-by-12 internal timing which gives the 8051 a minimum instruction cycle of 1 μsec. with a 12MHz crystal. The XTAL2 pin is the output of a high-gain amplifier, while XTAL1 is its input. A crystal connected between XTAL1 and XTAL2 provides the feedback and phase shift required for oscillation. If XTAL1 is being driven by an external frequency source, XTAL2 should be a no connect. The 1.2MHz to 12MHz range is also accommodated when an external clock is applied to XTAL1 as the frequency source.

BOOLEAN PROCESSOR

The Boolean Processor is an integral part of the 8051's architecture. It is an independent bit processor with its own instruction set, its own accumulator (the carry register) and its own bit-addressable RAM and I/O. The bit manipulation instructions allow the Direct Addressing of 128 bits within the Internal Data RAM and 128 bits within the Special Function Registers as shown in Figures 2-3 and 2-4. The Special Function Registers with an address evenly divisable by eight (P0, TCON, P1, SCON, P2, IEC, P3, IPC, PSW, A, and B) contain Directly Addressable bits.

a.) RAM Bit Addresses.

RAM BYTE	(MSB)							(LSB)
7FH								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH	Bank 3							
18H								
17H	Bank 2							
10H								
0FH	Bank 1							
08H								
07H	Bank 0							
00H								

Figure 2-3. Data RAM Bit Address Space

On any addressable bit, the Boolean processor can perform the bit operations of set, clear, complement, jump-if-set, jump-if-not-set, jump-if-set-then-clear and move to/from carry. Between any addressable bit (or its complement) and the carry flag it can perform the bit operation of logical and/or logical or with the result returned to the carry flag.

ON-CHIP PERIPHERALS

The second section of Chapter 2 describes the on-chip peripherals and external memory timing. This section begins with the interrupt system.

Interrupt System

Interrupts result in a transfer of control to a new program location. The program servicing the request begins at this address. In the 8051 there are five hardware resources that can generate an interrupt request. The starting address of the interrupt service program for each interrupt source is shown in Table 2-1.

A resource requests an interrupt by setting its associated interrupt request flag in the TCON or SCON register, as detailed in Table 2-2. The interrupt request will be acknowledged if its interrupt enable bit in the Interrupt Enable register (shown in Table 2-3) is set and if it is the highest priority level as established by the polarity of a bit in the Interrupt Priority register. These bit assignments are shown in Table 2-4. Setting the resource's associated bit to a one (1) programs it to the higher level. The priority of multiple interrupt requests occurring simultaneously and assigned to the same priority level is also shown in Table 2.4.

b.) Hardware Register Bit Addresses.

Direct Byte Address	(MSB)			Bit Addresses				(LSB)	Hardware Register Symbol
0FFH									
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0B8H	—	—	—	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
0A8H	AF	—	—	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Figure 2-4. Special Function Register Bit Address Space

The servicing of a resource's interrupt request occurs at the end of the instruction-in-progress. The processor transfers control to the starting address of this resource's interrupt service program and begins execution.

Within the Interrupt Enable register (IE) there are six addressable flags. Five flags enable/disable the five interrupt sources when set/cleared. Setting/clearing the sixth flag permits a global enable/disable of each enabled interrupt request.

Setting/clearing a bit in the Interrupt Priority (IP) register establishes its associated interrupt request as a high/low priority. If a low-priority level interrupt is being serviced, a high-priority level interrupt will interrupt it. However, an interrupt source cannot interrupt a service program of the same or higher level.

The processor records the active priority level(s) by setting internal flip-flop(s). One of these non-addressable flip-flops is set while a low-level interrupt is being serviced. The other flip-flop is set while the high-level interrupt is being serviced. The appropriate flip-flop is set when the processor transfers control to the service program. The flip-flop corresponding to the interrupt level being serviced is reset when the processor executes an RETI Instruction. To summarize, the sequence of events for an interrupt is: A resource provokes an interrupt by setting its associated interrupt request bit to let the processor know an interrupt condition has occurred. The CPU's internal hardware latches the internal request near the falling-edge of ALE in the tenth, twenty-second, thirty-fourth and forty-sixth oscillator period of the instruction-in-progress. The interrupt request is conditioned by bits in the interrupt enable and interrupt priority registers. The processor acknowledges the interrupt by setting one of the two internal "priority-level active" flip-flops and performing a hardware subroutine call. This call pushes the PC (but not the PSW) onto the stack and, for most sources, clears the interrupt request flag. The service program is then executed. Control is returned to the main program when the RETI instruction is executed. The RETI instruction also clears one of the internal "priority-level active" flip-flops. Most interrupt request flags (IE0, IE1, TF0 and TF1) are cleared when the processor transfers control to the first instruction of the interrupt service program. The TI and RI interrupt request flags are the exceptions and must be cleared as part of the serial port's interrupt service program.

Table 2-1. Starting Address for Interrupt Service Programs

Interrupt Source	Starting Address
External Request 0	3 (0003 H)
Internal Timer/Counter 0	11 (000B H)
External Request 1	19 (0013 H)
Internal Timer/Counter 1	27 (001B H)
Internal Serial Port	35 (0023 H)

Table 2-2. Interrupt Request Flags

Interrupt Source	Request Flag	Bit Location
External Request 0	IE0	TCON .1
Internal Timer/Counter 0	TF0	TCON.5
External Request 1	IE1	TCON.3
Internal Timer/Counter 1	TF1	TCON.7
Internal Serial Port (xmit)	TI	SCON.1
Internal Serial Port (rcvr)	RI	SCON.0

Table 2-3. Interrupt Enable Flags

Interrupt Source	Enable Flag	Bit Location
External Request 0	EX0	IE.0
Internal Timer/Counter 0	ET0	IE.1
External Request 1	EX1	IE.2
Internal Timer/Counter 1	ET1	IE.3
Internal Serial Port	ES	IE.4
Reserved	None	IE.5
Reserved	None	IE.6
All Enabled	EA	IE.7

Table 2-4. Interrupt Priority Flags

Interrupt Source	Priority Flag	Priority Within Level	Bit Location
External Request 0	PX0	.0 (highest)	IP.0
Internal Timer/ Counter 0	PT0	.1	IP.1
External Request 1	PX1	.2	IP.2
Internal Timer/ Counter 1	PT1	.3	IP.3
Internal Serial Port	PS	.4 (lowest)	IP.4
Reserved	None		IP.5
Reserved	None		IP.6
Reserved	None		IP.7

The process whereby a high-level interrupt request interrupts a low-level interrupt service program is called nesting. In this case the address of the next instruction in the low-priority service program is pushed onto the stack, the stack pointer is incremented by two (2) and processor control is transferred to the Program Memory location of the first instruction of the high-level service program. The last instruction of the high-priority interrupt service program must be an RETI instruction. This instruction clears the higher "priority-level-active" flip-flop. RETI also returns processor control to the next instruction of the low-level interrupt service program. Since the lower "priority-level-active" flip-flop has remained set, high priority interrupts are re-enabled while further low-priority interrupts remain disabled.

The highest-priority interrupt request gets serviced at the end of the instruction-in-progress unless the request is

made in the last fourteen oscillator periods of the instruction-in-progress. Under this circumstance, the next instruction will also execute before the interrupt's subroutine call is made. The first instruction of the service program will begin execution twenty-four oscillator periods (the time required for the hardware subroutine call) after the completion of the instruction-in-progress or, under the circumstances mentioned earlier, twenty-four oscillator periods after the next instruction.

Thus, the greatest delay in response to an interrupt request is 86 oscillator periods (approximately 7μsec @ 12MHz). Examples of the best and worst case conditions are illustrated in Table 2-5.

Table 2-5. Best and Worst Case Response to Interrupt Request

Instruction	Time (Oscillator Periods)	
	Best Case	Worst Case
1) External interrupt request generated immediately before (best)/after (worst) the pin is sampled. (Time until end of bus cycle.)	2 + ϵ	2 − ϵ
2) Current or next instruction finishes in 12 oscillator periods	12	12
3) Next instruction is MUL or DIV	don't care	48
4) Internal latency for hardware subroutine call	24	24
	38	86

EXTERNAL INTERRUPTS

The external interrupt request inputs ($\overline{INT0}$ and $\overline{INT1}$) can be programmed for either transition-activated or level-activated operation. Control of the external interrupts is provided by the four low-order bits of TCON as shown in Table 2-6.

Table 2-6. Function of Bits in TCON (Lower Nibble)

Function	Flag	Bit Location
External Interrupt Request Flag 1	IE1	TCON.3
Input $\overline{INT1}$ Transition-Activated	IT1	TCON.2
External Interrupt Request Flag 0	IE0	TCON.1
Input $\overline{INT0}$ Transition Activated	IT0	TCON.0

When IT0 and IT1 are set to one (1), interrupt requests on $\overline{INT0}$ and $\overline{INT1}$ are transition-activated (high-to-low), or else they are low-level activated. IE0 and IE1 are the interrupt request flags. These flags are set when their corresponding interrupt request inputs at $\overline{INT0}$ and $\overline{INT1}$, respectively, are low when sampled by the 8051 and the transition-activated scheme is selected by IT0 and IT1. When IT0 and IT1 are programmed for level-activated interrupts, the IE0 and IE1 flags are not affected by the inputs at $\overline{INT0}$ and $\overline{INT1}$, respectively.

Transition-Activated Interrupts

The external interrupt request inputs ($\overline{INT0}$ and $\overline{INT1}$) can be programmed for high-to-low transition-activated operation. For transition-activated operation, the input must remain low for greater than twelve oscillator periods, but need not be synchronous with the oscillator. It is internally latched by the 8051 near the falling-edge of ALE during an instruction's tenth, twenty-second, thirty-fourth and forty-sixth oscillator periods and, if the input is low, IE0 and IE1 is set. The upward transition of a transition-activated input may occur at any time after the twelve oscillator period latching time, but the input must remain high for twelve oscillator periods before reactivation.

Level-Activated Interrupts

The external interrupt request inputs ($\overline{INT0}$ and $\overline{INT1}$) can be programmed for level-activated operation. The input is sampled by the 8051 near the falling-edge of ALE during the instruction's tenth, twenty-second, thirty-fourth and forty-sixth oscillator periods. If the input is low during the sampling that occurs fourteen oscillator periods before the end of the instruction in progress, an interrupt subroutine call is made. The level-activated input need be low only during the sampling that occurs fourteen oscillator periods before the end of the instruction-in-progress and may remain low during the entire execution of the service program. However, the input must be raised before the service program completes to avoid possibly envoking a second interrupt.

Ports and I/O Pins

There are 32 I/O pins configured as four 8-bit ports. Each pin can be individually and independently programmed as input or output and each can be configured dynamically (i.e., on-the-fly) under software control.

An instruction that uses a port's bit/byte as a source operand reads a value that is the logical and of the last value written to the bit/byte and the polarity being applied to the pin/pins by an external device (this assumes that none of the 8051's electrical specs are being violated). An instruction that reads a bit/byte, operates

on the content, and writes the result back to the bit/byte, reads the last value written to the bit/byte instead of the logic level at the pin/pins. Pins comprising a single port can be made a mixed collection of inputs and outputs by writing a "one" to each pin that is to be an input. Each time an instruction uses a port as the destination, the operation must write "ones" to those bits that correspond to the input pins. An input to a port pin need not be synchronized to the oscillator. Each port pin is sampled near the falling-edge of ALE during the read instruction's tenth or twenty-second oscillator period. If an input is in transition when it is sampled near the falling-edge of ALE it will be read as an indeterminate value.

The instructions that perform a read of, operation on, and write to a port's bit/byte are INC, DEC, CPL, JBC, CJNE, DJNZ, ANL, ORL, and XRL. The source read by these operations is the last value that was written to the port, without regard to the levels being applied at the pins. This insures that bits written to a one (1) for use as inputs are not inadvertently cleared; see Figure 2-5.

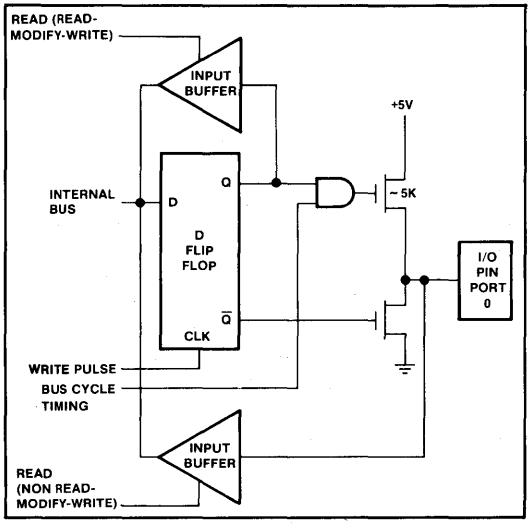


Figure 2-5. "Bidirectional" Port Structure

When used as a port, Port 0 has an open-drain output. When used as a bus, it has a standard three-state driver. The Port 0 output driver can sink/source two TTL loads.

Ports 1, 2 and 3 have quasi-bidirectional output drivers which incorporate a pull-up resistor of 10K-to-20K Ohms as shown in Figure 2-6.

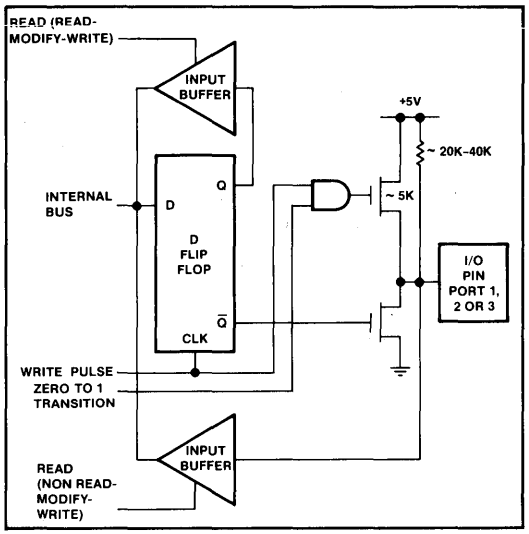


Figure 2-6. "Quasi-Bidirectional" Port Structure

In Ports 1, 2 and 3 the output driver provides source current for two oscillator periods if, and only if, software updates the bit in the output latch from a zero (0) to a one (1). Sourcing current only on "zero to one" transition prevents a pin, programmed as an input, from sourcing current into the external device that is driving the input pin. The output drivers in Ports 1, 2 and 3 can sink/source one TTL load.

Secondary functions ($\overline{RD}$, $\overline{WR}$, etc.) can be selected individually and independently for the pins of Port 3. Port 3 generates these secondary control signals automatically as long as the pin corresponding to the appropriate signal is programmed as an input.

Accessing External Memory

When accessing external memory the 8051 emits the upper address byte from Port 2 and the lower address byte, as well as the data, from Port 0. It uses ALE, $\overline{PSEN}$ and two pins from Port 3 ($\overline{RD}$ and $\overline{WR}$) for memory control. ALE is used for latching the address into the external memory. The $\overline{PSEN}$ signal enables the external Program Memory to Port 0, the $\overline{RD}$ signal enables External Data Memory to Port 0 and the $\overline{WR}$ signal latches the data byte emitted by Port 0 into the External Data Memory. Externally the $\overline{PSEN}$ and $\overline{RD}$ signals can be combined logically if a contiguous external program and data memory space (similar to a "von Neuman" machine) is desired. The P3.7 ($\overline{RD}$) and P3.6 ($\overline{WR}$) output latches must be programmed to a one (1) if External Data Memory is to be accessed. When P3.7 and P3.6 are programmed as $\overline{RD}$ and $\overline{WR}$, respectively, the remain-

ing pins of Port 3 may be individually programmed as desired. The 8051 can address 64K bytes of external Program Memory when the $\overline{EA}$ pin is tied low. When $\overline{EA}$ is high, the 8051 fetches instructions from internal Program Memory when the address is between 0 and 4095, and from external Program Memory when the addressed memory location is between 4096 and 64K. In either case, Ports 2 and 0 are automatically configured as an external bus, based on the value of the PC. Instruction execution times are the same for code fetched from internal or external Program Memory.

Up to 64K of External Data Memory can be accessed using the MOVX instructions. These instructions automatically configure Port 0, and often Port 2, as an external bus. The MOVX instructions use the DPTR, R1 or R0 register as a pointer into the External Data Memory. The 16-bit DPTR register is used when successive accesses cover a wide range of the 64K space. The 8-bit R1 and R0 registers provide greatest byte efficiency when successive accesses are constrained to a 256-byte block of the External Data Memory space. When using R1 and R0, a subsequent block can be accessed by updating the output latch of Port 2. Port 2 is not affected by execution of a MOVX instruction that uses R1 or R0. If, for example, 32K or less of external data memory is present, only part of Port 2 needs to be used for selecting the desired block; the remaining pins of Port 2 can be used for I/O. When a MOVX using DPTR is executed, the value in Port 2's output latch is altered only during the external access and then is returned to its prior value. This permits efficient external block moves by interleaving MOVX instructions that use DPTR and R1 and R0.

The ALE signal is generated every sixth oscillator period during reads from either internal or external Program Memory. The $\overline{PSEN}$ signal is generated every sixth oscillator period when reading from the external Program Memory. When a read or write from External Data Memory is being performed, a single ALE and a $\overline{RD}$ or a $\overline{WR}$ signal is generated during a twelve oscillator period interval. The 8051 always fetches an even number of bytes from its Program Memory. If an odd number of bytes are executed prior to a branch or to an External Data Memory access, the nonexecuted byte is ignored by the 8051. If an instruction requires more oscillator periods for its execution than for its fetch, the first byte of the next instruction is fetched repeatedly while the first instruction completes execution. If the CPU does not address External Data Memory, then ALE is generated every sixth oscillator period and can be used as an external clock. When External Data Memory is present, external logic may be used to combine the occurrence of $\overline{RD}$, $\overline{WR}$, and ALE to generate an external clock with a period equal to six oscillator periods.

ACCESSING EXTERNAL MEMORY—OPERATION OF PORTS

The Port 0 bus is time-multiplexed to permit transfer of both addresses and data. This bus is used directly by memory and peripheral devices that incorporate on-chip address latching (MCS-85 memories with peripherals), or it can be de-multiplexed with an address latch to generate a non-multiplexed bus (MCS-80 peripherals and memory). During an external access, the low-order byte of the address and the data (for a write) is emitted by the Port 0 output drivers. Ones (1's) are automatically written to Port 0 at the very end of the bus cycle. Since the Port 0 output latches will contain ones (1's) at the end of the bus cycle, Port 0 will be in its high impedance state when a bus cycle is not in progress. Port 2 emits the upper 8-bits of the address when a MOVX instruction using DPTR is executed. Port 2's output drivers provide source current for two oscillator periods when emitting the address. Port 2's internal pull-up resistors sustain the high level.

ACCESSING EXTERNAL MEMORY—BUS CYCLE TIMING

(The following section is a description of the 8051's timings. For design purposes, please refer to the specification section in Chapter 6 for 8051 timing parameters.)

Each Program Memory bus cycle consists of six oscillator periods. These are referred to as T1, T2, T3, T4, T5 and T6 on Figure 2-7. The address is emitted from the processor during T3. Data transfer occurs on the bus during T5, T6, and the following bus cycle's T1. When fetching from external Program Memory, the 8051 will always fetch an even number of bytes. If an odd number of bytes are executed prior to a branch, or an External Data Memory access, the non-executed byte will be ignored by the 8051. An even number of idle bus cycles (each 6 oscillator periods in duration) can occur between external bus cycles when the processor is fetching from internal Program Memory. The read cycle begins during T2, with the assertion of address latch enable signal ALE ①. The falling edge of ALE ② is used to latch the address information, which is present on the bus at this time ③, into the 8282 latch if a non-multiplexed bus is required. At T5, the address is removed from the Port 0 bus and the processor's bus drivers go to the high-impedance state ④. The program memory read control signal ($\overline{PSEN}$) ⑤ is also asserted during T5. $\overline{PSEN}$ causes the addressed device to enable its bus drivers to the now-released bus. At some later time, valid instruction data will become available on the bus ⑥. When the 8051 subsequently returns $\overline{PSEN}$ to the high level ⑦, the addressed device will then float its bus drivers, relinquishing the bus again ⑧.

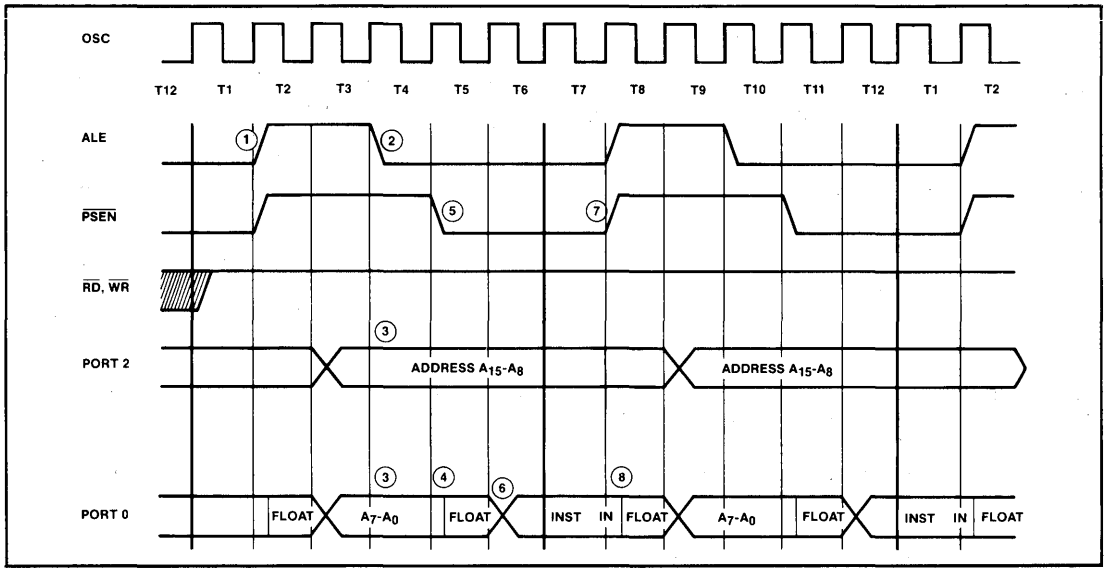


Figure 2-7. Program Memory Read Cycle Timing

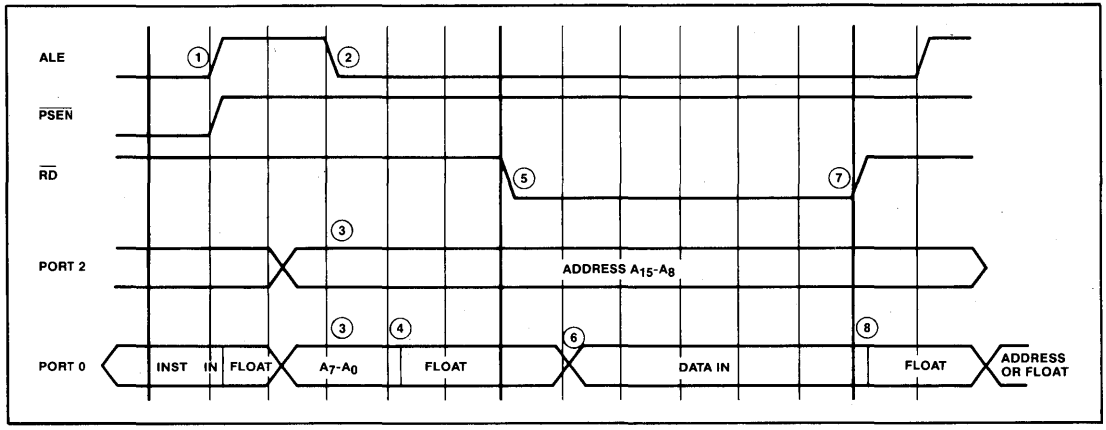


Figure 2-8. Data Memory Read Cycle Timing

For the MOVC instruction, the op-code is fetched in the first six-oscillator period, the first byte of the next instruction is fetched during the second six-oscillator period, the table entry is fetched in a third six-oscillator period and the first byte of the next instruction is again fetched in the fourth six-oscillator period.

Each External Data Memory bus cycle consists of twelve oscillator periods. These are shown as T1 through T12 on Figure 2-8. The twelve-period External Data Memory cycle allows the 8051 to use peripherals that are relatively slower than its program memories. The address is emitted from the processor during T3. Data transfer occurs on the bus during T7 through T12. T5 and T6 is the period during which the direction of the bus is changed for the read operation. The read cycle begins during T2, with the assertion of address latch enable signal ALE ①. The falling edge of ALE ② is used to latch the address information, which is present on the bus at this time ③, into the 8282 latch if a non-multiplexed bus is required. At T5, the address is removed from the Port 0 bus and the processor's bus drivers go to the high-impedance state ④. The data memory read control signal $\overline{RD}$ ⑤, is asserted during T7. $\overline{RD}$ causes the addressed device to enable its bus drivers to the now-released bus. At some later time, valid data will be available on the bus ⑥. When the 8051 subsequently returns $\overline{RD}$ to the high level ⑦, the

addressed device will then float its bus drivers, relinquishing the bus again ⑧.

The write cycle, like the read cycle, begins with the assertion of ALE ① and the emission of an address ② as shown in Figure 2-9. In T6, the processor emits the data to be written into the addressed data memory location ③. This data remains valid on the bus until the end of the following bus cycle's T2 ④. The write signal $\overline{WR}$ goes low at T6 ⑤ and remains active through T12 ⑥.

Timer/Counters

Two independent 16-bit timer/counters are on-board the 8051 for use in measuring time intervals, measuring pulse widths, counting events, and causing periodic (repetitious) interrupts.

TIMER/COUNTER MODE SELECTION

Counter 1 can be configured in one of four modes:

Mode 0) Provides an 8-bit counter with a divide-by-32 prescaler or an 8-bit timer with a divide-by-32 prescaler. A read/write of TH1 accesses counter 1's bits 12-5. A read/write of TL1 accesses counter 1's bits 7-0. TL1 bits 4-0 are the prescalar (counter 1's bits 4-0), while bits 7-5 are indeterminate and should be ignored. The programmer should clear the prescaler (counter 1's bits 4-0) before setting the run flag.

Mode 1) Configures counter 1 as a 16-bit timer/counter.

Mode 2) Configures counter 1 as an 8-bit auto-reload timer/counter. TH1 holds the reload value. TL1 is incremented. The value in TH1 is reloaded into TL1 when TL1 overflows from all ones (1's). An 8048 compatible counter is achieved by configuring to mode 2 after zeroing TH1.

Mode 3) When counter 1's mode is reprogrammed to mode 3 (from mode 0, 1 or 2), it disables the incrementing of the counter. This mode is provided as an alternative to using the TR1 bit (TCON.6) to start and stop counter 1.

The serial port receives a pulse each time that counter 1 overflows. The standard UART modes divide this pulse rate to generate the transmission rate.

Counter 0 can also be configured in one of four modes:

Modes 0-2) Modes 0-2 are the same as for counter 1.

Mode 3) In Mode 3, the configuration of TH0 is not affected by the bits in TMOD or TCON (see next section). It is configured solely as an 8-bit timer that is enabled for incrementing by TCON's TR1 bit. Upon TH0's overflow, the TF1 flag gets set. Thus, neither TR1 nor TF1 is available to counter 1 when counter 0 is in Mode 3. The function of TR1 can be done by placing counter

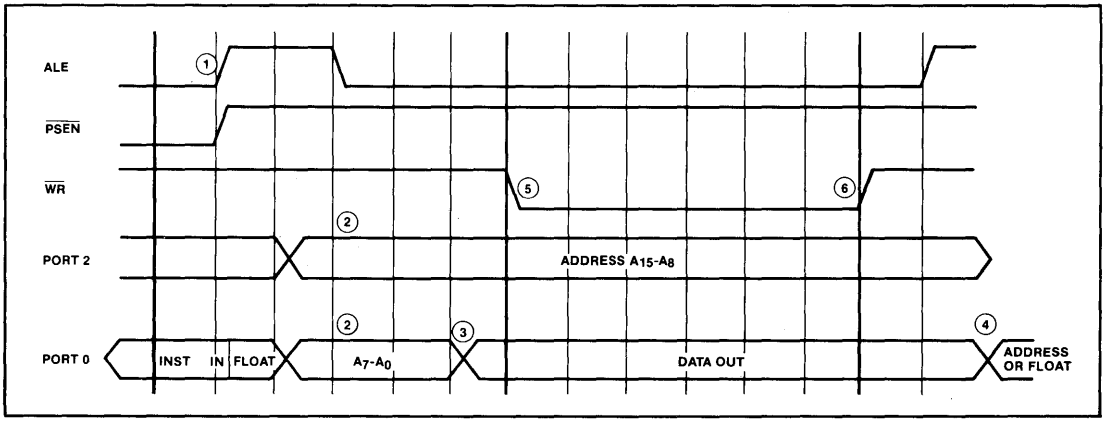


Figure 2-9. Data Memory Write Cycle Timing

1 in Mode 3, so only the function of TF1 is actually given up by counter 1. In Mode 3, TL0 is configured as an 8-bit timer/counter and is controlled, as usual, by the Gate (TMOD.3) C/$\overline{T}$ (TMOD.2), TRO (TCON.4) and TF0 (TCON.5) control bits.

CONFIGURING THE TIMER/COUNTER INPUT

The use of the timer/counter is determined by two 8-bit registers, TMOD (timer mode) and TCON (timer control). The counter input circuitry is shown in Figures 2-10 and 2-11. The input to the counter circuitry is from an external reference (for use as a counter), or from the on-chip oscillator (for use as a timer), depending on whether TMOD's C/$\overline{T}$ bit is set or cleared, respectively. When used as a time base, the on-chip oscillator frequency is divided by twelve (12) before being input to the counter circuitry. When TMOD's Gate bit is set (1), the external reference input (T1, T0) or the oscillator input is gated to the counter conditional upon a second external input ($\overline{INT0}$), ($\overline{INT1}$) being high. When the Gate bit is zero (0), the external reference, or oscillator input, is unconditionally enabled. In either case, the normal interrupt function of $\overline{INT0}$ and $\overline{INT1}$ is not affected by the counter's operation. If enabled, an interrupt will occur when the input at $\overline{INT0}$ or $\overline{INT1}$ is low. The counters are enabled for incrementing when TCON's TR1 and TR0 bits are set. When the counters overflow, the TF1 and TF0 bits in TCON get set, and interrupt requests are generated. The functions of the bits in TCON are shown in Table 2-7.

Table 2-7. Function of Bits in TCON (Upper Nibble)

Function	Flag	Bit Location
Counter interrupt request and overflow Flag	TF1	TCON.7
Counter enable/disable bit	TR1	TCON.6
Counter interrupt request and overflow Flag	TF0	TCON.5
Counter enable/disable bit	TR0	TCON.4

The functions of the bits in TMOD are shown in Table 2-8. Recall that the bits in TMOD are not bit addressable.

Table 2-8. Functions of Bits in TMOD

Function	Flag	Bit Location
Enable input at T1 using $\overline{INT1}$	Gate	TMOD.7
Counter 1/Timer 1 select	C/$\overline{T}$	TMOD.6
C 1/T 1 Mode select MSb	M1	TMOD.5
C 1/T 1 Mode select LSb	M0	TMOD.4
Enable input to T0 using $\overline{INT0}$	Gate	TMOD.3
Counter 0/Timer 0 select	C/$\overline{T}$	TMOD.2
C 0/T 0 Mode select MSb	M1	TMOD.1
C 0/T 0 Mode select LSb	M0	TMOD.0

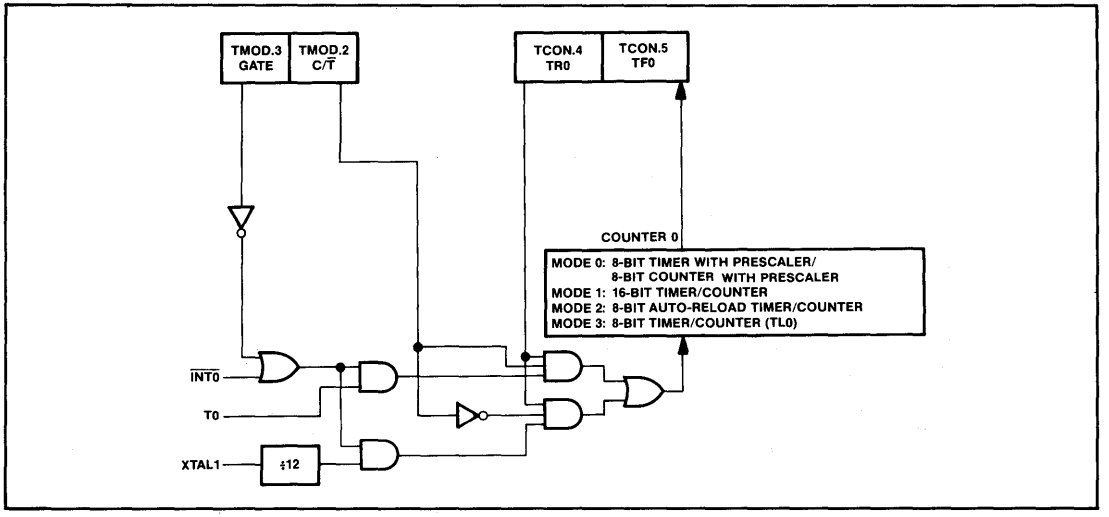


Figure 2-10. Timer/Event Counter 0 Control and Status Flag Circuitry

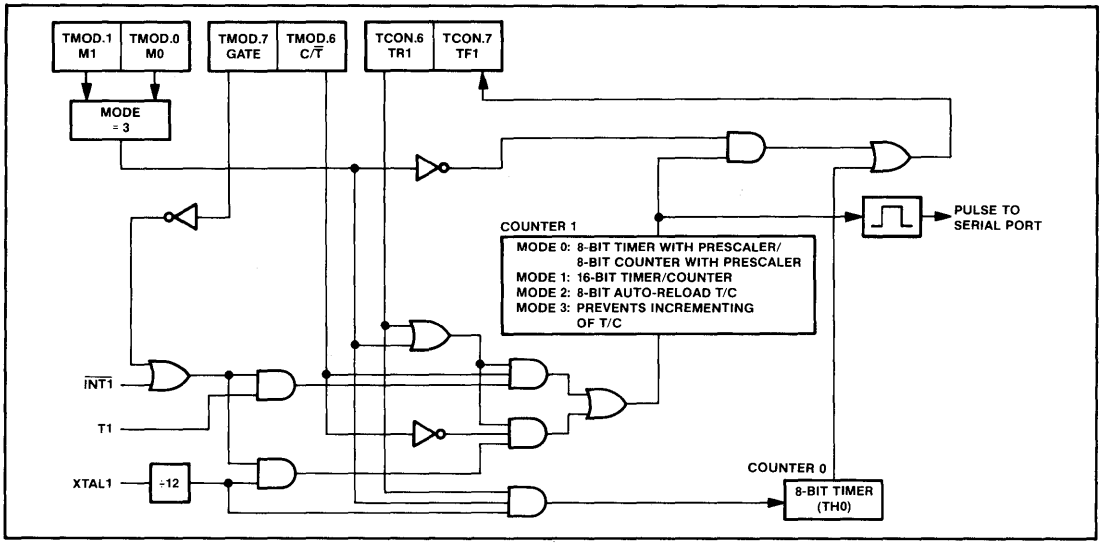


Figure 2-11. Timer/Event Counter 1 Control and Status Flag Circuitry

OPERATION

The counter circuitry counts up to all 1's and then overflows to either 0's or the reload value. Upon overflow, TF1 or TF0 gets set. When an instruction changes the timer's mode or alters its control bits, the actual change occurs at the end of the instruction's execution.

The T1 and T0 inputs are sampled near the falling-edge of ALE in the tenth, twenty-second, thirty-fourth and forty-sixth oscillator periods of the instruction-in-progress. They are also sampled in the twenty-second oscillator period of MOVX, despite the absence of ALE. Thus, an external reference's high and low times must each be a minimum of twelve oscillator periods in duration. There is a twelve oscillator period delay from the time when a toggled input (transition from high to low) is sampled to the time when the counter is incremented.

Serial Channel

The 8051 has a serial channel useful for serially linking UART (universal asynchronous receiver/transmitter) devices and for expanding I/O. This full-duplex serial I/O port can be programmed to function in one of four operating modes:

Mode 0) Synchronous I/O expansion using TTL or CMOS shift registers

Mode 1) UART interface with 10-bit frame and variable transmission rate

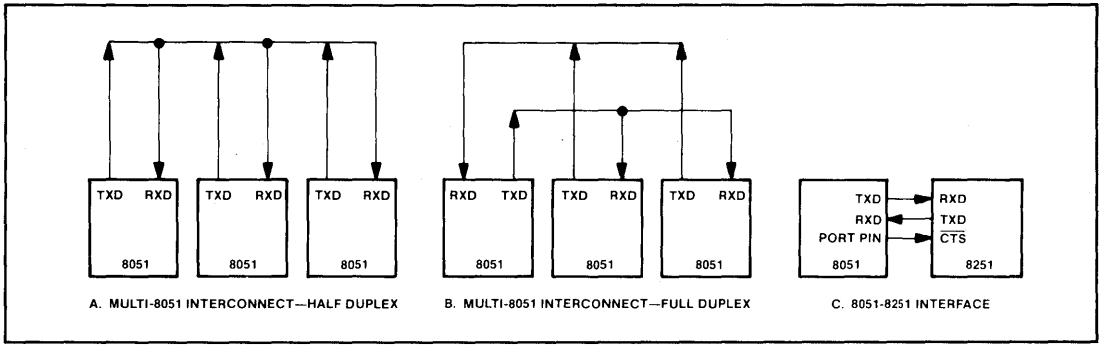


Figure 2-12. UART Interface Technique

Mode 2) UART interface with 11-bit frame and fixed transmission rate

Mode 3) UART interface with 11-bit frame and variable transmission rate

Modes 2 and 3 also provide automatic wake-up of slave processors through interrupt driven address-frame recognition for multiprocessor communications. Several schemes of UART interfacing are shown in Figure 2-12, and an I/O expansion technique is shown in Figure 2-13.

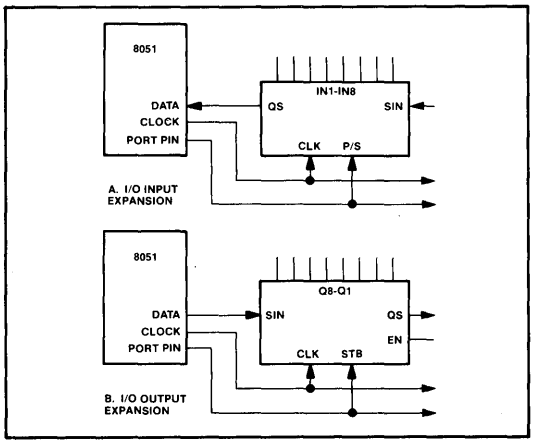


Figure 2-13. I/O Expansion Technique

SERIAL CHANNEL CONTROL AND DATA REGISTERS

Data for transmission and from reception reside in the serial port buffer register (SBUF). A write to SBUF updates the transmitter register, while a read from SBUF reads a buffer that is updated by the receiver register if/when flag RI is reset. The receiver is double-buffered to eliminate the overrun that would occur if the CPU failed to respond to the receiver's interrupt before the beginning of the next frame. In general, double buffering of the transmitter is not needed for the high performance 8051 to maintain the serial link at its maximum rate. A minor degradation in data rate can occur in rare events, such as when the servicing of the transmitter has to wait for a lengthy interrupt service program to complete. In asynchronous mode, false start-bit rejection is provided on received frames. A two-out-of-three vote is taken on each received bit for noise rejection. The serial port's control and the monitoring of its status is provided by the serial port control register (SCON). The contents of the 8-bit SCON register are shown in Table 2-9.

Table 2-9. Functions of Bits in SCON

Function	Flag	Bit Location
Serial Port Operation Mode (MSb)	SM0	SCON. 7
Serial Port Operation Mode (LSb)	SM1	SCON. 6
Conditional Receiver Enable	SM2	SCON. 5
Receiver Enable	REN	SCON. 4
Transmitter Data Bit 8	TB8	SCON. 3
Received Data Bit 8	RB8	SCON. 2
Transmission Complete Interrupt Flag	TI	SCON. 1
Reception Complete Interrupt Flag	RI	SCON. 0

Mode control bits SM0 and SM1 program the serial port in one of four operating modes. A detailed description of the modes is provided later in the text. The receiver-enable bit (REN) resets the receiver's start/stop logic. When software sets REN to one (1), the receiver's transmission-rate generator is initialized and reception is enabled. REN must be set as part of the serial channel's initialization program. When REN is cleared, reception is disabled.

The CPU is informed that the transmitter portion of SBUF is empty, or the receiver portion is full, by TI and RI, respectively. TI and RI must be cleared as part of the interrupt service program so as not to continuously interrupt the CPU. Since TI and RI are "or-ed" together to generate the serial port's interrupt request, they must be polled to determine the source of the interrupt.

OPERATING MODES

Operating Mode 0

The I/O expansion mode, Mode 0, is used to expand the number of input and output pins. In this mode, a clock output is provided for synchronizing the shifting of bits into, or from, an external register. Eight bits will be shifted out each time a data byte is written to the serial channel's data buffer (SBUF), even if TI is set. Each time software clears the RI flag, eight bits are shifted into SBUF before the RI flag is again set. The receiver must be enabled [i.e., REN set to (1)] for reception to occur.

The synchronizing clock is output on pin P3.1 and toggles from high to low near the falling-edge of ALE in the fifteenth oscillator period following execution of the

instruction that updated SBUF or cleared the RI flag. It then toggles near the falling-edge of ALE in each subsequent sixth oscillator period until 8-bits are transferred. The eighth rising-edge of clock (P3.1) sets the RI or TI flag. At this point, shifting is complete and the clock is once again high. The first bit is shifted out of P3.0 at the beginning of the eighteenth oscillator period, following the instruction that updated SBUF. The first bit shifted in from P3.0 is latched by the clock's rising-edge in the twenty-fourth oscillator period, following the instruction that cleared the RI flag. One bit is shifted every twelfth oscillator period, until all eight bits have been shifted.

Operating Modes 1 through 3

In the UART Modes (i.e., 1 through 3), the transmission rate is subdivided into 16 "ticks." The value of a received bit is determined by taking a majority vote after it has been sampled during the seventh, eighth and ninth "ticks." If two or three ones (1's) are detected, the bit will be given a one (1) value; if two or three zeros (0's) are detected, the bit will be given a zero (0) value.

Until a start bit arrives, the receiver samples the RXD input pin (P3.0) every "tick." Approximately one-half bit time (eight "ticks") after the start bit is detected (i.e., a low input level was sampled on "tick" one), the serial port checks its validity (majority vote from "ticks" seven, eight and nine) and accepts or rejects it. This provides rejection of false start bits.

The contents of the receiver's input shift register is moved to SBUF and RB8 (Modes 2 and 3), and RI is set when a frame's ninth (Mode 1) or tenth (Modes 2 and 3) bit is received. Upon reception of a second frame's ninth or tenth bit, the data bits in the shift register are again transferred to SBUF and RB8, but only if software has reset the RI flag. If RI has not been reset, then overrun will occur, since the shift register will continue to accept bits. Double buffering the receiver provides the CPU with one frame-time in which to empty the SBUF and RB8 registers. The RI flag is set and bit RB8 is loaded during the ninth "tick" of the received frame's ninth or tenth bit. The serial port begins looking for the next start bit approximately one-half bit time after the center of a stop bit is received.

Data is transmitted from the TXD output pin (P3.1) each time a byte is written to SBUF, even if TI is set. Transmission of the start bit begins at the end of the instruction that updates SBUF. TI is set at the beginning of the transmitted tenth (Mode 1) or eleventh (Mode 2 or 3) bit. After TI becomes set, if SBUF is written-to prior to the end of the stop (tenth or eleventh) bit, the transmission of the next frame's start bit will not begin until the end of the stop bit.

In Modes 2 and 3, if SM2 is set, frames are received, but an interrupt request is generated only when the received data bit 8 (RB8) is a one (1). This feature permits interrupt generated wake-up during interprocessor communications when multiple 8051's are connected to a serial bus. Thus, data bit 8 (RB8) awakens all processors on the serial bus only when the master is changing address to a different processor. Each processor not addressed then ignores the subsequent transmission of control information and data. A protocol for multi-8051 serial communications is shown in Figure 2-14. The SM2 bit has no effect in Modes 0 and 1.

1. **The hardware in each slave's serial port begins by listening for an address. Receipt of an address frame will force an interrupt if the slave's SM2 bit is set to one (1) to enable "interrupt on address frame only".**
2. **The master then transmits a frame containing the 8-bit address of the slave that is to receive the subsequent commands and data. A transmitted address frame has its ninth data bit (TB8) set equal to one (1).**
3. **When the address frame is received, each slave's serial port interrupts its CPU. The CPU then compares the address sent to its own.**
4. **The 8051 slave which has been addressed then resets its SM2 bit to zero (0) to receive all subsequent transmissions. All other 8051's leave their SM2 bits at a one (1) to ignore transmissions until a new address arrives.**
5. **The master device then sends control information and data, which in turn is accepted by the previously addressed 8051 [i.e., the one that had set its SM2 bit to zero (0)].**

Figure 2-14. Protocol for Multi-Processor Communications

THE SERIAL FRAME

A frame is a string of bits. The frame transmitted and received in Mode 0 is 8 bits in length. The data bits of the frame are transmitted SBUF.0 first and SBUF.7 last.

The frame transmitted and received in Mode 1 is ten bits in length. The frame transmitted and received in Modes 2 and 3 is eleven bits in length. These frames consist of one start bit, eight or nine data bits and a stop bit. Data bits 0-7 are loaded into SBUF.0-SBUF.7, respectively, and data bit 8 into RB8 (receive) or TB8 (transmit). With nominal software overhead, the last data bit can be made a parity bit, as shown in Figure 2-15.

Figure 2-16 shows some typical frame formats for different applications. The data bits of the frame are transmitted least significant bit first (SBUF.0) and TB8 last.

```
MOV C, P      ; Parity moved to carry (byte
                already in A).
MOV TB8, C    ; Put carry into Transmitter Bit 8
MOV SBUF, A; Load Transmit Register
```

Figure 2-15. Generating Parity and Transmitting Frame

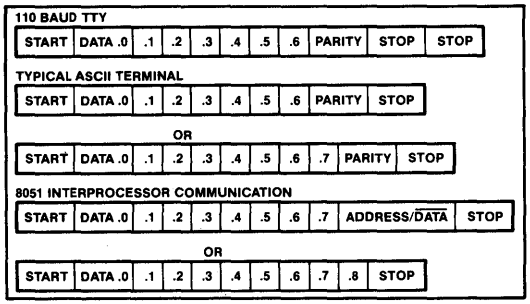


Figure 2-16. Typical Frame Formats

TRANSMISSION RATE GENERATION

The proper timing for the serial I/O data is provided by a transmission-rate generator. On-board the 8051, three different methods of transmission rate generation are provided. The transmission-rate achievable is dependent upon the operating mode of the serial port.

In the I/O expansion mode (Mode 0) the oscillator frequency is simply divided by 12 to generate the transmission rate. This produces a transmission rate of 1M bits per second at 12MHz. If Modes 1 or 3 are being used, the transmission rate can be generated from the oscillator frequency or from an external reference frequency. In these modes, either one-twelfth the oscillator frequency, or the T1 input frequency, is divided by 256-minus-the-value-in-TH1 (counter 1 must be configured in auto-reload mode by software) and then divided by 32 to generate the transmission rate. When the oscillator frequency input (rather than T1) is selected, this method produces a transmission rate of 122 to 31,250 bits per second (including start and stop bits) at 12MHz. The T1 external input is selected by setting the C/$\overline{T}$ bit to one (1). When Mode 2 is used, the oscillator frequency is simply divided by 64 to generate the transmission rate. This produces a transmission rate of 187,500 bits per second (including start and stop bits) at 12MHz.

UART ERROR CONDITIONS

There are two UART error conditions that should be accounted for when designing systems that use the serial channel.

First, the 8051's serial channel provides no indication that a valid stop bit has been received. However, since a start bit is detected as a high-level to low-level transition, the UART will not receive additional frames if a stop bit is not received. Second, the RI flag is set and SBUF and RB8 are loaded from the receiver's input shift register when the received last data bit (i.e., ninth or tenth received bit) is sampled. As long as RI is set, the loading of SBUF, the updating of RB8, and the generation of further receiver interrupts is inhibited. Thus, overrun will occur if the programmer does not reset RI before reception of the next frame's last data bit, since the receiver's input shift register will shift in a third frame.

Processor Reset and Initialization

Processor initialization is accomplished with activation of the RST/VPD pin. To reset the processor, this pin should be held high for at least twenty-four oscillator periods. Upon powering up, RST/VPD should be held high for at least 24 oscillator periods, after the power supply stabilizes, to allow the oscillator to stabilize. Crystal operation below 6MHz will increase the time necessary to hold RST/VPD high. 24 oscillator periods after receipt of RST, the processor ceases instruction execution and remains dormant for the duration of the pulse. The low-going transition then initiates a sequence which requires approximately twelve oscillator periods to execute before ALE is generated and normal operation commences with the instruction at absolute location 0000H. Program Memory locations 0000H through 0003H are reserved for the initialization routine of the microcomputer. This sequence ends with registers initialized as shown in Table 2-10.

Table 2-10. Register Initialization

Register	Content
PC	0000H
SP	07H
PSW, DPH, DPL, A, B,	00H
IP, IE, SCON, TCON,	00H
TMOD, TH1, TH0,	00H
TL1, TL0	00H
IP	E0H
IE	60H
SBUF	Indeterminate
Port 3-Port 0	FFH (configures all I/O pins as inputs)
Internal RAM	Unchanged if VPD applied; else indeterminate

In addition, certain of the control pins are driven to a high level during reset. These are ALE/$\overline{PROG}$ and $\overline{PSEN}$. Thus, no ALE or $\overline{PSEN}$ signals are generated while RST/VPD is high. After the processor is reset, all ports are written with one (1's). Outputs are undefined until the reset period is complete. An external reset circuit, such as that in Figure 2-17, can be used to reset the microcomputer.

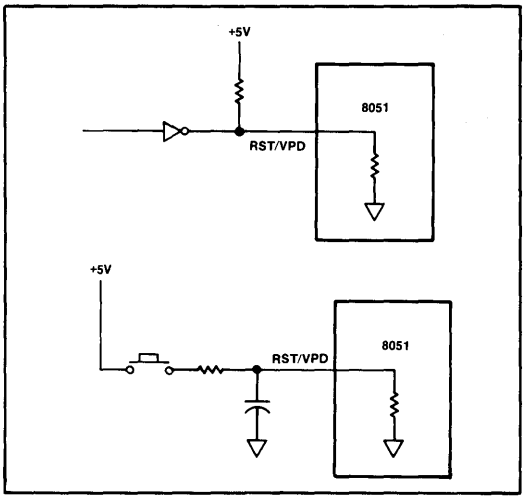


Figure 2-17. External Reset

Power Down (Standby) Operation of Internal RAM

Data can be maintained valid in the Internal Data RAM while the remainder of the 8051 is powered down. When powered down, the 8051 consumes about 10% of its normal operating power. During normal operation, both the CPU and the internal RAM derive their power from VCC. However, the internal RAM will derive its power from RST/VPD when the voltage of VCC is zero.

When a power-supply failure is imminent, the user's system generates a "power-failure" signal to interrupt the processor via $\overline{INT0}$ or $\overline{INT1}$. This power-failure signal must be early enough to allow the 8051 to save all data that is relevant for recovery before VCC falls below its operating limit. The program servicing the power-failure interrupt request may save any important data and machine status into the Internal Data RAM. The service program must also enable the backup power supply to RST/VPD pin. Applying power to the RST/VPD pin resets the 8051 and retains the internal RAM data valid as the VCC power supply falls below limit. Normal operation resumes when RST/VPD is returned low. Figure 2-18 shows the waveforms for the power-down sequence.

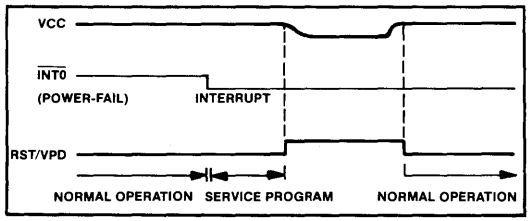


Figure 2-18. Power-Down Sequence

EPROM Programming

The 8751 is programmed, and the 8051 and 8751 are verified, using the UPP-851 programming card. For programming and verification, address is input on Port 1 and Port pins 2.0-2.3. Pins P2.4 and P2.5 are held to a TTL low (V_{IL}). Data is input and output through Port 0. RST/VPD is held at a TTL high level (V_{IH1}) and PSEN is held at TTL low level (V_{IL}) during program and verify. To program, ALE/PROG is held at a TTL low level (V_{IL}). The programming supply voltage is held at 21V. The $\overline{EA}$/VDD pin receives this programming supply voltage. ALE/PROG is held at TTL high level (V_{IH}) to verify the program. Port pin 2.7 forces the Port 0 output drivers to the high impedance state when held at a TTL high level and is held at a TTL low level for verification. Erasure of an 8751 will leave the EPROM programmed to an all one's (1's) state.

Data is introduced by programming "0's" into the desired bit locations. Although only "0's" will be programmed, both "1's" and "0's" can be presented in the data word. The only way to change a "0" to a "1" is by ultra-violet light erasure.

When $\overline{EA}$/VDD is at 21V, the 8751 is in the programming mode. It is necessary to put a capacitor between $\overline{EA}$/VDD and ground to block spurious voltage transients. ALE/PROG receives the 50 msec, active low TTL program pulse when the address and data are stable. A program pulse must be applied at each address location to be programmed. You can program any location at any time—either sequentially or at random.

A verify may be performed on the programmed bits to ensure correct programming. Data is output on Port 0 when pin 2.7 is at a TTL low level (V_{IH}). Pull-up resistors must be provided on Port 0 pins during verification. This verification mode can also be used to check the 8051's ROM pattern.

Table 2-11 describes the levels needed on the pins to program and verify the 8751.

Table 2-11. Voltage Inputs for EPROM Programming/Verifying

PINS / MODE	VPD/RST	PSEN	PROG/ALE	VDD/EA	P24-26	P27
PROGRAM	V_{IH1}	V_{IL}	V_{IH}	V_{PP}	V_{IL}	V_{IL}
VERIFY	V_{IH1}	V_{IL}	V_{IL}	V_{CC}	V_{IL}	V_{IL} (enable) V_{IH} (disable)

Memory Organization, Addressing Modes and Instruction Set

3

CHAPTER 3
MEMORY ORGANIZATION, ADDRESSING MODES AND INSTRUCTION SET

Chapter 3 is divided into these categories:

- Memory Organization
- Addressing Modes
- Instruction Set

The first part, Memory Organization, describes the memory spaces of the 8051. The Addressing Modes section describes addressing techniques used to reach the memory spaces. The final section explains the instruction set, including functional groupings, opcodes and a software example.

MEMORY ORGANIZATION

In the 8051 family the memory is organized over three address spaces and the program counter. The memory spaces shown in Figure 3-1 are the:

- 16-bit Program Counter
- 64K-byte Program Memory address space
- 64K-byte External Data Memory address space
- 384-byte Internal Data Memory address space

The 16-bit Program Counter register provides the 8051 with its 64K addressing capabilities. The Program Counter allows the user to execute calls and branches to any locations within the Program Memory space. There are no instructions that permit program execution to move from the Program Memory space to any of the data memory spaces.

In the 8051 and 8751, the lower 4K of the 64K Program Memory address space is filled by internal ROM and EPROM, respectively. By tying the $\overline{EA}$ pin high, the processor can be forced to fetch from the internal ROM/EPROM for Program Memory addresses 0 through 4K. Bus expansion for accessing Program Memory beyond 4K is automatic, since external instruction fetches occur automatically when the Program Counter increases above 4095. If the $\overline{EA}$ pin is tied low, all Program Memory fetches are from external memory. The execution speed of the 8051 is the same, regardless of whether fetches are from internal or external Program Memory. If all program storage is on-chip, byte location 4095 should be left vacant to prevent an undesired pre-fetch from external Program Memory address 4096.

Certain locations in Program Memory are reserved for specific programs. Locations 0000 through 0002 are reserved for the initialization program. Following reset, the CPU always begins execution at location 0000. Locations 0003 through 0042 are reserved for the five interrupt-request service programs.

The 64K-byte External Data Memory address space is automatically accessed when the MOVX instruction is executed.

Functionally, the Internal Data Memory is the most flexible of the address spaces. The Internal Data Memory space is subdivided into a 256-byte Internal Data RAM address space and a 128-byte Special Function Register address space. Within these address spaces are 256 individually addressable bits. Figure 3-2 shows the locations of the address spaces.

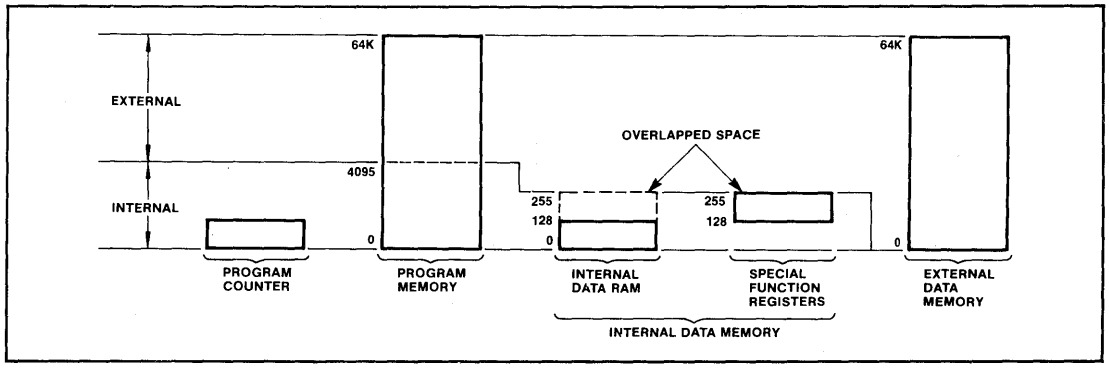


Figure 3-1. 8051 Family Memory Organization

AFN-01739A

The Internal Data RAM address space is 0 to 255. Four banks of eight registers occupy locations 0 through 31. The stack can be located anywhere in the Internal Data RAM address space. In addition, 128 bit locations of the on-chip RAM are accessible through Direct Addressing. (See next section, Addressing Modes.) These bits reside in Internal Data RAM at byte locations 32 through 47, as shown in Figure 3-3. Currently locations 0 through 127 of the Internal Data RAM address space are filled with on-chip RAM. Locations 128 through 255 may be filled on later products without affecting existing software.

The stack depth is limited only by the available Internal Data RAM, thanks to an 8-bit reloadable Stack Pointer. The stack is used for storing the Program Counter during subroutine calls, and may be used for passing parameters. Any byte of Internal Data RAM or Special Function Register's accessible through Direct Addressing can be pushed/popped.

The Special Function Register address space is 128 to 255. All registers except the Program Counter and the four banks of eight working registers reside here. Memory mapping the Special Function Registers allows them to be accessed as easily as internal RAM. As such, they can be operated on by most instructions. In addition, 128 bit locations within the Special Function Register address space can be accessed using Direct Addressing as shown in Figure 3-4. These bits reside in the Special Function Register address space and can be accessed using Direct Addressing. The addressable bits are located at byte addresses divisible by eight. An easy way to determine which byte locations are bit addressable are those byte locations ending in zero (0) or eight (8) when represented in hexadecimal notation. The twenty Special Function Registers are listed in Table 3-1. Their mapping in the Special Function Register address space is shown in Figure 3-5.

ADDRESSING MODES

Since the MCS-51 architecture differentiates between Data Memory and Program Memory, there are different addressing modes for each. These are explained below.

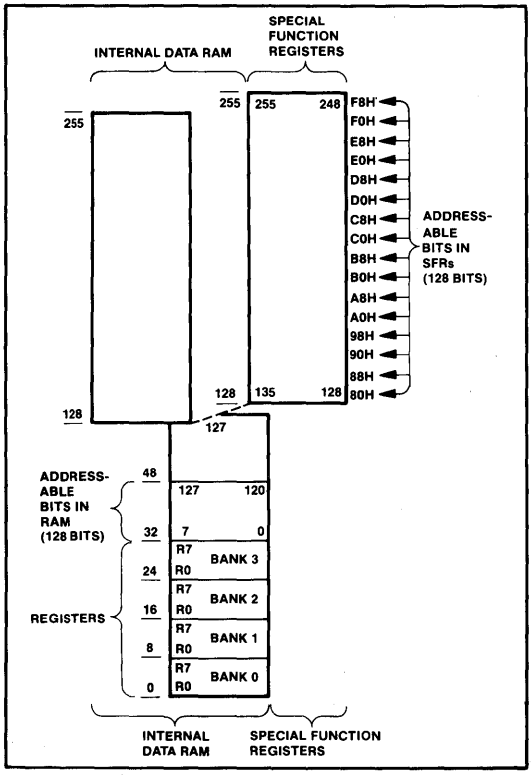


Figure 3-2. Internal Data Memory Address Space

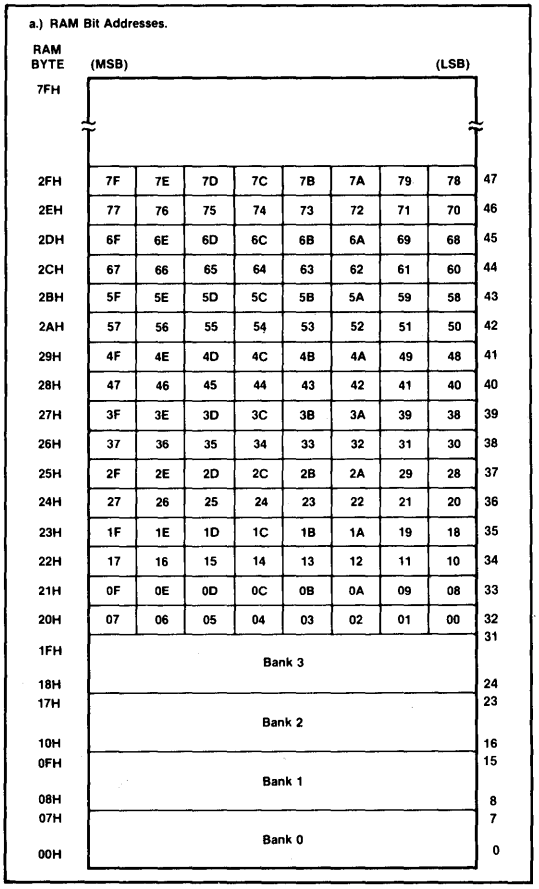


Figure 3-3. RAM Bit Addresses

There are five general addressing modes operating on bytes. One of these five addressing modes, however, operates on both bytes and bits.

Register Addressing

Register addressing encodes, in the low-order three bits of the instruction opcode, the number of a register in the currently enabled register bank. RS1 (PSW.4) and RS0 (PSW.3) determine which register bank is enabled. In the MCS-51 assembly language, register addressing is indicated by the register symbols R0 through R7, or by a symbolic name defined earlier as a register. The instruction set mnemonic for Register Addressing is "Rn" where n can be any value from 0 to 7.

Direct Addressing

BYTE OPERANDS

Direct Byte addressing specifies an on-chip RAM location or a Special Function Register. An additional byte is appended to the instruction opcode to provide the memory location address. The highest-order bit of this byte selects one of two groups of addresses: values between 0 and 127 (00H-7FH) access internal RAM locations, while values between 128 and 255 (80H-0FFH) access one of the Special Function Registers. In the assembly language, direct addresses are specified with the address of the variable or register, or a symbolic name defined earlier as a direct address. The instruction set mnemonic for Direct Byte Addressing is "direct".

BIT OPERANDS

Direct Bit addressing (bit) lets a number of instructions manipulate or test any of 128 user-defined software flags in internal data RAM, and manipulate or test 128 bits in the Special Functions Registers address space. An additional byte appended to the opcode specifies the flag or bit to be accessed. Values between 0 and 127 (00H-7FH) correspond to software flags in sixteen internal RAM locations, addresses 20H-2FH. Bit addresses 00H-07H correspond to bits 0-7 of RAM location 20H, respectively; addresses 77H-7FH correspond to bits 0-7 of RAM location 2FH. Bit address values between 128

Table 3-1. Special Function Registers

	Special Function Register	ASM-51	Byte Location in Memory
		Symbolic Name	
	Accumulator*	ACC	224(E0H)
Arithmetic	B register*	B	240(F0H)
Registers	Program Status Word*	PSW	208(D0H)
	Stack Pointer	SP	129(81H)
Pointers	Data Pointer (high)	DPH	131(83H)
	Data Pointer (low)	DPL	130(82H)
	Port 3*	P3	176(B0H)
Parallel	Port 2*	P2	160(A0H)
I/O	Port 1*	P1	144(90H)
Ports	Port 0*	P0	128(80H)
	Interrupt Priority	IPC	184(B8H)
Interrupt	Control*		
System	Interrupt Enable Control*	IEC	168(A8H)
	Timer Mode	TMOD	137(89H)
	Timer Control*	TCON	136(88H)
	Timer 1 (high)	TH1	141(8DH)
Timers	Timer 1 (low)	TH0	139(8BH)
	Timer 0 (high)	TH0	140(8CH)
	Timer 0 (low)	TL0	138(8AH)
Serial I/O	Serial Control*	SCON	152(98H)
Channel	Serial Data Buffer	SBUF	153(99H)

*Bit addressable byte location

b.) Hardware Register Bit Addresses.

Direct Byte Address	(MSB)			Bit Addresses				(LSB)	Hardware Register Symbol
240	F7	F6	F5	F4	F3	F2	F1	F0	B
224	E7	E6	E5	E4	E3	E2	E1	E0	ACC
208	D7	D6	D5	D4	D3	D2	D1	D0	PSW
184	–	–	–	BC	BB	BA	B9	B8	IP
176	B7	B6	B5	B4	B3	B2	B1	B0	P3
168	AF	–	–	AC	AB	AA	A9	A8	IE
160	A7	A6	A5	A4	A3	A2	A1	A0	P2
152	9F	9E	9D	9C	9B	9A	99	98	SCON
144	97	96	95	94	93	92	91	90	P1
136	8F	8E	8D	8C	8B	8A	89	88	TCON
128	87	86	85	84	83	82	81	80	P0

Figure 3-4. Special Function Register Bit Address

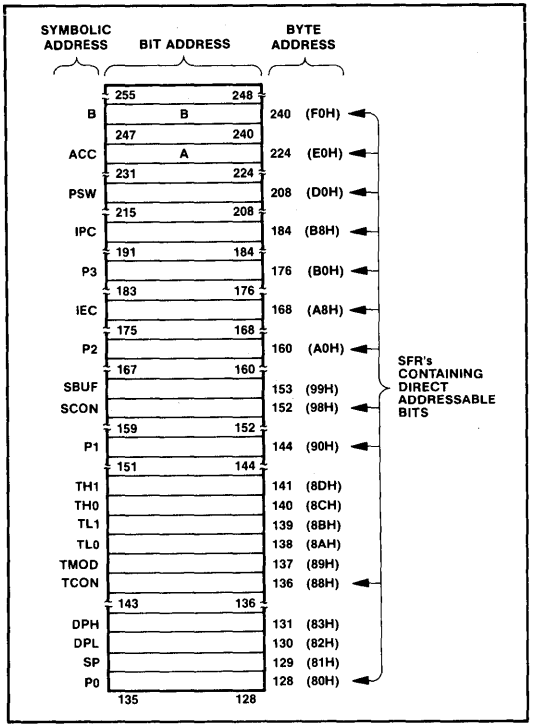


Figure 3-5. Special Function Registers Address Space

and 255 (80H-0FFH) select bits in the special function registers; the high-order five-bit field of the address byte is the same as that of the register used, while the low order three bits give the bit position within that register. The assembly language allows three representations for a direct bit address: by the bit's sequential number (0-255), by specifying the register or RAM location which contains the bit, and the bit's position therein (e.g., 32.6), separated by a period (.) or by a symbol previously defined as a direct bit address. The instruction set signifies a bit location by the mnemonic "bit".

Base-Register-Plus Index Register-Indirect Addressing

Base-Register-plus Index Register-Indirect Addressing simplifies accessing look-up tables (LUT) resident in Program Memory. A byte may be accessed from an LUT via an indirect move from a location whose address is the sum of a base register (the DPTR or PC) and the index register (A).

Register-Indirect Addressing

Register-Indirect Addressing (@Ri) accesses a RAM location whose address is determined by the current contents of R0 or R1, according to bit 0 of the instruction opcode. In the 8051 assembly language, indirect addressing is represented by a commercial "at" sign ("@") preceding R0, R1, or a symbol defined by the user to be equal to R0 or R1. The instruction set mnemonic for Register-Indirect Addressing uses the "at" ("@") also.

Immediate Addressing

Immediate Addressing (#data) appends an additional byte to the instruction to hold the source variable. In the 8051 assembly language and the 8051 instruction set, a number sign (#) precedes the value to be used, which may refer to a constant, an expression, or a symbolic name. Since the value used is fixed at the time of ROM manufacture or EPROM programming, it may not be altered during program execution. (A special case of immediate addressing exists for the instruction MOV DPTR, #data16. Two bytes following the opcode hold the 16 bits of data loaded into the data pointer.)

Figure 3-6 illustrates how the addressing modes reach different Internal Data Memory.

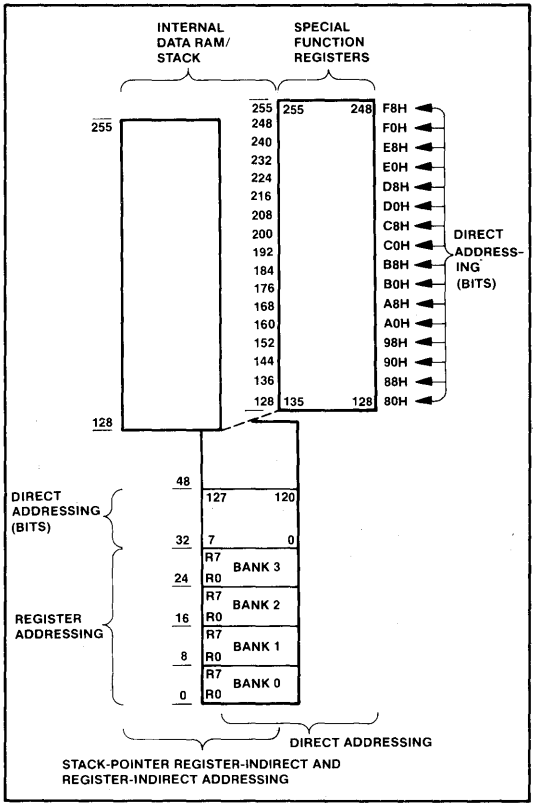


Figure 3-6. Internal Data Memory Addressing Modes

Branch Destination Addressing Modes

Three program memory addressing modes are used by conditional and unconditional branch operations.

RELATIVE ADDRESSING (rel)

Relative addressing (rel) encodes an 8-bit signed displacement value in the last instruction byte. During execution, the CPU computes the destination address by extending the sign-bit of this byte to 16 bits and adding this value to the incremented program counter. In the 8051 assembly language, the programmer only needs to specify the address or label assigned to the desired destination instruction. The assembler will compute the signed displacement needed and produce an error message if the destination is "out of range."

ABSOLUTE ADDRESSING (addr11)

Absolute Addressing (addr11) encodes the low-order 11 bits of the destination address in three bits of the opcode and the second instruction byte. The high-order five bits of the destination address are taken from the high-order five bits of the incremented program counter. Note that this means that an AJMP or ACALL instruction located in addresses 07FEH, for example, will reach a destination between addresses 0800H and 0FFFH.

LONG ADDRESSING (addr16)

Long Addressing (addr16) uses the second and third byte of the instruction to hold the high-order and low-order bytes of the 16-bit destination address, respectively. The destination can be anywhere in the full 64 kilobyte program memory address space.

In the MCS-51 assembly language, only the address or label of the destination instruction is given in each case. The assembler computes the address encoding needed by the operation mnemonic and produces an error message if the destination is "out of range."

Special Addressing Considerations

If an indirect on-chip RAM or stack address is greater than the amount of RAM provided (e.g., greater than 127 on the 8051), or if no special function register corresponds to a direct byte or bit address, then the result of the instruction is undefined.

Note also that the two accumulators, the byte accumulator and the bit accumulator, (the Boolean Processor) can be addressed in two ways using different ASM-51 mnemonics.

When using Register Addressing, the byte accumulator may be reached using the "A" mnemonic, while the bit accumulator may be reached using the "C" mnemonic. If both accumulators are accessed as memory locations using Direct Addressing, different mnemonics are used. "ACC" is the symbol for the byte accumulator and "CY" is the symbol for the bit accumulator.

Even though there are two different addressing modes and a set of mnemonics for each accumulator, both accumulators have only one physical space on the chip.

When an I/O port or pin is the destination of a data move instruction, data is written into a corresponding data latch. When an I/O port or pin is the source for a data transfer, or other two operand instructions, the data present at the input pins is read.

Instructions which use the port as both a source and destination (such as INC P1 or ORL P1, #20H) read the internal buffer rather than the input pins, so only the desired output latch bits will be affected.

When an I/O pin is the destination of a SETB, CLR, CPL, or MOV instruction, the on-chip data latch corresponding to that pin is affected. When an I/O pin is the source operand for a Boolean move or two-operand instruction, the instruction reads the data present at the input pin. The CPL and JBC instructions read the internal buffer rather than the input pin state.

Since the parity flag (PSW.0) is updated after every instruction cycle, instructions which explicitly alter the PSW or this bit will have no apparent effect on P, as if PSW.0 is a read-only bit. Bits 7, 6, and 5 of register IPC, and bits 6 and 5 of register IEC are not implemented on the 8051 and are reserved.

INSTRUCTION SET OVERVIEW

The MCS-51 instruction set includes 51 fundamental operations broken into five functional groupings. Combining them with various addressing modes for Boolean (1-bit), nibble (4-bit), byte (8-bit), and address (16-bit) data types produces the 111 instructions listed in Table 3-2.

Each assembly language instruction consists of an operation mnemonic and (depending on the operation) up to four operands. The mnemonic abbreviates the basic function or operation to be performed, while the operands, separated by commas, clarify which variables are involved, what data to use, or what instruction to execute next. Instructions which need two data operands always specify the destination first, followed by the source variable, except for the move operation between two Directly addressable bytes. In this case the source operand is first and the destination operand is second.

Table 3-2. 8051 Instruction Set Summary

Interrupt Response Time: To finish execution of current instruction, respond to the interrupt request, push the PC and to vector to the first instruction of the interrupt service program requires 38 to 81 oscillator periods (3 to 7μs @ 12 MHz).

INSTRUCTIONS THAT AFFECT FLAG SETTINGS[1]

INSTRUCTION	FLAG			INSTRUCTION	FLAG		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

[1]Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e. the PSW or bits in the PSW) will also affect flag settings.

Notes on instruction set and addressing modes:

- Rn — Register R7-R0 of the currently selected Register Bank.
- data — 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e. I/O port, control register, status register, etc. (128-255)].
- @Ri — 8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.
- #data — 8-bit constant included in instruction.
- #data 16 — 16-bit constant included in instruction
- addr16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.
- addr11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.
- rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
- bit — Direct Addressed bit in Internal Data RAM or Special Function Register.
- * — New operation not provided by 8048/8049.

ARITHMETIC OPERATIONS

	Mnemonic	Description	Byte	Oscillator Period
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect Ram to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract register from Acc with borrow	1	12

ARITHMETIC OPERATIONS Cont.

	Mnemonic	Description	Byte	Oscillator Period
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from Acc with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment indirect RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Table 3-2. Instruction Set Summary (continued)

ARITHMETIC OPERATIONS Cont.

Mnemonic		Description	Byte	Oscillator Period
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12

LOGICAL OPERATIONS

Mnemonic		Description	Byte	Oscillator Period
ANL	A,Rn	AND register Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12

LOGICAL OPERATIONS Cont.

Mnemonic		Description	Byte	Oscillator Period
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12

Table 3-2. Instruction Set Summary (continued)

DATA TRANSFER

Mnemonic		Description	Byte	Oscillator Period
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A + DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A + PC	Move Code byte relative to PC and Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24

DATA TRANSFER Cont.

Mnemonic		Description	Byte	Oscillator Period
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External Ram (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Table 3-2. Instruction Set Summary (continued)

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Oscillator Period
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to Carry	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24

PROGRAMING BRANCHING

Mnemonic		Description	Byte	Oscillator Period
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return for Subroutine	1	24
RETI		Return for interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24

PROGRAM BRANCHING Cont.

Mnemonic		Description	Byte	Oscillator Period
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Comare immediate to Acc and Jump if Not Equal	3	24
CJNE	RN,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	3	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

Functional Groupings

The five functional groupings are as follows:

ARITHMETIC OPERATIONS

The 8051 implements the arithmetic operations of add, increment, decrement, compare-to-zero, decrement-and-compare-to-zero, decimal-add-adjust, subtract-with-borrow, compare, multiply and divide.

Only unsigned binary integer arithmetic is performed in the Arithmetic/Logic Unit. In the two-operand operations of add, add-with-carry and subtract-with-borrow, the A register (the accumulator) is the first operand and receives the result of the operation. The second operand can be an immediate byte, a register in the selected Register Bank, a Register-Indirect Addressed byte or a Direct Addressed byte. These instructions affect the overflow (OV), carry (C), auxiliary-carry (AC), and parity (P) flags in the Program Status Word (PSW). The carry flag facilitates nonsigned integer arithmetic and multi-precision rotations. Handling two's-complement-integer (signed) addition and subtraction can easily be accommodated with software's monitoring of the PSW's overflow flag. The auxiliary-carry flag simplifies BCD arithmetic. An operation that has an arithmetic aspect similar to a subtract is the compare-and-jump-if-not-equal operation. This operation performs a conditional branch if a register in the selected Register Bank, or a Register-Indirect Addressed byte of Internal Data RAM, does not equal an immediate value; or if the A register does not equal a byte in the Direct Addressable Internal Data RAM, or a Special Function Register. While the destination operand is not updated and neither source operand is affected by the compare operation, the carry flag is set if the first operand is less than the second operand; otherwise it is cleared. A summary of the two-operand add/subtract operations is shown in Figure 3-7.

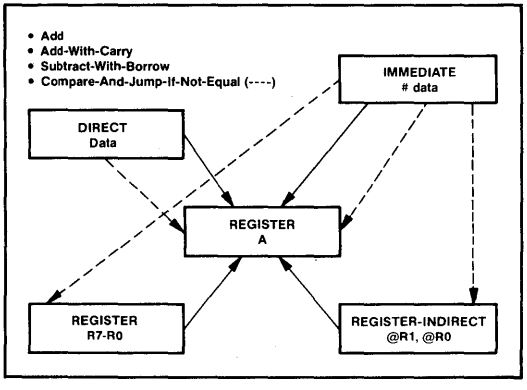


Figure 3-7. Internal Data Memory Arithmetic Operations

There are three arithmetic operations that operate exclusively on the A register (the accumulator). These are the decimal-adjust for BCD addition and the two-test conditions shown in Figure 3-8. The decimal-adjust operation converts the result from a binary addition of two two-digit BCD values to yield the correct two-digit BCD result. During this operation the auxiliary-carry flag helps effect the proper adjustment. Conditional branches may be taken based on the value in the accumulator being zero or not zero.

The 8051 simplifies the implementation of software counters since the increment and decrement operations can be performed on the accumulator, a register in the selected Register Bank, Register-Indirect Addressed byte in the Internal Data RAM or a byte in the Direct Addressed Internal Data RAM or Special Function Register. The 16-bit Data Pointer can be incremented. For efficient loop control, the decrement-and-jump-if-not-zero operation is provided. This operation can decrement a register in the selected Register Bank, any Special Function Register or any byte of Internal Data RAM accessible through Direct Addressing, and force a branch if the result is not zero. The increment/decrement operations are summarized in Figure 3-9.

The multiply operation multiplies the one-byte A register by the one-byte B register and returns a double-byte result

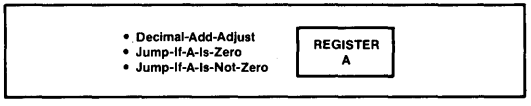


Figure 3-8. Internal Data Memory Arithmetic Operations (Register A Specific)

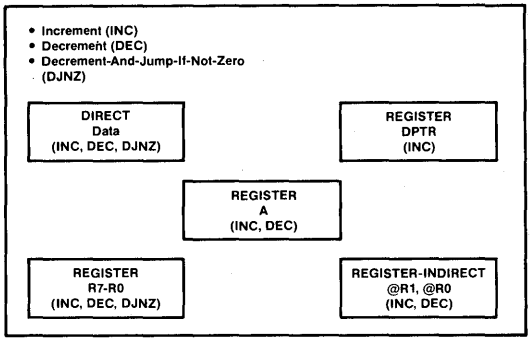


Figure 3-9. Internal Data Memory Arithmetic Operations

(MSB in B, LSB in A). The divide operation divides the one-byte accumulator by the one-byte B register and returns a byte quotient to the A register and a byte remainder to the B register. These are shown in Figure 3-10.

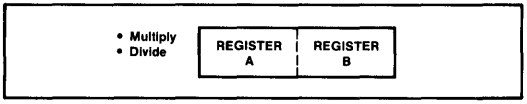


Figure 3-10. Internal Data Memory Arithmetic Operations (Register A with Register B)

LOGIC OPERATIONS

The 8051 permits the logic operations of and, or, and exclusive-or to be performed on the A register by a second operand which can be an immediate value, a register in the selected Register Bank, a Register-Indirect Addressed byte of Internal Data RAM or a Direct Addressed byte of Internal Data RAM or Special Function Register. In addition, these logic operations can be performed on a Direct Addressed byte of the Internal Data RAM or Special Function Register using the A Register as the second operand. Also, use of Immediate Addressing with Direct Addressing permits these logic operations to set, clear or complement any bit anywhere in the Internal Data RAM or Special Function Registers without affecting the PSW, Register Bank registers or accumulator. When one takes into account that the registers R7-R0 and the accumulator can be Direct Addressed, the two-operand logic operations allow the destination (first operand) to be a byte in the Internal Data RAM, a Special Function Register, Register Bank registers (R7-R0) or the accumulator, while the choice of the second operand can be any of the aforementioned or an immediate value. The 8051 can also perform a logical or, or a logical and, between the Boolean accumulator (i.e., the carry register) and any bit, or its complement, that can be accessed through Direct Addressing. The and, or, and exclusive-or logic operations are summarized in Figure 3-11.

In addition to the logic operations that are performed on Internal Data Memory as shown in Figure 3-11, there are also logic operations that are performed specifically on the accumulator. These are summarized in Figure 3-12.

In addition to the "and" and "or" bit logicals shown in Figure 3-11, there are logicals that can operate exclusively on a Direct Addressed bit. These operations are listed in Figure 3-13. The carry flag is also addressed as a register and can be set, cleared, or complemented with one-byte instructions.

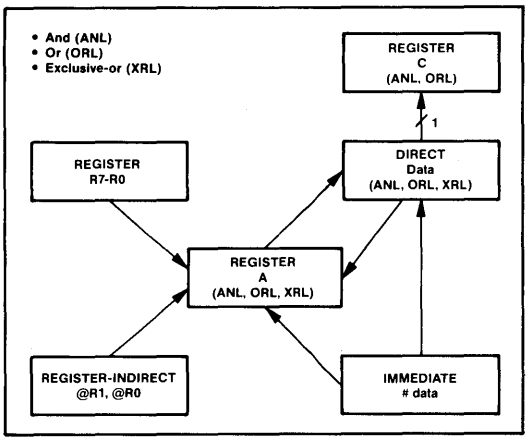


Figure 3-11. Internal Data Memory Logic Operations

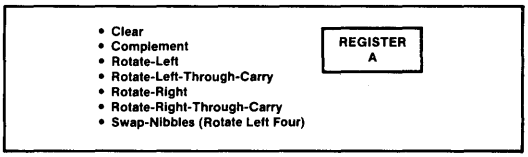


Figure 3-12. Internal Data Memory Logic Operations (Register A Specific)

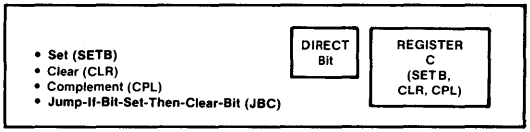


Figure 3-13. Internal Data Memory Logic Operations (Bit-Specific)

DATA TRANSFER OPERATIONS

Look-up tables resident in Program Memory can be accessed by indirect moves. A byte constant can be transferred to the A register (i.e., accumulator) from the Program Memory location whose address is the sum of a base register (the PC or DPTR) and the index register (A). This provides a convenient means for programming translation algorithms such as ASCII to seven segment conversions. The Program Memory move operations are shown diagrammatically in Figure 3-14.

A byte location within a 256-byte block of External Data Memory can be accessed using R1 or R0 in Register-Indirect Addressing. Any location within the full 64K External Data Memory address space can be accessed through Register-Indirect Addressing using a 16-bit base

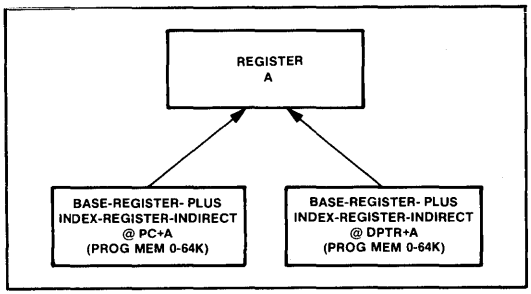


Figure 3-14. Program Memory Move Operations

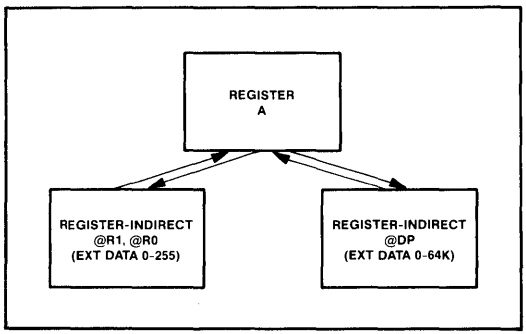


Figure 3-15. External Data Memory Move Operations

register (i.e., the Data Pointer). These moves are shown in Figure 3-15.

The byte in-code-constant (immediate) moves and byte variable moves within the 8051 are highly orthogonal as detailed in Figure 3-16. When one considers that the accumulator and the registers in the Register Banks can be Direct Addressed, the two-operand data transfer operations allow a byte to be moved between any two of the Register Bank registers, Internal Data RAM, accumulator and Special Function Registers. Also, immediate operands can be moved to any of these locations. Of particular interest is the Direct Address to Direct Address move which permits the value in a port to be moved to the Internal Data RAM without using any Register Bank registers or the accumulator. The Data Pointer register can be loaded with a double-byte immediate value.

The A Register can be exchanged with a register in the selected Register Bank, with a Register-Indirect Addressed byte in the Internal Data RAM, or with a Direct Addressed byte in the Internal Data RAM, or Special Function Register. The least significant nibble of the A register can also be exchanged with the least significant nibble of a Register-Indirect Addressed byte in the Internal Data RAM. The exchange operation is shown in Figure 3-17.

BOOLEAN VARIABLE OPERATIONS

A powerful set of instructions perform data transfer, conditional and logical operations on Boolean (1-bit) variables. The 8051's Boolean Processor can move any of 256 bits to or from the carry register (C) using Direct Addressing. Individual instructions will set, clear, or complement these 256 addressable bits or the carry register with Direct Addressing. In conjunction with the bit-test instructions described below, these instructions provide direct 8051 code for logic equations and Boolean expressions.

The carry register is a "Boolean Accumulator" for logical "and" or logical "or" operations on Boolean variables. The carry register acts as a source operand and the destination for the logical operations. The source operand can be one of the 256 addressable bits or its complement.

The 8051 also provides test operations of jump-if-bit-set, jump-if-bit-not-set and jump-if-bit-set-then-clear. These branching instructions are relative to the address of the next instruction (PC + 127 to PC − 128). Jumps can also be taken on the status of the Carry register. A jump can be taken if the carry is set or not set.

CONTROL TRANSFER

The 8051 has a non-paged Program Memory to accommodate relocatable code. The advantage of a non-paged memory is that a minor change to a program that causes a shift of the code's position in memory will not cause page boundary readjustments to be necessary. This also makes relocation possible. Relocation is desirable since it permits several programmers to write relocatable modules in various assembly and high-level languages which can later be linked together to form the machine-object code.

Sixteen-bit jumps and calls are provided to allow branching to any location in the contiguous 64K Program Memory address space and pre-empt the need for Program Memory bank switching. Eleven-bit jumps and calls are also provided to maintain compatibility with the 8048 and to provide an efficient jump within a 2K program module. Unlike the 8048, the 8051's call operations do not push the Program Status Word (PSW) to the stack along with the Program Counter, since many subroutines written for the 8051 do not affect the PSW. Hence the 8051 return operations pop only the Program Counter. The 8051's branch, call and return operations are shown diagrammatically in Figures 3-18, 3-19, and 3-20, respectively.

The 8051 also provides a method for performing condi-

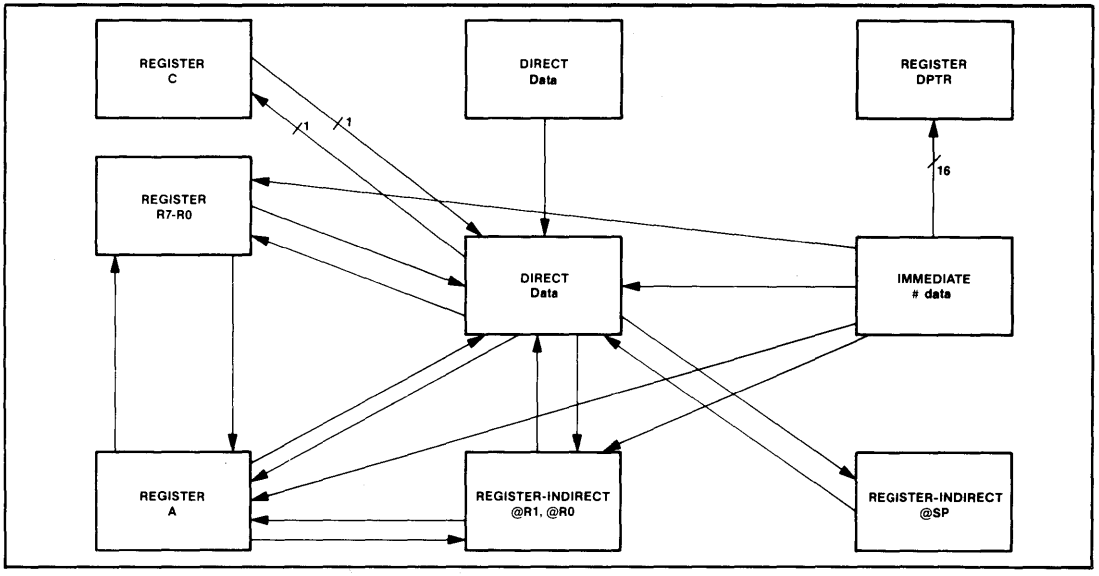


Figure 3-16. Internal Data Memory Move Operations

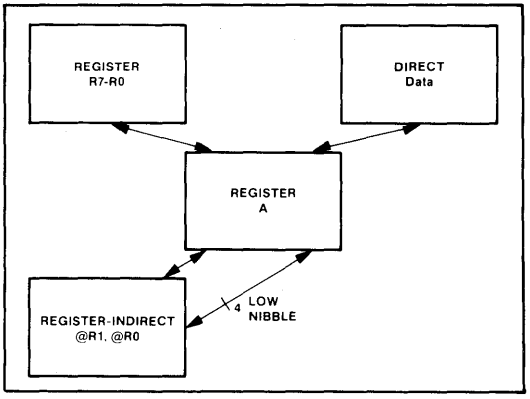


Figure 3-17. Internal Data Memory Exchange Operations

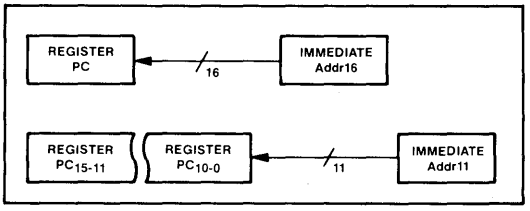


Figure 3-18. Unconditional Branch Operations

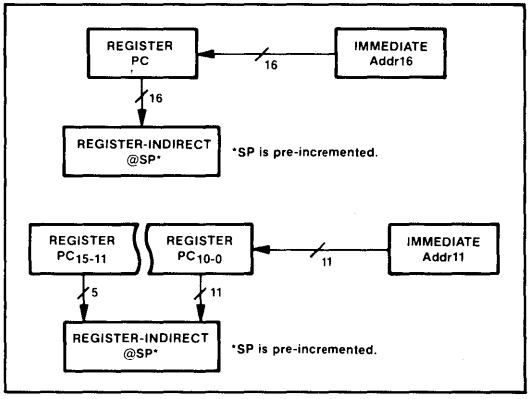


Figure 3-19. Call Operations

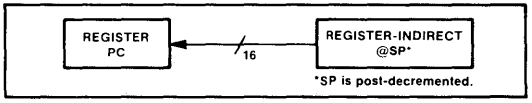


Figure 3-20. Return Operations

tional and unconditional branching, relative to the starting address of the next instruction (PC + 127 to PC − 128). The accumulator test operations allow a conditional branch based on the accumulator being zero or non-zero. Also provided are compare-and-jump-if-not-equal and decrement-and-jump-if-not-zero. These are shown in Figure 3-21. The register-indirect jump in the 8051 permits branching relative to a base register (DPTR) with an offset provided by the non-signed integer value in the index register (A). This accommodates N-way branching. The indirect jump is shown in Figure 3-22.

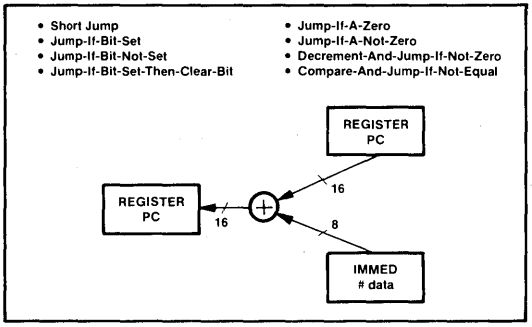


Figure 3-21. Unconditional Short Branch and Conditional Branch Operations

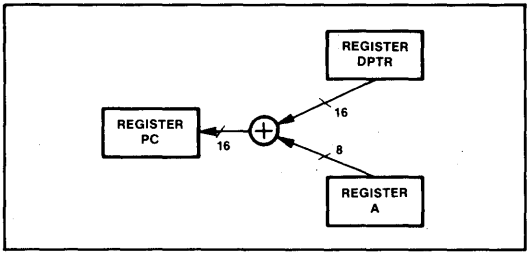


Figure 3-22. Unconditional Branch (Indirect) Operation

Arithmetic Flags

The program status word (PSW) contains eight bits. Four bits are hardware status flags set or cleared by the CPU to show the result of certain calculations. In general, these flags are used for the following purposes:

Carry: The carry flag (CY) is set by an arithmetic instruction if there is a carry-out of the highest order bit (from addition) or if a borrow is needed for the highest-order bit (from subtraction or a comparison); otherwise it is cleared. It is also affected by several rotate operations. The carry flag is also the Boolean accumulator. When treated as a Boolean accumulator, the carry mnemonic is "C"; otherwise it is CY which denotes an address.

Auxiliary Carry: The auxiliary-carry flag (AC) is set if an arithmetic instruction results in a carry-out of bit 3 (from addition) or a borrow into bit 3 (for subtraction); otherwise it is cleared. This flag is useful for BCD arithmetic.

Overflow: The overflow flag (OV) is set if the addition or subtraction of signed variables produces an overflow error (i.e., if the magnitude of the sum or difference is too great for the seven magnitude bits in two's complement representation); otherwise it is cleared. The same flag also indicates when the product resulting from multiplication overflows one byte, and if division by zero was attempted.

Parity: The parity flag (P) is updated after *every* instruction cycle to indicate the parity of the accumulator. It is set if the number of "1" bits in the accumulator is odd, otherwise, it is cleared.

The other four PSW bits consist of a general purpose flag, F0, two bits, RS1 and RS0, which select one of four working register banks, and a reserved bit location.

Instruction Definitions

The rest of this chapter defines all the instructions and operations which the MCS-51 CPU can perform. There is a separate section for each of the 51 basic operations, ordered alphabetically according to the operation mnemonic.

When an operation may apply to more than one data type (generally bit and byte data), the MCS-51 assembly language uses the same mnemonic for each, reducing the number of mnemonics the programmer must remember. The assembler determines which instruction is appropriate from the operands specified. Thus, the mnemonic "CLR" can operate on the eight-bit accumulator ("CLR A"), or on one-bit variables ("CLR F0"). The mnemonics ANL, ORL, CPL, and MOV can relate to more than one data type as well. These operations present each data type in a separate section.

Each section then describes the action taken by the operation, the flags and registers affected, and shows a short example of how an instruction might be used in a program. Next comes the number of bytes and machine cycles required, the corresponding binary machine-language encoding, and a symbolic description or restatement of the function implemented.

Note: Only the carry, auxiliary-carry, and overflow flags are discussed in these instruction descriptions. Since the parity bit (PSW.0) is recomputed *after every instruction cycle* any instruction that alters the accumulator – either inherently or as a special function register – could affect the parity flag. Similarly, instructions which alter directly addressed registers could affect the other status flags if the

instruction is applied to the PSW. Status flags can also be modified by the generalized bit-manipulation instructions.

Nineteen operations allow more than one addressing mode for the source and/or destination operand. The headings for these sections show the instruction format with such operands enclosed in angle brackets (for example, MOV <dest-byte> , <src-byte>). The operation description tells what modes (or combinations of modes) are allowed, and gives the assembly language notation, byte and cycle counts, encoding format, and a symbolic description for each.

The information in this chapter is directed towards defining the capabilities of the MCS-51 architecture and hardware. For details on the assembly language or ASM51 capabilities refer to the *MCS-51 Macro Assembler User's Guide,* publication number 9800937.

ACALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the stack pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345H. After executing the instruction,

```
ACALL    SUBRTN
```

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.

Bytes: 2

Cycles: 2

Encoding:

a10	a9	a8	1	0	0	0	1

a7	a6	a5	a4	a3	a2	a1	a0

Operation: ACALL

(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the accumulator, leaving the result in the accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

will leave 6DH (01101101B) in the accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A,Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	1	r	r	r

Operation: ADD

(A) ← (A) + (Rn)

ADD A,direct

Bytes: 2

Cycles: 1

Encoding:

0	0	1	0	0	1	0	1

direct address

Operation: ADD

(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	1	1	i

Operation: ADD

(A) ← (A) + ((Ri))

ADD A,#data

Bytes: 2

Cycles: 1

Encoding:

0	0	1	0	0	1	0	0

immediate data

Operation: ADD

(A) ← (A) + #data

ADDC A,<src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the accumulator contents, leaving the result in the accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carryout of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the accumulator with AC cleared and both the carry flag and OV set to 1.

ADDC A,Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	1	r	r	r

Operation: ADDC

(A)←(A) + (C) + (R_n)

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding:

0	0	1	1	0	1	0	1

direct address

Operation: ADDC

(A)←(A) + (C) + (direct)

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	1	1	i

Operation: ADDC

(A)←(A) + (C) + ((R_i))

ADDC A,#data

Bytes: 2

Cycles: 1

Encoding:

0	0	1	1	0	1	0	0

immediate data

Operation: ADDC

(A)←(A) + (C) + #data

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123H. The instruction,

```
AJMP    JMPADR
```

is at location 0345H and will load the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10	a9	a8	0	0	0	0	1

a7	a6	a5	a4	a3	a2	a1	a0

Operation:

```
(PC) ← (PC) + 2
(PC10-0) ← page address
```

ANL <dest-byte> , <src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

```
ANL     A,R0
```

will leave 41H (01000001B) in the accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the accumulator at run-time. The instruction,

```
ANL     P1,#01110011B
```

will clear bits 7, 3, and 2 of output port 1.

ANL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	0	1	1	r	r	r

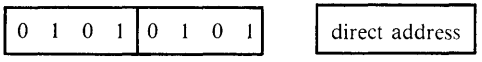
Operation:

```
ANL
(A) ← (A) ∧ (Rn)
```

ANL A,direct

Bytes: 2
Cycles: 1

Encoding:

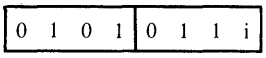


Operation: ANL
(A)←(A) ∧ (direct)

ANL A,@Ri

Bytes: 1
Cycles: 1

Encoding:

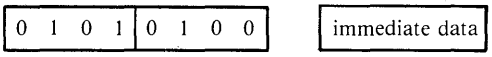


Operation: ANL
(A)← (A) ∧ ((Ri))

ANL A,#data

Bytes: 2
Cycles: 1

Encoding:

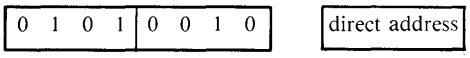


Operation: ANL
(A)← (A) ∧ #data

ANL direct,A

Bytes: 2
Cycles: 1

Encoding:



Operation: ANL
(direct)← (direct) ∧ (A)

ANL direct,#data

Bytes: 3
Cycles: 2

Encoding:

Operation: ANL
(direct)← (direct) ∧ #data

ANL C, <src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct bit addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

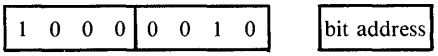
```
MOV   C,P1.0          ;LOAD CARRY WITH INPUT PIN STATE
ANL   C,ACC.7         ;AND CARRY WITH ACCUM. BIT 7
ANL   C,/OV           ;AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C,bit

Bytes: 2

Cycles: 2

Encoding:



Operation: ANL

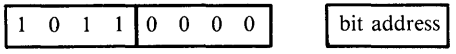
(C) ← (C) ∧ (bit)

ANL C,/bit

Bytes: 2

Cycles: 2

Encoding:



Operation: ANL

(C) ← (C) ∧ ¬ (bit)

CJNE <dest-byte>,<src-byte>,rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
                CJNE    R7,#60H, NOT_EQ
;               . . .   . . . . .           ; R7 = 60H.
NOT_EQ:         JC      REQ_LOW             ; IF R7<60H.
;               . . .   . . . . .           ; R7>60H.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to port 1 is also 34H, then the instruction,

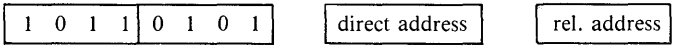
```
WAIT:  CJNE  A,P1,WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

CJNE A,direct,rel

Bytes: 3
Cycles: 2

Encoding:



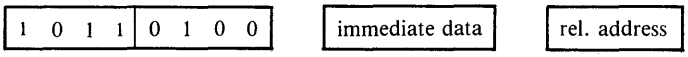
Operation:

```
CJNE
(PC)←(PC) + 3
 IF (direct) < (A)
    THEN  (PC)←(PC) + rel and (C)←0
          OR
 IF (direct) > (A)
    THEN (PC)←(PC) + rel and (C)←1
```

CJNE A,#data,rel

Bytes: 3
Cycles: 2

Encoding:



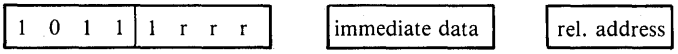
Operation:

```
CJNE
(PC) ←— (PC) + 3
 IF  #data < (A)
     THEN  (PC)←(PC) + rel and (C)←0
           OR
 IF  #data > (A)
     THEN  (PC)←(PC) + rel and (C)←1
```

CJNE Rn,#data,rel

Bytes: 3
Cycles: 2

Encoding:



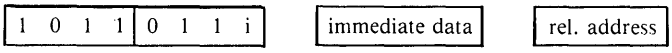
Operation:

```
CJNE
(PC)←——(PC) + 3
 IF  #data < (Rn)
     THEN  (PC)←(PC) + rel and (C)←0
           OR
 IF  #data > (Rn)
     THEN  (PC)←(PC) + rel and (C)←1
```

CJNE @Ri,#data,rel

Bytes: 3
Cycles: 2

Encoding:



Operation:

```
CJNE
(PC)←(PC) + 3
 IF  #data < ((Ri))
     THEN  (PC)←(PC) + rel and (C)←1
           OR
 IF  #data > ((Ri))
     THEN  (PC)←(PC) + rel and (C)←0
```

CLR A

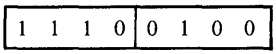
Function: Clear Accumulator
Description: The accumulator is cleared (all bits set to zero). No flags are affected.
Example: The accumulator contains 5CH (01011100B). The instruction,

```
CLR   A
```

will leave the accumulator set to 00H (00000000B).

Bytes: 1
Cycles: 1

Encoding:



Operation:

```
CLR
(A)← 0
```

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction,

```
CLR     P1.2
```

will leave the port set to 59H (01011001B).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	0	1	1

Operation:

```
CLR
(C)←0
```

CLR bit

Bytes: 2

Cycles: 1

Encoding:

1	1	0	0	0	0	1	0

bit address

Operation:

```
CLR
(bit)←0
```

CPL A

Function: Complement Accumulator

Description: Each bit of the accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to zero and vice-versa. No flags are affected.

Example: The accumulator contains 5CH (01011100B). The instruction,

```
CPL     A
```

will leave the accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	0	1	0	0

Operation:

```
CPL
(A)← ¬(A)
```

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Port 1 has previously been written with 5BH (01011101B). The instruction sequence,

```
CPL   P1.1
CPL   P1.2
```

will leave the port set to 5BH (01011011B).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1	0	1	1	0	0	1	1

Operation:

CPL
(C) ← ¬ (C)

CPL bit

Bytes: 2

Cycles: 1

Encoding:

1	0	1	1	0	0	1	0

bit address

Operation:

CPL
(bit) ← ¬ (bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the accumulator, depending on initial accumulator and PSW conditions.

Note: DA A *cannot* simply convert a hexadecimal number in the accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence,

```
ADDC    A,R3
DA      A
```

will first perform a standard twos-complement binary addition, resulting in the value 0BEH (10111110) in the accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the accumulator initially holds 30H (representing the digits of 30 decimal), then the instruction sequence,

```
ADD     A,#99H
DA      A
```

will leave the carry set and 29H in the accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 − 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	0	0

Operation:

```
DA
-contents of Accumulator are BCD
IF  [[(A3-0) >9] ∨ [(AC) = 1]]
    THEN  (A3-0)←(A3-0) + 6
          AND
IF  [[(A7-4) >9] ∨ [(C) = 1]]
    THEN  (A7-4)←(A7-4) + 6
```

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register = indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

```
DEC  @R0
DEC  R0
DEC  @R0
```

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	1	0	0

Operation:

```
DEC
(A) ← (A) – 1
```

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the accumulator by the unsigned eight-bit integer in register B. The accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00H, the values returned in the accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

```
DIV    AB
```

will leave 13 in the accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 x 18) + 17. Carry and OV will both be cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	0	0	0	1	0	0

Operation:

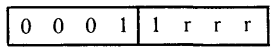
```
DIV
(A)15-8
(B)7-0 ← (A) / (B)
```

DEC Rn

Bytes: 1

Cycles: 1

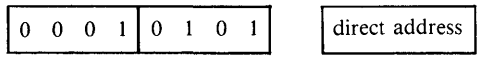
Encoding:



Operation: DEC
(Rn)←(Rn) – 1

DEC direct
Bytes: 2
Cycles: 1

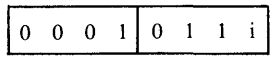
Encoding:



Operation: DEC
(direct)←(direct) – 1

DEC @Ri
Bytes: 1
Cycles: 1

Encoding:



Operation: DEC
((Ri))←((Ri)) – 1

DJNZ <byte>,<rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Internal Ram locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The instruction sequence,

```
DJNZ    40H,LABEL_1
DJNZ    50H,LABEL_2
DJNZ    60H,LABEL_3
```

will cause a jump to the instruction at label LABEL_2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
        MOV   R2,#8
TOGGLE: CPL   P1.7
        DJNZ  R2,TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output port 1. Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

DJNZ Rn,rel

Bytes: 2

Cycles: 2

Encoding:

Operation:

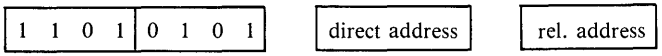
```
DJNZ
(PC)←(PC) + 2
(Rn)←(Rn) − 1
IF  (Rn) > 0 or (Rn) < 0
    THEN
        (PC)←(PC) + rel
```

DJNZ direct,rel

Bytes: 3

Cycles: 2

Encoding:



Operation:

```
DJNZ
(PC)←(PC) + 2
(direct)←(direct) − 1
IF  (direct) > 0 or (direct) < 0
    THEN
        (PC)←(PC) + rel
```

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register = indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7EH (011111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

```
INC  @R0
INC  R0
INC  @R0
```

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes: 1
Cycles: 1

Encoding:

0 0 0 0	0 1 0 0

Operation:
INC
(A)←(A) + 1

INC Rn

Bytes: 1
Cycles: 1

Encoding:

0 0 0 0	1 r r r

Operation:
INC
(Rn)←(Rn) + 1

INC direct

Bytes: 2
Cycles: 1

Encoding:

0 0 0 0	0 1 0 1	direct address

Operation:
INC
(direct)←(direct) + 1

INC @Ri

Bytes: 1
Cycles: 1

Encoding:

0 0 0 0	0 1 1 i

Operation:
INC
((Ri))←((Ri)) + 1

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,

```
INC     DPTR
INC     DPTR
INC     DPTR
```

will change DPH and DPL to 13H and 01H.

Bytes: 1

Cycles: 2

Encoding:

1	0	1	0	0	0	1	1

Operation:

```
INC
(DPTR)←(DPTR) + 1
```

JB bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The accumulator holds 56 (01010110B). The instruction sequence,

```
JB      P1.2,LABEL1
JB      ACC.2,LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0	0	1	0	0	0	0	0	bit address	rel. address

Operation:

```
JB
(PC)←(PC) + 3
IF  (bit) = 1
    THEN
            (PC)←(PC) + rel
```

JBC bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, branch to the address indicated: otherwise proceed with the next instruction. *In either case, clear the designated bit.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.
Note: When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: The accumulator holds 56H (01010110B). The instruction sequence,

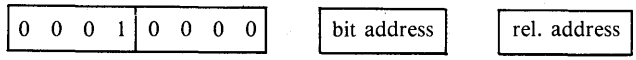
```
JBC     ACC.3,LABEL1
JBC     ACC.2,LABEL2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the accumulator modified to 52H (01010010B).

Bytes: 3

Cycles: 2

Encoding:



Operation:

```
JBC
(PC)←(PC) + 3
IF  (bit) = 1
    THEN
            (bit)←0
            (PC)←(PC) + rel
```

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

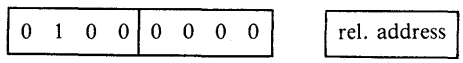
```
JC      LABEL1
CPL     C
JC      LABEL2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
JC
(PC)←(PC) + 2
IF  (C) = 1
    THEN
            (PC)←(PC) + rel
```

JMP @A+DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the accumulator nor the data pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
        MOV     DPTR,#JMP_TBL
        JMP     @A+DPTR
JMP_TBL: AJMP   LABEL0
        AJMP    LABEL1
        AJMP    LABEL2
        AJMP    LABEL3
```

If the accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0	1	1	1	0	0	1	1

Operation:

JMP
(PC)←(A) + (DPTR)

JNB bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The accumulator holds 56H (01010110B). The instruction sequence,

```
JNB     P1.3,LABEL1
JNB     ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0	0	1	1	0	0	0	0	bit address	rel. address

Operation:

JNB
(PC)←(PC) + 3
IF (bit) = 0
THEN (PC)←(PC) + rel.

JNC rel

Function: Jump if Carry not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

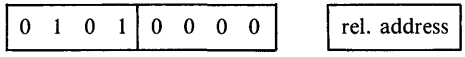
```
JNC   LABEL1
CPL   C
JNC   LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
JNC
(PC)←(PC) + 2
IF  (C) = 0
    THEN  (PC)←(PC) + rel
```

JNZ rel

Function: Jump if accumulator Not Zero

Description: If any bit of the accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The accumulator is not modified. No flags are affected.

Example: The accumulator originally holds 00H. The instruction sequence,

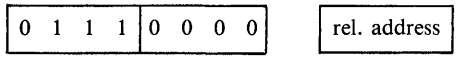
```
JNZ   LABEL1
INC   A
JNZ   LABEL2
```

will set the accumulator to 01H and continue at label LABEL2.

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
JNZ
(PC)←(PC) + 2
IF  (A) ≠ 0
    THEN  (PC)←(PC) + rel
```

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the accumulator are zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The accumulator is not modified. No flags are affected.

Example: The accumulator originally contains 01H. The instruction sequence,

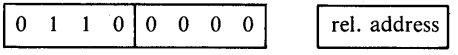
```
JZ    LABEL1
DEC   A
JZ    LABEL2
```

will change the accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:



Operation:

JZ
(PC) ← (PC) + 2
IF (A) = 0
THEN (PC) ← (PC) + rel

LCALL addr16

Function: Long Call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the stack pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K-byte program memory address space. No flags are affected.

Example: Initially the stack pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

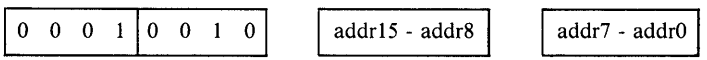
```
LCALL    SUBRTN
```

at location 0123H, the stack pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1235H.

Bytes: 3

Cycles: 2

Encoding:



Operation:

LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC) ← $addr_{15-0}$

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H. The instruction,

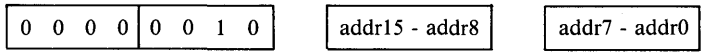
```
LJMP    JMPADR
```

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:



Operation:

LJMP
(PC)←addr$_{15\text{-}0}$

MOV <dest-byte>,<src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH).

```
MOV   R0,#30H          ;R0<= 30H
MOV   A,@R0            ;A <= 40H
MOV   R1,A             ;R1 <= 40H
MOV   R,@R1            ;B <= 10H
MOV   @R1,P1           ;RAM (40H) <= 0CAH
MOV   P2,P1            ;P2 #0CAH
```

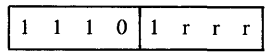
leaves the value 30H in register 0, 40H in both the accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A,Rn

Bytes: 1

Cycles: 1

Encoding:



Operation:

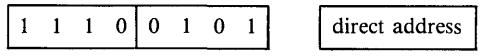
MOV
(A)←(Rn)

MOV A,direct

Bytes: 2

Cycles: 1

Encoding:



Operation:

MOV
(A)←(direct)

MOV A,@RI

Bytes: 1
Cycles: 1

Encoding:

1	1	1	0	0	1	1	i

Operation: MOV
(A) ← ((Ri))

MOV A,#data

Bytes: 2
Cycles: 1

Encoding:

0	1	1	1	0	1	0	0	immediate data

Operation: MOV
(A) ← #data

MOV Rn,A

Bytes: 1
Cycles: 1

Encoding:

1	1	1	1	1	r	r	r

Operation: MOV
(Rn) ← (A)

MOV Rn,direct

Bytes: 2
Cycles: 2

Encoding:

1	0	1	0	1	r	r	r	direct addr.

Operation: MOV
(Rn) ← (direct)

MOV Rn,#data

Bytes: 2
Cycles: 1

Encoding:

0	1	1	1	1	r	r	r	immediate data

Operation: MOV
(Rn) ← #data

MOV direct,A

Bytes: 2
Cycles: 1

Encoding:

1	1	1	1	0	1	0	1	direct address

Operation: MOV
(direct) ← (A)

MOV direct,Rn

Bytes: 2
Cycles: 2

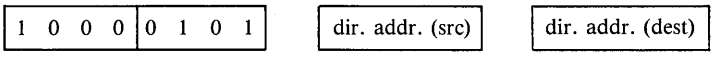
Encoding:

1	0	0	0	1	r	r	r	direct address

Operation: MOV
(direct)←(Rn)

MOV direct,direct
Bytes: 3
Cycles: 2

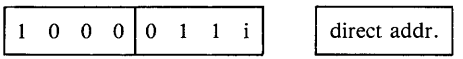
Encoding:



Operation: MOV
(direct)←(direct)

MOV direct,@Ri
Bytes: 2
Cycles: 2

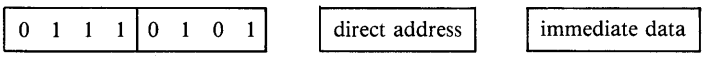
Encoding:



Operation: MOV
(direct)←((Ri))

MOV direct,#data
Bytes: 3
Cycles: 2

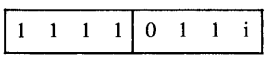
Encoding:



Operation: MOV
(direct)←#data

MOV @Ri,A
Bytes: 1
Cycles: 1

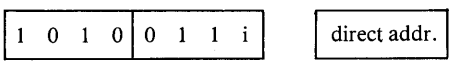
Encoding:



Operation: MOV
((Ri))←(A)

MOV @Ri,direct
Bytes: 2
Cycles: 2

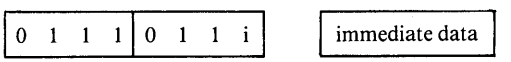
Encoding:



Operation: MOV
((Ri))←(direct)

MOV @Ri,#data
Bytes: 2
Cycles: 1

Encoding:



Operation: MOV
((RI))←#data

MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input port 3 is 11000101B. The data previously written to output port 1 is 35H (00110101B).

```
MOV  P1.3,C
MOV  C,P3.3
MOV  P1.2,C
```

will leave the carry cleared and change port 1 to 39H (00111001B).

MOV C,bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 0	0 0 1 0	bit address

Operation:

MOV
(C) ← (bit)

MOV bit,C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address

Operation:

MOV
(bit) ← (C)

MOV DPTR,#data16

Function: Load Data Pointer with a 16-bit constant

Description: The data pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

Example: The instruction,

```
MOV     DPTR,#1234H
```

will load the value 1234H into the data pointer: DPH will hold 12H and DPL will hold 34H.

Bytes: 3

Cycles: 2

Encoding:

1 0 0 1	0 0 0 0	immed. data15 - 8	immed. data7 - 0

Operation:

MOV
(DPTR) ← #data$_{15\text{-}0}$

MOVC A,@A+<base-reg>

Function: Move Code byte

Description: The MOVC instructions load the accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit accumulator contents and the contents of a sixteen-bit base register, which may be either the data pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the accumulator: otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the accumulator. The following instructions will translate the value in the accumulator to one of four values defined by the DB (define byte) directive.

```
REL_PC: INC     A
        MOVC    A,@A+PC
        RET
        DB      66H
        DB      77H
        DB      88H
        DB      99H
```

If the subroutine is called with the accumulator equal to 01H, it will return with 77H in the accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the accumulator instead.

MOVC A,@A+DPTR

Bytes: 1

Cycles: 2

Encoding:

1	0	0	1	0	0	1	1

Operation: MOVC
(A)←((A) + (DPTR))

MOVC A,@A+PC

Bytes: 1

Cycles: 2

Encoding:

1	0	0	0	0	0	1	1

Operation: MOVC
(PC)←(PC) + 1
(A)←((A) + (PC))

MOVX <dest-byte>,<src-byte>

Function: Move External

Description: The MOVX instructions transfer data between the accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the data pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. P2 retains the high-order bits; any data previously on P2 is lost. This form is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the data pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256 byte RAM using multiplexed address/data lines (e.g., an Intel® 8155 RAM/I/O/Timer) is connected to the 8051 Port 0. Port 3 provides control ones for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

```
MOVX    A,@R1
MOVX    @R0,A
```

copies the value 56H into both the accumulator and external RAM location 12H.

MOVX A,@Ri

Bytes: 1

Cycles: 2

Encoding:

1	1	1	0	0	0	1	i

Operation:
MOVX
(A)←((Ri))

MOVX A,@DPTR

Bytes: 1

Cycles: 2

Encoding:

1	1	1	0	0	0	0	0

Operation:
MOVX
(A)←((DPTR))

MOVX @Ri,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	1	i

Operation:
MOVX
((Ri))←(A)

MOVX @DPTR,A

Bytes: 1
Cycles: 2

Encoding:

1	1	1	1	0	0	0	0

Operation: MOVX
(DPTR)←(A)

MUL A

Function: Multiply
Description: MUL AB multiplies the unsigned eight-bit integers in the accumulator and register B. The low-order byte of the sixteen-bit product is left in the accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.
Example: Originally the accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H). The instruction,

```
MUL     AB
```

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1
Cycles: 4

Encoding:

1	0	1	0	0	1	0	0

Operation: MUL
$(B)_{15-8}$←(A) X (B)
$(A)_{7-0}$

NOP

Function: No Operation
Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.
Example: It is desired to produce a low-going output pulse on bit 7 of port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence,

```
CLR     P2.7
NOP
NOP
NOP
NOP
SETB    P2.7
```

Bytes: 1
Cycles: 1

Encoding:

0	0	0	0	0	0	0	0

Operation: NOP
(PC)←(PC) + 1

ORL <dest-byte> <src-byte>

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,

```
ORL     A,R0
```

will leave the accumulator holding the value 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the accumulator at run-time. The instruction,

```
ORL     P1,#00110010B
```

will set bits 5, 4, and 1 of output port 1.

ORL A,Rn

Bytes: 1
Cycles: 1

Encoding:

0	1	0	0	1	r	r	r

Operation:
ORL
(A)←(A) ∨ (Rn)

ORL A,direct

Bytes: 2
Cycles: 1

Encoding:

0	1	0	0	0	1	0	1	direct address

Operation:
ORL
(A)←(A) ∨ (direct)

ORL A,@Ri

Bytes: 1
Cycles: 1

Encoding:

0	1	0	0	0	1	1	i

Operation:
ORL
(A)←(A) ∨ ((Ri))

ORL A,#data

Bytes: 2
Cycles: 1

Encoding:

0	1	0	0	0	1	0	0	immediate data

Operation: ORL
(A)←(A) ∨ #data

ORL direct,A
Bytes: 2
Cycles: 1

Encoding:

0 1 0 0	0 0 1 0	direct address

Operation: ORL
(direct)←(direct) ∨ (A)

ORL direct,#data
Bytes: 3
Cycles: 2

Encoding:

0 1 0 0	0 0 1 1	direct addr.	immediate data

Operation: ORL
(direct)←(direct ∨ #data

ORL C, <src-bit>

Function: Logical-OR for bit variables
Description Set the carry flag if the Boolean value is a logical l; leave the carry in its current state otherwise. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.
Example: Set the carry flag if and only if P1.0 = 1, ACC.7 = 1, or OV = 0:

```
MOV  C,P1.0        ;LOAD CARRY WITH INPUT PIN P10
ORL  C,ACC.7       ;OR CARRY WITH THE ACC. BIT 7
ORL  C,/OV         :OR CARRY WITH THE INVERSE OF OV
```

ORL C,bit
Bytes: 2
Cycles: 2

Encoding:

0 1 1 1	0 0 1 0	bit address

Operation: ORL
(C)←(C) ∨ (bit)

ORL C,/bit
Bytes: 2
Cycles: 2

Encoding:

1 0 1 0	0 0 0 0	bit address

Operation: ORL
(C)←(C) ∨ /(bit)

POP direct

Function: Pop from stack

Description: The contents of the internal RAM location addressed by the stack pointer is read, and the stack pointer is decremented by one. The value read is the transfer to the directly addressed byte indicated. No flags are affected.

Example: The stack pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP     DPH
POP     DPL
```

will leave the stack pointer equal to the value 30H and the data pointer set to 0123H. At this point the instruction,

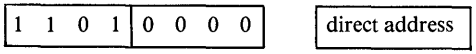
```
POP     SP
```

will leave the stack pointer set to 20H. Note that in this special case the stack pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
POP
(direct)←((SP))
  (SP)←(SP) – 1
```

PUSH direct

Function: Push onto stack

Description: The stack pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the stack pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the stack pointer contains 09H. The data pointer holds the value 0123H. The instruction sequence,

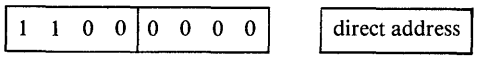
```
PUSH    DPL
PUSH    DPH
```

will leave the stack pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
PUSH
 (SP)←(SP) + 1
((SP))←(direct)
```

RET

Function: Return from subroutine

Description: RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the stack pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The stack pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

```
RET
```

will leave the stack pointer equal to the value 09H. Program execution will continue at location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	0	0	0	1	0

Operation:

```
RET
(PC15-8)←((SP))
(SP)←(SP) – 1
(PC7-0)←((SP))
(SP)←(SP) – 1
```

RETI

Function: Return from interrupt

Description: RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The stack pointer is left decremented by two. No other registers are affected; the PSW is *not* automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The stack pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

```
RETI
```

will leave the stack pointer equal to 09H and return program execution to location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	1	0	0	1	0

Operation:

```
RETI
(PC15-8)←((SP))
(SP)←(SP) – 1
(PC7-0)←((SP))
(SP)←(SP) – 1
```

RL A

Function: Rotate accumulator Left

Description: The eight bits in the accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The accumulator holds the value 0C5H (11000101B). The instruction,

RL A

leaves the accumulator holding the value 8BH (10001011B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	0	1	1

Operation:

RL

$(A_{n+1}) \leftarrow (An)$ $n = 0-6$

$(A0) \leftarrow (A7)$

RLC A

Function: Rotate accumulator Left through the Carry flag

Description: The eight bits in the accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RLC A

leaves the accumulator holding the value 8BH (10001010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1

Operation:

RLC

$(A_{n+1}) \leftarrow (An)$ $n = 0-6$

$(A0) \leftarrow (C)$

$(C) \leftarrow (A7)$

RR A

Function: Rotate accumulator Right

Description: The eight bits in the accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	1

Operation:

RR

$(An) \leftarrow (A_{n+1})$ $n = 0-6$

$(A7) \leftarrow (A0)$

RRC A

Function: Rotate accumulator Right through Carry flag

Description: The eight bits in the accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The accumulator holds the value 0C5H (11000101B), the carry is zero. The instruction,

```
RRC     A
```

leaves the accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	0	1	1

Operation:

RRC
$(An) \leftarrow (A_{n+1})$ $n = 0-6$
$(A7) \leftarrow (C)$
$(C) \leftarrow (A0)$

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output port 1 has been written with the value 34H (00110100B). The instructions,

```
SETB    C
SETB    P1.0
```

will leave the carry flag set to 1 and change the data output on port 1 to 35H (00110101B).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	0	1	1

Operation:

SETB
$(C) \leftarrow 1$

SETB bit

Bytes: 2

Cycles: 1

Encoding:

1	1	0	1	0	0	1	0	bit address

Operation:

SETB
$(bit) \leftarrow 1$

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

```
SJMP    RELADR
```

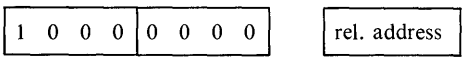
will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(Note: Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
SJMP
(PC)←(PC) + 2
(PC)←(PC) + rel
```

SUBB A, <src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the accumulator, leaving the result in the accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

Example: The source operand allows four addressing modes: register, direct, register-indirect, or immediate. The accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

```
SUBB    A,R2
```

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

SUBB A,Rn

Bytes: 1
Cycles: 1

Encoding:

1	0	0	1	1	r	r	r

Operation: SUBB
(A)←(A) – (C) – (Rn)

SUBB A,direct

Bytes: 2
Cycles: 1

Encoding:

1	0	0	1	0	1	0	1	direct address

Operation: SUBB
(A)←(A) – (C) – (direct)

SUBB A,@Ri

Bytes: 1
Cycles: 1

Encoding:

1	0	0	1	0	1	1	i

Operation: SUBB
(A)←(A) – (C) – ((Ri))

SUBB A,#data

Bytes: 2
Cycles: 1

Encoding:

1	0	0	1	0	1	0	0	immediate data

Operation: SUBB
(A)←(A) – (C) – #data

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leaves the accumulator holding the value 5CH (01011100B).

Bytes: 1
Cycles: 1

Encoding:

1	1	0	0	0	1	0	0

Operation: SWAP
$(A_{3-0}) \rightleftarrows (A_{7-4})$, $(A_{7-4}) \leftarrow (A_{3-0})$

XCH A,<byte>

Function: Exchange Accumulator with byte variable

Description: XCH loads the accumulator with the contents of the indicated variable, at the same time writing the original accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20H. The accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

```
XCH     A,@R0
```

will leave RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the accumulator.

XCH A,Rn

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	1	r	r	r

Operation:

XCH
(A) ⇄ (Rn)

XCH A,direct

Bytes: 2

Cycles: 1

Encoding:

1	1	0	0	0	1	0	1

direct address

Operation:

XCH
(A) ⇄ (direct)

XCH A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	1	1	i

Operation:

XCH
(A) ⇄ ((Ri))

XCHD A,@Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the accumulator (bits 3-0), generally representing a hexadecimal or BCD digit) with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

```
XCHD    A,@R0
```

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the accumulator.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	1	i

Operation:

XCHD
$(A_{3-0}) \rightleftarrows ((Ri_{3-0}))$

XRL <dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

(Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.)

Example: If the accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

```
XRL     A,R0
```

will leave the accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the accumulator at run-time. The instruction,

```
XRL     P1,#00110001B
```

will complement bits 5, 4, and 0 of output port 1.

XRL A,Rn

Bytes: 1
Cycles: 1

Encoding:

0	1	1	0	1	r	r	r

Operation:
XRL
(A)←(A) ⊕ (Rn)

XRL A,direct

Bytes: 2
Cycles: 1

Encoding:

0	1	1	0	0	1	0	1

direct address

Operation:
XRL
(A)←(A) ⊕ (direct)

XRL A,@Ri

Bytes: 1
Cycles: 1

Encoding:

0	1	1	0	0	1	1	i

Operation:
XRL
(A)←(A) ⊕ ((Ri))

XRL A,#data

Bytes: 2
Cycles: 1

Encoding:

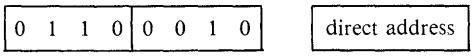
0	1	1	0	0	1	0	0

immediate data

Operation: XRL
(A)←(A) ⊕ #data

XRL direct,A
Bytes: 2
Cycles: 1

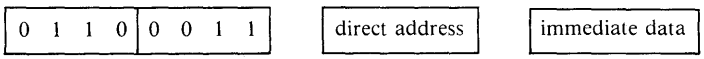
Encoding:



Operation: XRL
(direct)←(direct) ⊕ (A)

XRL direct,#data
Bytes: 3
Cycles: 2

Encoding:



Operation: XRL
(direct)←(direct) ⊕ #data

Expanded 8051 Family

4

Chapter 4
EXPANDED 8051 FAMILY

This chapter shows in very general terms some basic circuits for expanding the 8051 Family. As the product matures and Intel tests specific circuits, the User Manual will be updated. Also application notes will be published to help show actual, tested circuits designed by Intel personnel. The schematics included in this chapter should give the designer an insight into connecting external peripherals and memories to the 8051.

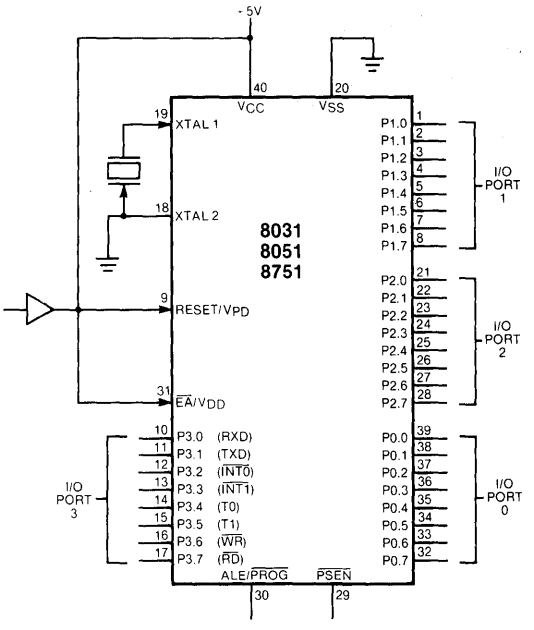
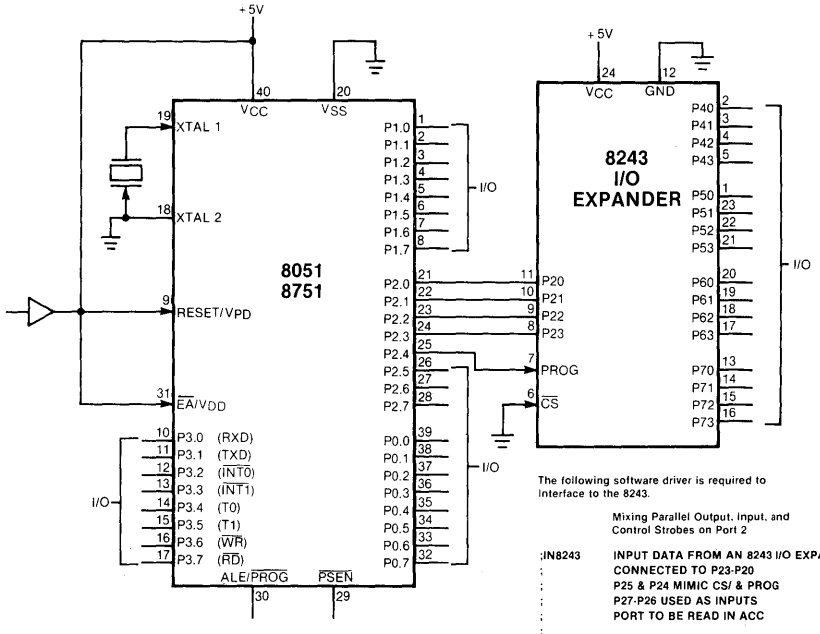


Figure 4-1. The Standalone 8051



The following software driver is required to Interface to the 8243.

Mixing Parallel Output, Input, and Control Strobes on Port 2

```
;IN8243     INPUT DATA FROM AN 8243 I/O EXPANDER
;           CONNECTED TO P23-P20
;           P25 & P24 MIMIC CS/ & PROG
;           P27-P26 USED AS INPUTS
;           PORT TO BE READ IN ACC
;
IN8243:     ORL     A,#11010000B
            MOV     P2,A        ;OUTPUT INSTRUCTION CODE
            CLR     P2,4        ;FALLING EDGE OF PROG
            ORL     P2,#00001111B    ,SET FOR INPUT
            MOV     A,P2        ;READ INPUT DATA
            SETB    P2,4        ;RETURN PROG HIGH
```

Figure 4-2. I/O Expansion Using an 8243

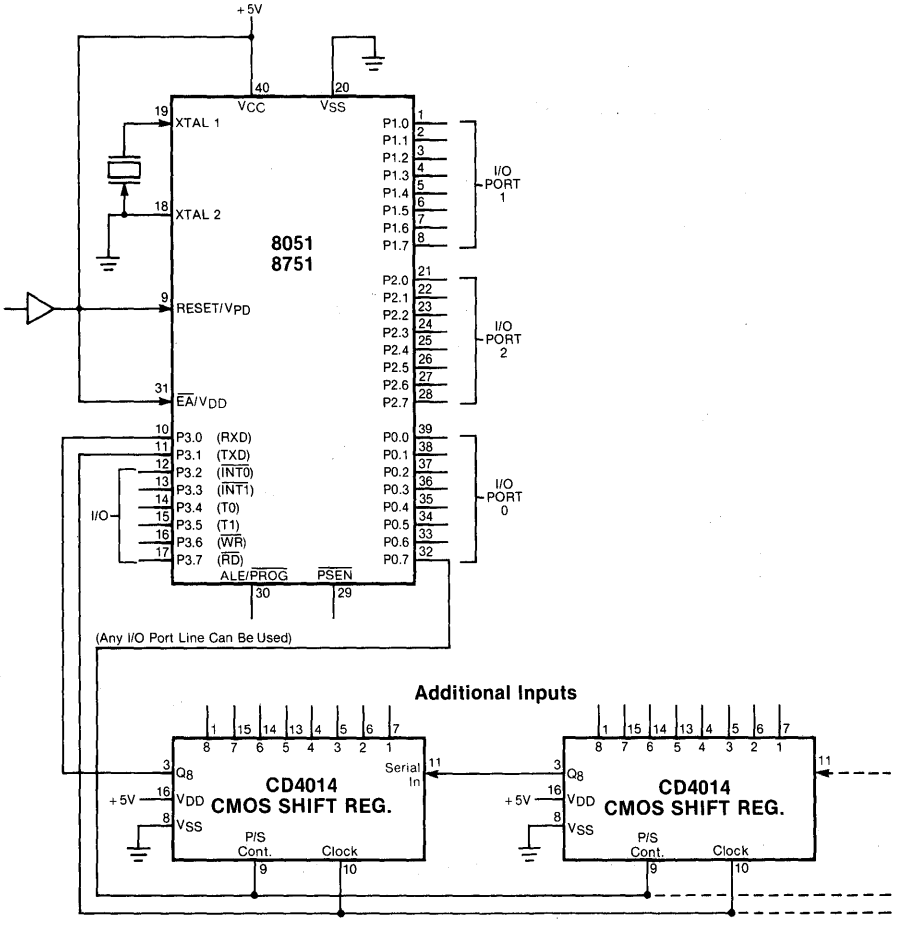


Figure 4-3. Expanding Input Lines via Serial Port

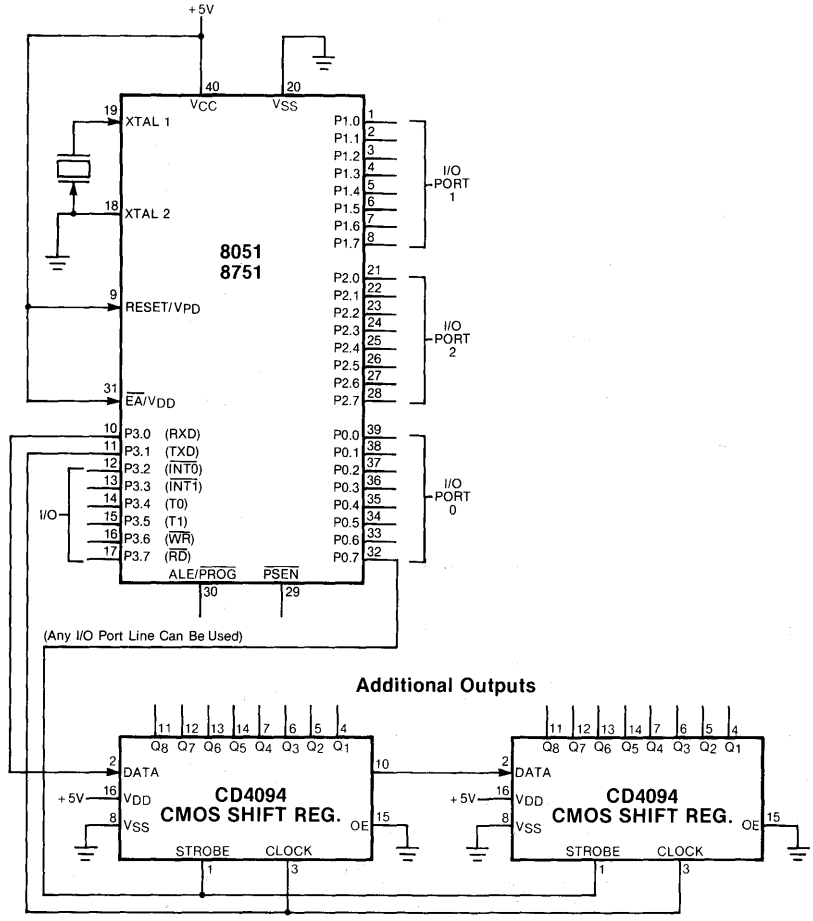
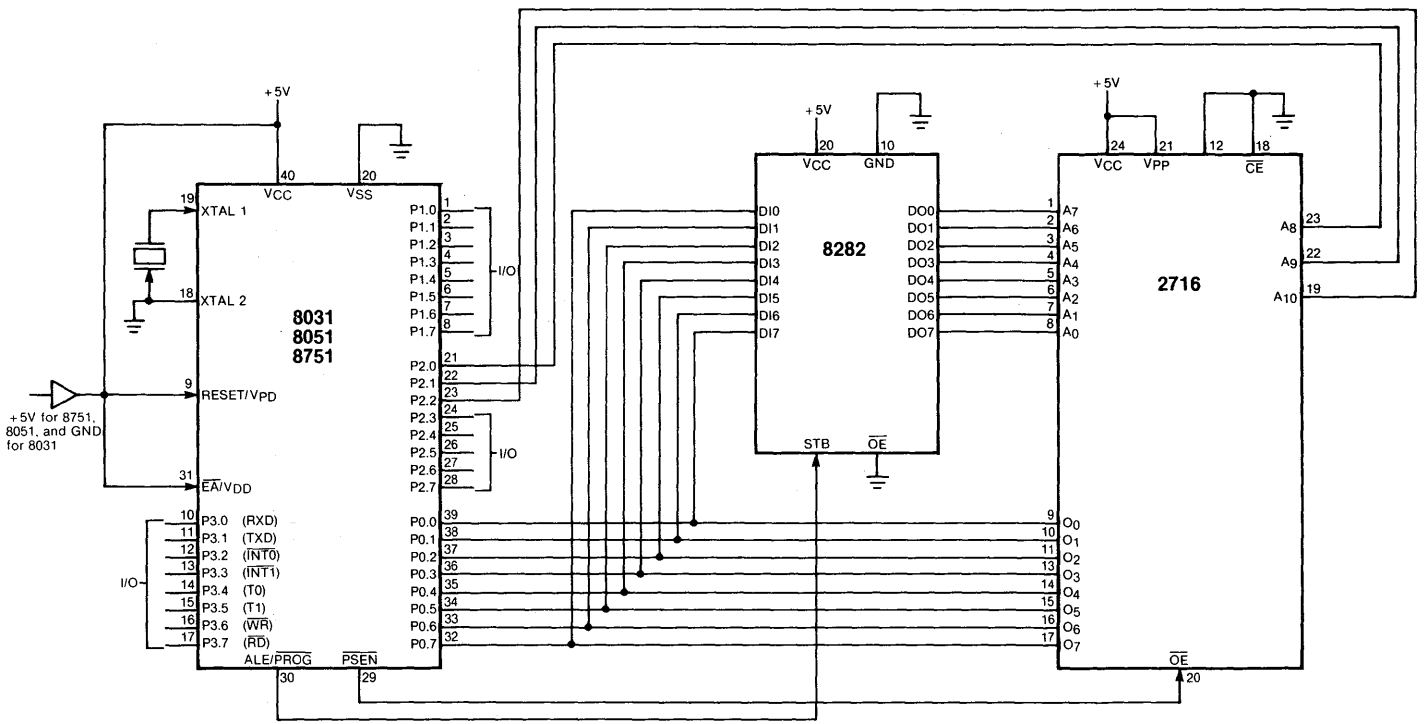


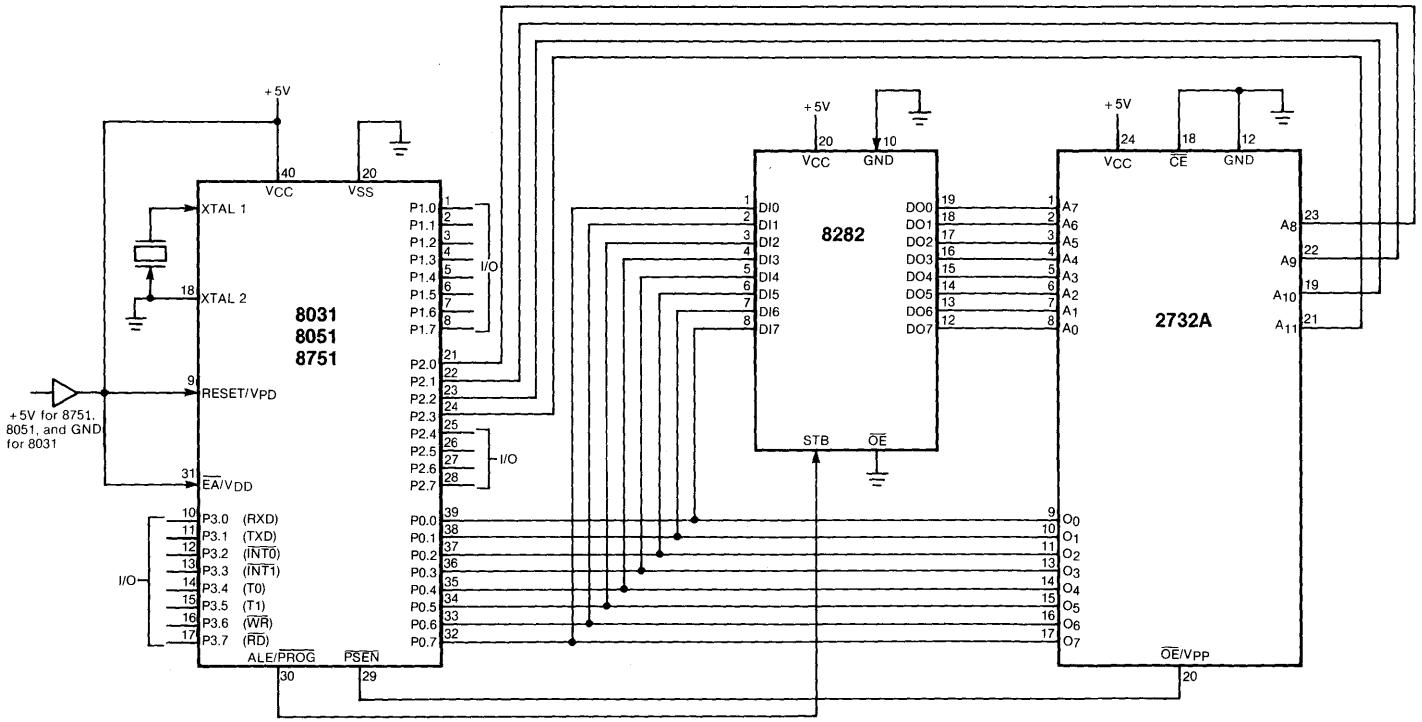
Figure 4-4. Expanding Output Lines via Serial Port



For 8751, 8051 $\overline{CE}$ of the 2716 must be driven by P2.4 through an inverter

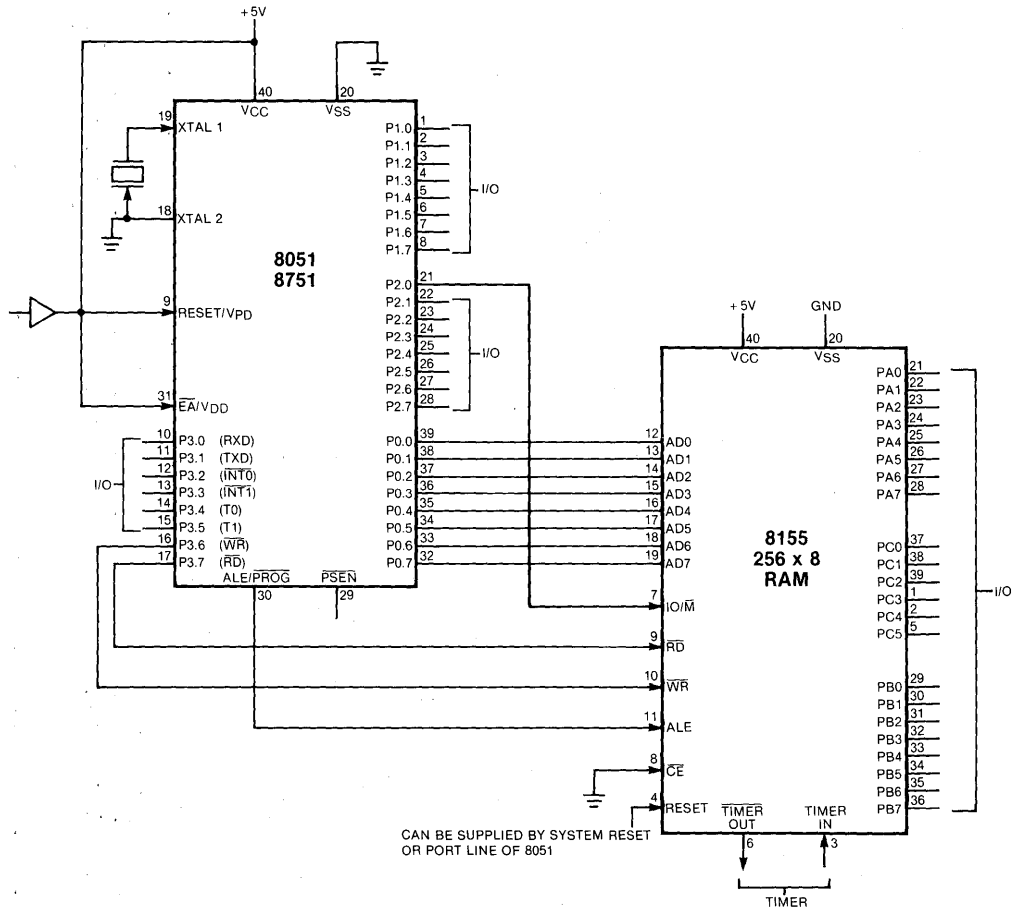
Figure 4-5. External Program Memory Using a 2716

AFN-01739A



For 8751, 8051 $\overline{CE}$ on 2732A must be driven by P2.4 through an Inverter.

Figure 4-6. External Program Memory Using a 2732



- **Both I/O and RAM are addressed as data memory.**
- **Writing a bit to P2.0 determines whether RAM or I/O is to be accessed.**

Figure 4-7. Adding a Data Memory and I/O Expander

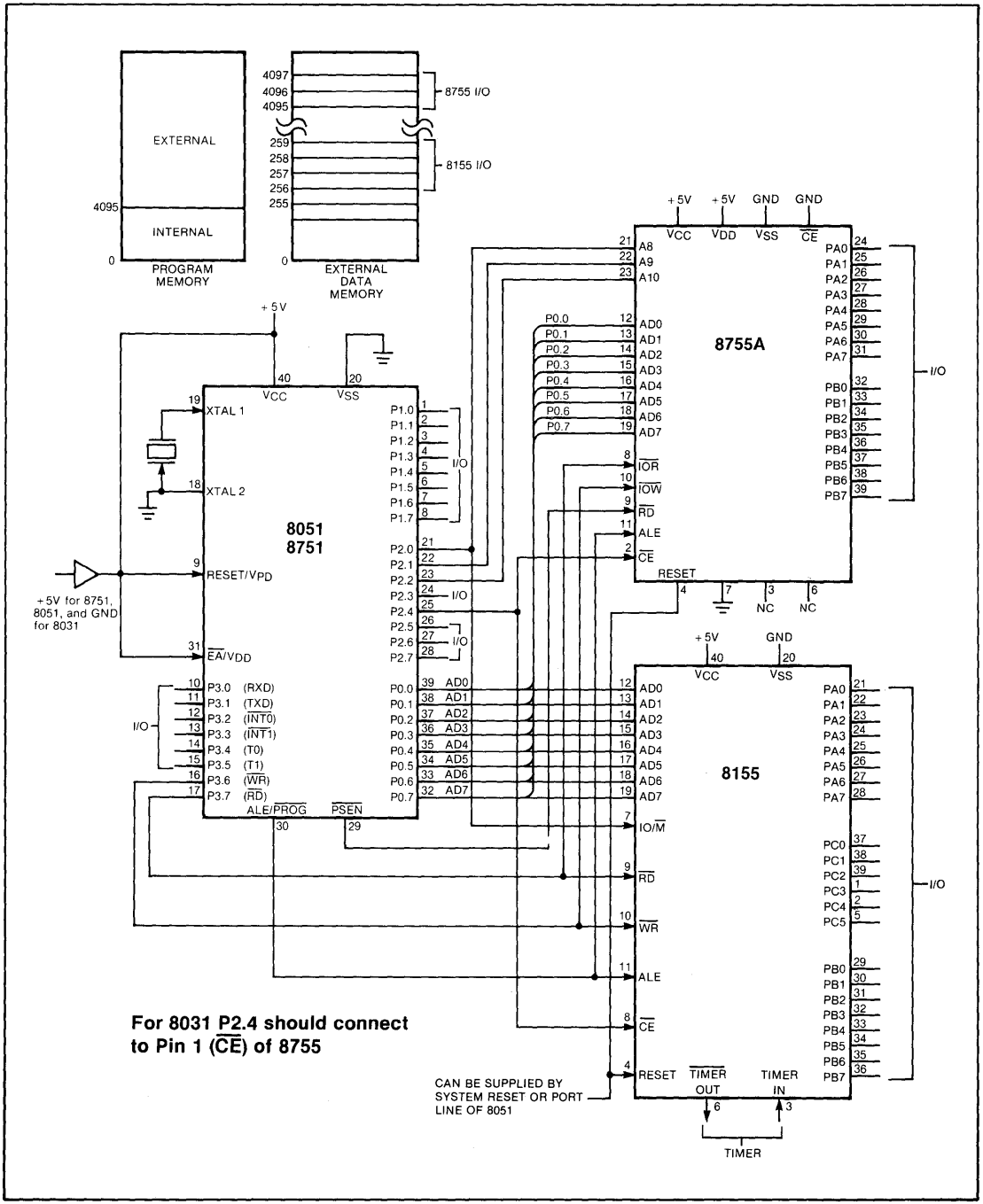


Figure 4-8. The Three-Chip System

AFN-01739A

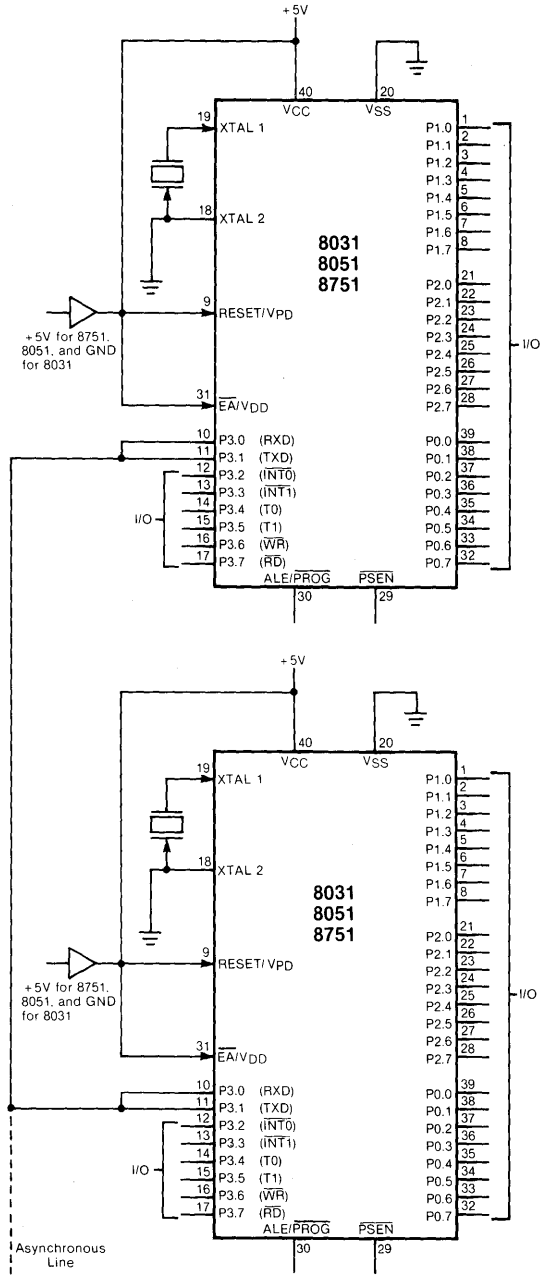
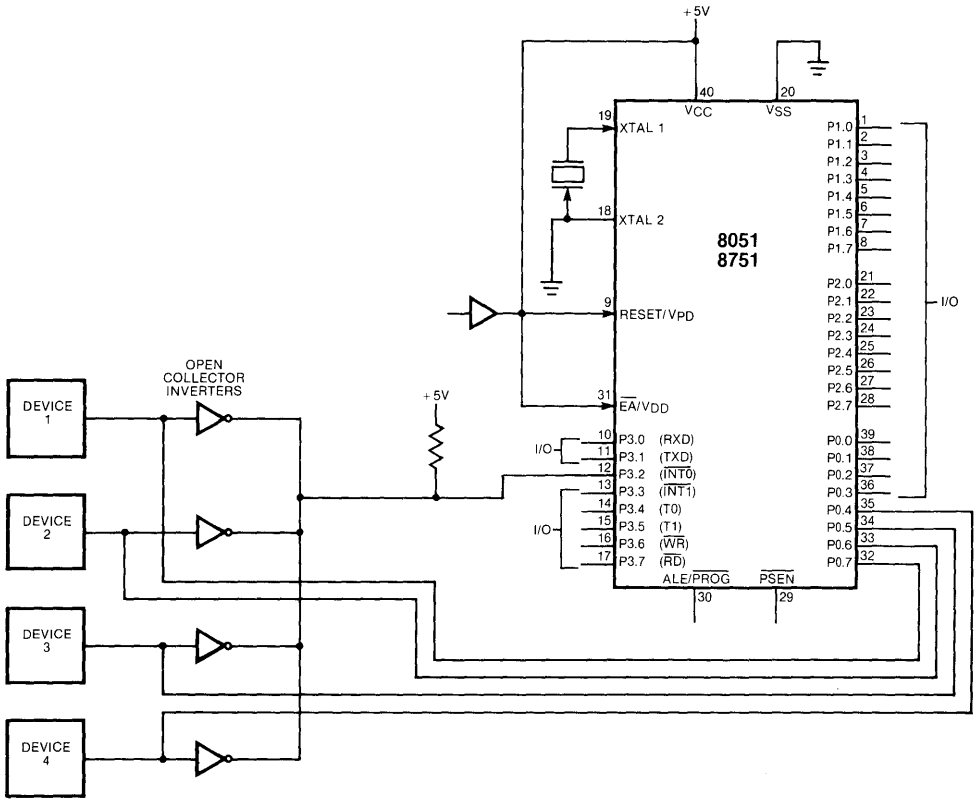


Figure 4-9. Multiple 8051's Using Half-Duplex Serial Communication



- All devices equal priority
- Processor polls Port 0 to determine interrupting device

Figure 4-10. Multiple Interrupt Sources

8051 Software Routines

5

CHAPTER 5
8051 SOFTWARE ROUTINES

Chapter 5 contains two sections:

- 8051 Programming Techniques
- Peripheral Interfacing Techniques

The first section has 8051 software examples for some common routines in controller applications. Some routines included are multiple-precision arithmetic and table look-up techniques.

Peripheral Interfacing Techniques include routines for handling the 8051's I/O ports, serial channel and timer/counters. Discussed in this section is I/O port reconfiguration, software delay timing, and transmitting serial port character strings along with other routines.

8051 PROGRAMMING TECHNIQUES

Radix Conversion Routines

The divide instruction can be used to convert a number from one radix to another. BINBCD is a short subroutine to convert an eight-bit unsigned binary integer in the accumulator (between 0 & 255) to a three-digit (two byte) BCD representation. The hundred's digit is returned in one variable (HUND) and the ten's and one's digits returned as packed BCD in another (TENONE).

```
;
;BINBCD     CONVERT 8-BIT BINARY VARIABLE IN ACCUMULATOR
;           TO 3-DIGIT PACKED BCD FORMAT.
;           HUNDREDS' PLACE LEFT IN VARIABLE 'HUND',
;           TENS' AND ONES' PLACES IN 'TENONE'.
;
HUND       DATA    21H
TENONE     DATA    22H
;
BINBCD:    MOV     B,#100          ;DIVIDED BY 100 TO
           DIV     AB              ;DETERMINE NUMBER OF HUNDREDS
           MOV     HUND,A
           MOV     A,#10           ;DIVIDE REMAINDER BY TEN TO
           XCH     A,B             ;DETERMINE NUMBER OF TENS LEFT
           DIV     AB              ;TEN'S DIGIT IN ACC, REMAINDER IS
                                   ;ONE'S DIGIT
           SWAP    A
           ADD     A,B             ;PACK BCD DIGITS IN ACC
           MOV     TENONE,A
           RET
;
```

The divide instruction can also separate data in the accumulator into sub-fields. For example, dividing packed BCD data by 16 will separate the two nibbles, leaving the high-order digit in the accumulator and the low-order digit (remainder) in B. Each is right-justified, so the digits can be processed individually. This example receives two packed BCD digits in the accumulator, separates the digits, computes their product, and returns the product in packed BCD format in the accumulator.

```
;
;MULBCD     UNPACK TWO BCD DIGITS RECEIVED IN ACCUMULATOR,
;           FIND THEIR PRODUCT, AND RETURN PRODUCT
;           IN PACKED BCD FORMAT IN ACCUMULATOR
;
```

```
MULBCD:  MOV    B,#10H          ;DIVIDE INPUT BY 16
         DIV    AB              ;A & B HOLD SEPARATED DIGITS
                                ;(EACH RIGHT JUSTIFIED IN REGISTER).
         MUL    AB              ;A HOLDS PRODUCT IN BINARY FORMAT (0-
                                ;99 (DECIMAL) = 0 — 63H)
         MOV    B,#10           ;DIVIDE PRODUCT BY 10
         DIV    AB              ;A HOLDS NUMBER OF TENS, B HOLDS
                                ;REMAINDER
         SWAP   A
         ORL    A,B             ;PACK DIGITS
         RET
```

Multiple Precision Arithmetic

The ADDC and SUBB instructions incorporate the previous state of the carry (borrow) flag to allow multiple-precision calculations by repeating the operation with successively higher-order operand bytes. If the input data for a multiple-precision operation is an unsigned string of integers, the carry flag will be set upon completion if an overflow (for ADDC) or underflow (for SUBB) occurs. With two's complement signed data, the most significant bit of the original input data's most significant byte indicates the sign of the string, so the overflow flag (OV) will indicate if overflow or underflow occurred.

```
;
;SUBSTR   SUBTRACT STRING INDICATED BY R1
;         FROM STRING INDICATED BY R0 TO
;         PRECISION INDICATED BY R2.
;         CHECK FOR SIGNED UNDERFLOW WHEN DONE.
;
SUBSTR:  CLR    C               ;BORROW =0.
SUBS1:   MOV    A,@R0           ;LOAD MINUEND BYTE
         SUBB   A,@R1           ;SUBTRACT SUBTRAHEND BYTE
         MOV    @R0,A           ;STORE DIFFERENCE BYTE
         INC    R0              ;BUMP POINTERS TO NEXT PLACE
         INC    R1
         DJNZ   R2,SUBS1        ;LOOP UNTIL DONE
;
;        WHEN DONE, TEST IF OVERFLOW OCCURRED
;        ON LAST ITERATION OF LOOP.
;
         JNB    OV,OV_OK
;        ...    .....           (OVERFLOW RECOVERY ROUTINE)
         OV-OK: RET             ;RETURN
```

Table Look-Up Sequences

The two versions of the MOVC instructions are used as part of a three-step sequence to access look-up tables in ROM. To use the DPTR version, load the Data Pointer with the starting address of a look-up table; load the accumulator with (or compute) the index of the entry desired; and execute MOVC A,@A+DPTR. The data pointer may be loaded with a constant for short tables, or to allow more complicated data structures, and tables with more than 256 entries, the values for DPH and DPL may be computed or modified with the standard arithmetic instruction set.

The PC-based version is used with smaller, "local" tables, and has the advantage of not affecting the data pointer. This makes it useful in interrupt routines or other situations where the DPTR contents might be significant. Again, a look-up sequence takes three steps: load the accumulator with the index; compensate for the offset from the look-up instruction's address to the start of the table by adding that offset to the accumulator; then execute the MOVC A,@A+PC instruction.

As a non-trivial situation where this instruction would be used, consider applications which store large multi-dimensional look-up tables of dot matrix patterns, non-linear calibration parameters, and so on in the linear (one-dimensional) program memory. To retrieve data from the tables, variables representing matrix indices must be converted to the desired entry's memory address. For a matrix of dimensions (MDIMEN x NDIMEN) starting at address BASE and respective indices INDEXI and INDEXJ, the address of element

(INDEXI, INDEXJ) is determined by the formula,

Entry Address = [BASE + (NDIMEN x INDEXI) + INDEXJ]

The subroutine MATRX1 can access an entry in any array with less than 255 elements (e.g., an 11x21 array with 231 elements). The table entries are defined using the Data Byte ("DB") directive, and will be contained in the assembly object code as part of the accessing subroutine itself.

To handle the more general case, subroutine MATRX2 allows tables to be unlimited in size, by combining the MUL instruction, double-precision addition, and the data pointer-based version of MOVC. The only restriction is that each index be between 0 and 255.

```
;
;MATRX1    LOAD CONSTANT READ FROM TWO DIMENSIONAL LOOK-UP
;          TABLE IN PROGRAM MEMORY INTO ACCUMULATOR
;          USING LOCAL TABLE LOOK-UP INSTRUCTION, 'MOVC A,@A+PC'.
;          THE TOTAL NUMBER OF TABLE ENTRIES IS ASSUMED TO
;          BE SMALL, I.E. LESS THAN ABOUT 255 ENTRIES.
;          TABLE USED IN THIS EXAMPLE IS 11 x 21.
;          DESIRED ENTRY ADDRESS IS GIVEN BY THE FORMULA,
;
;          [(BASE ADDRESS) + (21 X INDEXI) + (INDEXJ)]
;
INDEXI    EQU     R6                ;FIRST COORDINATE OF ENTRY (0-10).
INDEXJ    DATA    23H               ;SECOND COORDINATE OF ENTRY (0-20).
;
MATRX1:   MOV     A,INDEXI
          MOV     B, #21
          MUL     AB                ;(21 X INDEXI)
          ADD     A,INDEXJ          ;ADD IN OFFSET WITHIN ROW
;
;         ALLOW FOR INSTRUCTION BYTE BETWEEN "MOVC" AND
;         ENTRY (0,0).
;
          INC     A
          MOVC    A,@A+PC
          RET
BASE1:    DB      1                 ;(entry 0,0)
          DB      2                 ;(entry 0,1)
;         ..      ......
          DB      21                ;(entry 0,20)
          DB      22                ;(entry 1,0)
;         ..      ......
          DB      42                ;(entry 1,20)
;         ..      ......
;         ..      ......
;         DB      231               ;(entry 10,20)
MATRX2:   MOV     A,INDEXI          ;LOAD FIRST COORDINATE
          MOV     B,#NDIMEN
          MUL     AB                :INDEXI X NDIMEN
          ADD     A,#LOW(BASE2)     ;ADD IN 16-BIT BASE ADDRESS
          MOV     DPL,A
          MOV     A,B
          ADDC    A,#HIGH(BASE2)
          MOV     DPH,A             ;DPTR=(BASE ADDR) + (INDEXI+NDIMEN)
          MOV     A,INDEXJ
          MOVC    A,@A+DPTR         ;ADD INDEXJ AND FETCH BYTE
          RET
          ...     ......
```

```
BASE2:  DB      0               ;(entry 0,0)
        DB      0               ;(entry 0,1)
;       ...     ......
        DB      0               ;(entry 0, NDIMEN-1)
        DB      0               ;(entry 1,0)
;       ...     ......
        DB      0               ;(entry 1, NDIMEN-1)
;       ...     ......
;       ...     ......
        DB      0               ;(entry MDIMEN-1, NDIMEN-1)
```

Saving CPU Status during Interrupts

When the 8051 hardware recognizes an interrupt request, program control branches automatically to the corresponding service routine, by forcing the CPU to process a Long CALL (LCALL) instruction to the appropriate address. The return address is stored on the top of the stack. After completing the service routine, an RETI instruction returns the processor to the background program at the point from which it was interrupted.

Interrupt service routines must not change any variable or hardware registers modified by the main program, or else the program may not resume correctly. (Such a change might look like a spontaneous random error. An example of this will be given later in this section, in the second method of I/O port reconfiguration.) Resources used or altered by the service routine (Accumulator, PSW, etc.) must be saved and restored to their previous value before returning from the service routine. PUSH and POP provide an efficient and convenient way to save such registers on the stack.

If the SP register held 1FH when the interrupt was detected, then while the service routine was in progress the stack would hold the registers shown in Figure 5-1; SP would contain 26H. This is the most general case; if the service routine doesn't alter the B-register and data pointer, for example, the instructions saving and restoring those registers could be omitted.

```
;
LOC_TMP EQU     $               ;REMEMBER LOCATION COUNTER
;
        ORG     0003H           ;STARTING ADDRESS FOR INTERRUPT ROUTINE
        LJMP    SERVER          ;JUMP TO ACTUAL SERVICE ROUTINE LOCATE
                                ;ELSEWHERE
;       ...     ......
        ORG     LOC_TMP         ;RESTORE LOCATION COUNTER
SERVER: PUSH    PSW
        PUSH    ACC             ;SAVE ACCUMULATOR (NOTE DIRECT ADDRESS
                                ;NOTATION)
        PUSH    B               ;SAVE B REGISTER
        PUSH    DPL             ;SAVE DATA POINTER
        PUSH    DPH             ;
        MOV     PSW,#00001000B  ;SELECT REGISTER BANK 1
;       ...     ......
;       ...     ......
        POP     DPH             ;RESTORE REGISTERS IN REVERSE ORDER
        POP     DPL
        POP     B
        POP     ACC
        POP     PSW             ;RESTORE PSW AND RE-SELECT ORIGINAL
                                ;REGISTER BANK
        RETI                    ;RETURN TO MAIN PROGRAM AND RESTORE
                                ;INTERRUPT LOGIC
```

Passing Parameters on the Stack

The stack may also pass parameters to and from subroutines. The subroutine can indirectly address the parameters derived from the contents of the stack pointer, or simply pop the stack into registers before processing.

One advantage here is simplicity. Variables need not be allocated for specific parameters, a potentially large number of parameters may be passed, and different calling programs may use different techniques for determining or handling the variables.

For example, the subroutine HEXASC converts a hexadecimal value to ASCII code for its low-order digit. It first reads a parameter stored on the stack by the calling program, then uses the low-order bits to access a local 16-entry look-up table holding ASCII codes, stores the appropriate code back in the stack and then returns. The accumulator contents are left unchanged.

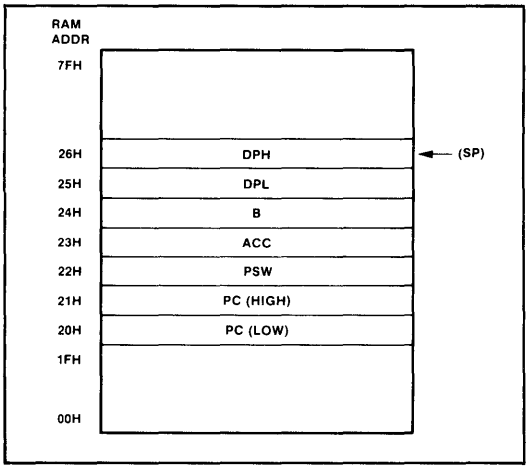


Figure 5-1. Stack contents during interrupt

```
HEXASC:  MOV    R0,SP
         DEC    R0              ;ACCESS LOCATION PARAMETER PUSHED INTO
         DEC    R0
         XCH    A,@R0           ;READ INPUT PARAMETER AND SAVE ACCUMULATOR
         ANL    A,#0FH          ;MASK ALL BUT LOW-ORDER 4 BITS
         ADD    A,#2            ;ALLOW FOR OFFSET FROM MOVC TO TABLE
         MOVC   A,@A+PC         ;READ LOOK-UP TABLE ENTRY
         XCH    A,@R0           ;PASS BACK TRANSLATED VALUE AND RESTORE
                                ;ACCUMULATOR
         RET                    ;RETURN TO BACKGROUND PROGRAM
ASCTBL:  DB     '0'             ;ASCII CODE FOR 00H
         DB     '1'             ;ASCII CODE FOR 01H
         DB     '2'             ;ASCII CODE FOR 02H
         DB     '3'             ;ASCII CODE FOR 03H
         DB     '4'             ;ASCII CODE FOR 04H
         DB     '5'             ;ASCII CODE FOR 05H
         DB     '6'             ;ASCII CODE FOR 06H
         DB     '7'             ;ASCII CODE FOR 07H
         DB     '8'             ;ASCII CODE FOR 08H
         DB     '9'             ;ASCII CODE FOR 09H
         DB     'A'             ;ASCII CODE FOR 0AH
         DB     'B'             ;ASCII CODE FOR 0BH
         DB     'C'             ;ASCII CODE FOR 0CH
         DB     'D'             ;ASCII CODE FOR 0DH
         DB     'E'             ;ASCII CODE FOR 0EH
         DB     'F'             ;ASCII CODE FOR 0FH
```

The background program may reach this subroutine with several different calling sequences, all of which PUSH a value before calling the routine and POP the result to any destination register or port later. There is even the option of leaving a value on the stack if it won't be needed until later. The example below converts the three-digit BCD value computed in the Radix Conversion example above to a three-character string, calling a subroutine SP_OUT to output an eight-bit code in the accumulator.

```
;       ...     ......
        PUSH    HUND
        CALL    HEXASC          ;CONVERT HUNDREDS DIGIT
        POP     ACC
        CALL    SP_OUT          ;TRANSMIT HUNDREDS CHARACTER
        PUSH    TENONE
        CALL    HEXASC          ;CONVERT ONE'S PLACE DIGIT
                                ;BUT LEAVE ON STACK!
        MOV     A, TENONE
        SWAP    A               ;RIGHT-JUSTIFY TEN'S PLACE
        PUSH    ACC             ;CONVERT TEN'S PLACE DIGIT
        CALL    HEXASC
        POP     ACC
        CALL    SP_OUT          ;TRANSMIT TEN'S PLACE CHARACTER
        POP     ACC
        CALL    SP_OUT          ;TRANSMIT ONE'S PLACE CHARACTER
        ...     ......
```

N-Way Branching

There are several different means for branching to sections of code determined or selected at run time. (The single destination addresses incorporated into conditional and unconditional jumps are, of course, fixed at assembly time.) Each has advantages for different applications.

In a typical N-way branch situation, the potential destinations are generally known at assembly time. One of a number of small routines is selected according to the value of an index variable determined while the program is running. The most efficient way to solve this problem is with the MOVC and an indirect jump instruction, using a short table of offset values in ROM to indicate the relative starting addresses of the several routines.

JMP @A + DPTR is an instruction which performs an indirect jump to an address determined during program execution. The instruction adds the eight-bit unsigned accumulator contents with the contents of the sixteen-bit data pointer, just like MOVC A,@A + DPTR. The resulting sum is loaded into the program counter and is used as the address for subsequent instruction fetches. Again, a sixteen-bit addition is performed: a carry-out from the low-order eight-bits may propagate through the higher-order bits. In this case, neither the accumulator contents nor the data pointer is altered.

The example subroutine below reads a byte of RAM into the accumulator from one of four alternate address spaces, as selected by the contents of the variable MEMSEL. The address of the byte to be read is determined by the contents of R0 (and optionally R1). It might find use in a printing terminal application, where four different model printers all use the same ROM code but use different types (and sizes) of buffer memory for different speeds and options.

```
;
MEMSEL  EQU     R3
;
JUMP_4: MOV     A,MEMSEL
        MOV     DPTR,#JMPTBL
        MOVC    A,@A+DPTR
        JMP     @A+DPTR
JMPTBL: DB      MEMSP0-JMPTBL
        DB      MEMSP1-JMPTBL
        DB      MEMSP2-JMPTBL
        DB      MEMSP3-JMPTBL
MEMSP0: MOV     A,@R0           ;READ FROM INTERNAL RAM
        RET
MEMSP1: MOVX    A,@R0           ;READ 256 BYTE EXTERNAL RAM
        RET
MEMSP2: MOV     DPL,R0          ;READ 64K BYTE EXTERNAL RAM
        MOV     DPH,R1
        MOVX    A,@DPTR
        RET
```

```
MEMSP3:  MOV    A,R1              ;READ 4K BYTE EXTERNAL RAM
         ANL    A,#07H
         ANL    P1,#11111000B
         ORL    P1,A
         MOVX   A,@R0
         RET
```

To use this approach, the size of the jump table plus the length of the alternate routines must be less than 256 bytes. The jump table and routines may be located anywhere in program memory and are independent of 256-byte program memory pages.

For applications where up to 128 destinations must be selected, all residing in the same 2K page of program memory, the following technique may be used. In the printing terminal example, this sequence could process 128 different codes for ASCII characters arriving via the 8051 serial port.

```
;
OPTION   EQU    R3
;        ...    ......
;        ...    ......
JMP128:  MOV    A,OPTION
         RL     A                 ;MULTIPLY BY 2 FOR 2-BYTE JUMP TABLE
         MOV    DPTR,#INSTBL      ;FIRST ENTRY IN JUMP TABLE
         JMP    @A+DPTR           ;JUMP INTO JUMP TABLE
;        ...    ......            ;
INSTBL:  AJMP   PROC00            ;128 CONSECUTIVE
         AJMP   PROC01            ;AJMP INSTRUCTIONS
         AJMP   PROC02
;        ...    ......
;        ...    ......
         AJMP   PROC7E
         AJMP   PROC7F
```

The destinations in the jump table (PROC00-PROC7F) are not all necessarily unique routines. A large number of special control codes could each be processed with their own unique routine, with the remaining printing characters all causing a branch to a common routine for entering the character into the output queue.

Computing Branch Destinations at Run Time

In some rare situations, 128 options are insufficient, the destination routines may cross a 2K page boundary, or a branch destination is not known at assembly time (for whatever reason), and therefore cannot be easily included in the assembled code. These situations can all be handled by computing the destination address at run-time with standard arithmetic or table look-up instructions, then performing an indirect branch to that address. There are two simple ways to execute this last step, assuming the 16-bit destination address has already been computed. The first is to load the address into the DPH and DPL registers, clear the accumulator and branch using the JMP @A + DPTR instruction; the second is to push the destination address onto the stack, low-order byte first (so as to mimic a call instruction) then pop that address into the PC by performing a return instruction. This also adjusts the stack pointer to its previous value. The code segment below illustrates the latter possibility.

```
;
RTEMP    EQU    R7
;        ...    ......
JMP256:  MOV    DPTR,#ADRTBL      ;FIRST ADDRESS TABLE ENTRY
         MOV    A,OPTION          ;LOAD INDEX INTO TABLE
         CLR    C
         RLC    A                 ;MULTIPLY BY 2 FOR 2-BYTE JUMP TABLE
         JNC    LOW128
         INC    DPH               ;FIX BASE IF INDEX >127.
```

```
LOW128:  MOV     RTEMP,A          ;SAVE ADJUSTED ACC FOR SECOND READ
         INC     A                ;READ LOW-ORDER BYTE FIRST
         MOVC    A,@A+DPTR        ;GET LOW-ORDER BYTE FROM TABLE
         PUSH    ACC
         MOV     A,RTEMP          ;RELOAD ADJUSTED ACC
         MOVC    A,@A+DPTR        ;GET HIGH-ORDERED BYTE FROM TABLE
         PUSH    ACC
;
;        THE TWO ACC PUSHES HAVE PRODUCED
;        A "RETURN ADDRESS" ON THE STACK WHICH CORRESPONDS
;        TO THE DESIRED STARTING ADDRESS.
;        IT MAY BE REACHED BY POPPING THE STACK
;        INTO THE PC.
         RET
;        ...      ......
;        ...      ......
;        ...      ......
ADRTBL:  DW       PROC00           ;UP TO 256 CONSECUTIVE DATA
         DW       PROC01           ;WORDS INDICATING STARTING ADDRESSES
;        ...      ......
;        ...      ......
         DW       PROCFF
;
```

In-Line-Code Parameter-Passing

Parameters can be passed by loading appropriate registers with values before calling the subroutine. This technique is inefficient if a lot of the parameters are constants, since each would require a separate register to carry it, and a separate instruction to load the register each time the routine is called.

If the routine is called frequently, a more code-efficient way to transfer constants is "in-line-code" parameter-passing. The constants are actually part of the program code, immediately following the call instruction. The subroutine determines where to find them from the return address on the stack, and then reads the parameters it needs from program memory.

For example, assume a utility named ADDBCD adds a 16-bit packed-BCD constant with a two-byte BCD variable in internal RAM and stores the sum in a different two-byte buffer. The utility must be given the constant and both buffer addresses. Rather than using four working registers to carry this information, all four bytes could be inserted into program memory each time the utility is called. Specifically, the calling sequence below invokes the utility to add 1234 (decimal) with the string at internal RAM address 56H, and store the sum in a buffer at location 78H.

The ADDBCD subroutine determines at what point the call was made by popping the return address from the stack into the data pointer high- and low-order bytes. A MOVC instruction then reads the parameters from program memory as they are needed. When done, ADDBCD resumes execution by jumping to the instruction following the last parameter.

```
         ...      ......
         CALL     ADDBCD
         DW       1234H            ;BCD CONSTANT
         DB       56H              ;SOURCE STRING ADDRESS
         DB       78H              ;DESTINATION STRING ADDRESS
         ...      ......           ;CONTINUATION OF PROGRAM
;
;
;
```

```
ADDBCD:  POP   DPH          ;POP RETURN ADDRESS INTO DPTR
         POP   DPL
         MOV   A,#2         ;INDEX FOR SOURCE STRING PARAMETER
         MOVC  A,@A+DPTR    ;GET SOURCE STRING LOCATION
         MOV   R0,A
         MOV   A,#3         ;INDEX FOR DESTINATION STRING PARAMETER
         MOVC  A,@A+DPTR    ;GET DESTINATION ADDRESS
         MOV   R1,A
         MOV   A,#1         ;INDEX FOR 16-BIT CONSTANT LOW BYTE
         MOVC  A,@A+DPTR    ;GET LOW-ORDER VALUE
         ADD   A,@R0        ;COMPUTE LOW-ORDER BYTE OF SUM
         DA    A            ;DECIMAL ADJUST FOR ADDITION
         MOV   @R1,A        ;SAVE IN BUFFER
         INC   R0
         INC   R1
         CLR   A            ;INDEX FOR HIGH-BYTE = 0
         MOVC  A,@A+DPTR    ;GET HIGH-ORDER CONSTANT
         ADDC  A,@R0
         DA    A            ;DECIMAL ADJUST FOR ADDITION
         MOV   @R1,A        ;SAVE IN BUFFER
         MOV   A,#4         ;INDEX FOR CONTINUATION OF PROGRAM
         JMP   @A+DPTR      ;JUMP BACK INTO MAIN PROGRAM
```

This example illustrates several points:

1) The "subroutine" does not end with a normal return statement; instead, an indirect jump relative to the data pointer returns execution to the first instruction following the parameter list. The two initial POP instructions correct the stack pointer contents.

2) Either an ACALL or LCALL works with the subroutine, since each pushes the address of the *next* instruction or data byte onto the stack. The call may be made from anywhere in the full 8051 address space, since the MOVC instruction accesses all 64K bytes.

3) The parameters passed to the utility can be listed in whatever order is most convenient, which may not be that in which they're used. The utility has essentially "random access" to the parameter list, by loading the appropriate constant into the accumulator before each MOVC instruction.

4) Other than the data pointer, the whole calling and processing sequence only affects the accumulator, PSW and pointer registers. The utility could have pushed these registers onto the stack (after popping the parameter list starting address), and popped before returning.

Passing parameters through in-line-code can be used in conjunction with other variable passing techniques.

The utility can also get input variables from working registers or from the stack, and return output variables to registers or to the stack.

PERIPHERAL INTERFACING TECHNIQUES

I/O Port Reconfiguration (First Approach)

I/O ports must often transmit or receive parallel data in formats other than as eight-bit bytes. For example, if an application requires three five-bit latched output ports (called X, Y, and Z), these "virtual" ports could be mapped onto the pins of "physical" ports 1 and 2 as shown below:

	PORT "Z"					PORT "Y"					PORT "X"				
-	PZ0	PZ1	PZ2	PZ3	PZ4	PY4	PY3	PY2	PY1	PY0	PX4	PX3	PX2	PX1	PX0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0

This pin assignment leaves P2.7 free for use as a test pin, input data pin, or control output through software.

Notice that the bits of port Z are reversed. The highest-order port Z pin corresponds to pin P2.2, and the lowest-order pin of port Z is P2.6, due to P.C. board layout considerations. When connecting an 8051 to an immediately adjacent keyboard column decoder or another device with weighted inputs, the corresponding pins may not be aligned. The interconnections must be "scrambled" to compensate either with interwoven circuit board traces or through software (as shown below).

Writing to the virtual ports must not affect any other pins. Since the virtual output algorithms are non-trivial, a subroutine is needed for each port: OUT_PX, OUT_PY and OUT_PZ. Each is called with data to output right-justified in the accumulator, and any data in bits ACC.7-ACC.5 is insignificant. Each subroutine saves the data in a "map" variable for the virtual port, then calls other subroutines which use the data in the various map bytes to compute and output the eight-bit pattern needed for each physical port affected.

```
PX_MAP    DATA    20H
PY_MAP    DATA    21H
PZ_MAP    DATA    22H
;         ...     ....
OUT_PX:   ANL     A,#00011111B      ;CLEAR BITS ACC.7 - ACC. 5
          MOV     PX_MAP,A          ;SAVE DATA IN MAP BYTE
          ACALL   OUT_P1            ;UPDATE PORT 1 OUTPUT LATCH
          RET
;         ....    ....
OUT_PY:   MOV     PY_MAP,A          ;SAVE IN MAP BYTE
          ACALL   OUT_P1            ;UPDATE PORT 1
          ACALL   OUT_P2            ;AND PORT 2 OUTPUT LATCHES
          RET
;         ....    ....
OUT_PZ:   MOV     PZ_MAP,A          ;SAVE DATA IN MAP BYTE
          ACALL   OUT_P2            ;UPDATE PORT 2.
          RET
;         ....    ....
;         ....    ....
OUT_P1:   MOV     A,PY_MAP          ;OUTPUT ALL P1 BITS
          SWAP    A
          RL      A                 ;SHIFT PY_MAP LEFT 5 BITS
          ANL     A,#11100000B      ;MASK OUT GARBAGE
          ORL     A,PX_MAP          ;INCLUDE PX_MAP BITS
          MOV     P1,A
          RET
;         ....    ....
OUT_P2:   MOV     C,PZ_MAP.0        ;LOAD CY WITH P2.6 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          MOV     C,PZ_MAP.1        ;LOAD CY WITH P2.5 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          MOV     C,PZ_MAP.2        ;LOAD CY WITH P2.4 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          MOV     C,PZ_MAP.3        ;LOAD CY WITH P2.3 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          MOV     C,PZ_MAP.4        ;LOAD CY WITH P2.2 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          MOV     C,PZ_MAP.4        ;LOAD CY WITH P2.1 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          MOV     C,PZ_MAP.3        ;LOAD CY WITH P2.0 BIT
          RLC     A                 ;AND SHIFT INTO ACC.
          SETB    ACC.7             ;(ASSUMING INPUT ON P2.7)
          MOV     P2.A
          RET
```

The two level structure of the above subroutines can be modified somewhat if code efficiency and execution speed are critical: incorporate the code shown as subroutines OUT_P1 and OUT_P2 directly into the code for OUT_PX and OUT_PZ, in place of the corresponding ACALL instructions. OUT_PY would not be changed, but now the destinations for its ACALL instructions would be alternate entry points in OUT_PX and OUT_PZ, instead of isolated subroutines.

I/O Port Reconfiguration (Second Approach)

A trickier situation arises if two sections of code which write to the same port or register, or call virtual output routines like those above, need to be executed at different interrupt levels. For example, suppose the background program wants to rewrite Port X (using the port associations in the previous example), and has computed the bit pattern needed for P1. An interrupt is detected just before the MOV P1,A instruction, and the service routine tries to write Port Y. The service routine would correctly update P1 and P2, but upon returning to the background program P1 is immediately *re*-written with the data computed *before* the interrupt! Now pins P2.1 and P2.0 indicate (correctly) data written to port Y in the interrupt routine, but the earlier data written to P1.7-P1.5 is no longer valid. The same sort of confusion could arise if a high-level interrupt disrupted such an output sequence.

One solution is to disable interrupts around any section of code which must not be interrupted (called a "critical section"), but this would adversely affect interrupt latency. Another is to have interrupt routines set or clear a flag ("semaphore") when a common resource is altered – a rather complex and elaborate system.

An easier way to ensure that any instruction which writes the port X field of P1 does not change the port Y field pins from their state *at the beginning of that instruction,* is shown next. A number of 8051 operations read, modify, and write the output port latches all in one instruction. These are the arithmetic and logical instructions (INC, DEC, ANL, ORL, etc.), where an addressed byte is both the destination variable and one of the source operands. Using these instructions, instead of data moves, eliminates the critical section problem entirely.

```
OUT_PX:  ANL   P1,#11100000B     ;CLEAR BITS P1.4 - P1.0
         ORL   P1,A              ;SET P1 PIN FOR EACH ACC BIT SET.
         RET
;        ...   ......
;        ...   ......
OUT_PY:  MOV   B,#20H
         MUL   AB                ;SHIFT <B><A> LEFT 5 BITS.
         ANL   P1,#00011111B     ;CLEAR PY FIELD OF PORT 1
         ORL   P1,A              ;SET PY BITS ON PORT 1
         MOV   A,B               ;LOAD 2 BITS SHIFTED INTO B
         ANL   P2,#11111100B     ;AND UPDATE P2
         ORL   P2,A
         RET
;        ...   ......
;        ...   ......
OUT_PZ:  RRC   A                 ;MOVE ORIGINAL ACC.0 INTO CY
         MOV   P2.6,C            ;AND STORE TO PIN P2.6.
         RRC   A                 ;MOVE ORIGINAL ACC.1 INTO CY
         MOV   P2.5,C            ;AND STORE TO PIN P2.5.
         RCC   A                 ;MOVE ORIGINAL ACC.2 INTO CY
         MOV   P2.4,C            ;AND STORE TO PIN P2.4.
         RRC   A                 ;MOVE ORIGINAL ACC.3 INTO CY
         MOV   P2.3,C            ;AND STORE TO PIN P2.3.
         RRC   A                 ;MOVE ORIGINAL ACC.4 INTO CY
         MOV   P2.2,C            ;AND STORE TO PIN P2.2.
         RET
```

8243 Interfacing

The 8051's quasi-bidirectional port structure lets each I/O pin input data, output data, or serve as a test pin or output strobe under software control. An example of these modes operating in conjunction is the host-processor interface expected by an 8243 I/O expander.

Even though the 8051 does not include 8048-type instructions for interfacing with an 8243, the parts can be interconnected and the protocol may be emulated with simple software; see Figure 5.2.

```
;
;IN8243     INPUT DATA FROM AN 8243 I/O EXPANDER
;           CONNECTED TO P23-P20.
;           P25 & P24 MIMIC CS &PROG.
;           P27-P26 USED AS INPUTS. CODE FOR
;           PORT TO BE READ IN ACC.1-ACC.0
;
PROG        BIT     P2.4            ;SYMBOLIC PIN DESCRIPTION
;
IN8243:     ORL     A,#11010000B    ;SET PROG AND PINS USED AS INPUT
            MOV     P2,A            ;OUTPUT PORT CODE AND OPERATION CODE
            CLR     PROG            ;LOWER PROG TO LATCH ADDRESS
            ORL     P2,#00001111B   ;SET LOW ORDER PINS FOR INPUT
            MOV     A,P2            ;READ IN PORT DATA
            ORL     P2,#00110000B   ;SET PROG AND CS HIGH
```

Software Delay Timing

Many 8051 applications involve exact control over output timing. A software-generated output strobe, for instance, might have to be *exactly* 50 μsec. wide. The DJNZ operation can insert a one instruction software delay into a piece of code, adding a moderate time delay of two instruction cycles per iteration. For example, two instructions can add a 49-μsec. software delay loop to code to generate a pulse on the $\overline{WR}$ pin.

```
CLR     WR
MOV     R2,#24
DJNZ    $2,$
SETB    WR
```

The dollar sign in this example is a special character meaning "the address of this instruction." It can be used to eliminate instruction labels on nearby source lines.

Serial Port and Timer Mode Configuration

Configuring the 8051's Serial Port for a given data rate and protocol requires essentially three short sections of software. On power-up or hardware reset the serial port and timer control words must be initialized to the appropriate values. Additional software is also needed in the transmit routine to load the serial port data register and in the receive routine to unload the data as it arrives.

To choose one arbitrary example, assume the 8051 should communicate with a standard CRT operating at 2400 baud (bits per second). Each character is transmitted as seven data bits, odd parity, and one stop bit. The resulting character rate is 2400 baud/9bits, approximately 265 characters per second.

For the sake of clarity, the transmit and receive subroutines here are driven by simple-minded software status polling code rather than interrupts. The serial port must be initialized to 8-bit UART mode (SM0, SM1 = 01), enabled to receive all messages (SM2=0, REN=1). The flag indicating that the transmit register is free for more data will be artificially set in order to let the output software know the output register is available. All this can be set up with instruction at label SPINIT.

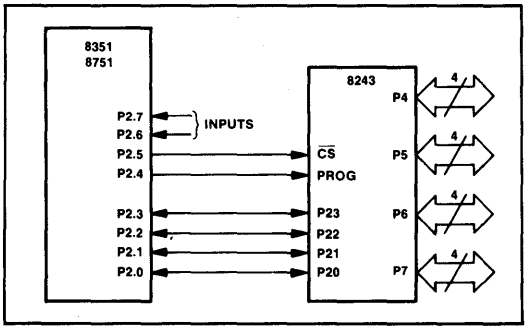


Figure 5-2. Connecting an 8051 with an 8243 I/O Expander

Timer 1 will be used in auto-reload mode as a baud rate generator. To achieve a data rate of 2400 baud, the timer must divide the 1MHz internal clock by

$$\frac{1 \times 10^6}{(32)\ (2400)}$$

which equals 13 (actually, 13.02) instruction cycles. The

timer must reload the value −13, or 0F3H, as shown by the code at label TIINIT. (ASM51 will accept both the signed decimal or hexadecimal representations.)

```
;
;               INITIALIZE SERIAL PORT
;               FOR 8-BIT UART MODE
;               & SET TRANSMIT READY FLAG.
SPINIT:         MOV     SCON,#01010010B
;               INITIALIZE TIMER 1 FOR
;               AUTO-RELOAD AT 32 X 2400HZ
;               (T0 USED AS GATED 16-BIT COUNTER.)
;TIINIT:        MOV     TCON,#11010010B
                MOV     TH1,#-13
                SETB    TR1
;               ...     ......
;
```

Simple Serial I/O Drivers

SP_OUT is a simple subroutine to transmit the character passed to it in the accumulator. First it must compute the parity bit, insert it into the data byte, wait until the transmitter is available, output the character, and then return.

SP_IN is an equally simple routine which waits until a character is received, sets the carry flag if there is an odd-parity error, and returns the masked seven-bit code in the accumulator.

```
;
;SP_OUT      ADD ODD PARITY TO ACC AND
;           TRANSMIT WHEN SERIAL PORT READY.
;
SP_OUT:     MOV     C,P
            CPL     C
            MOV     ACC.7,C
            JNB     TI,$
            CLR     TI
            MOV     SBUF,A
            RET
;
;
;SP_IN       INPUT NEXT CHARACTER FROM SERIAL PORT.
;           SET CARRY IF ODD-PARITY ERROR
;
SP_IN:      JNB     RI,$
            CLR     RI
            MOV     A,SBUF
            MOV     C,P
            CPL     C
            ANL     A,#7FH
            RET
```

Transmitting Serial Port Character Strings

Any application which transmits characters through a serial port to an ASCII output device will on occasion need to output "canned" messages, including error messages, diagnostics, or operator instructions. These character strings are most easily defined with in-line data bytes defined with the DB directive.

```
CR          EQU     0DH             ;ASCII CARRIAGE RET
LF          EQU     0AH             ;ASCII LINE-FEED
ESC         EQU     1BH             ;ASCII ESCAPE CODE
;           ...     ......
            CALL    XSTRING
            DB      CR,LF           ;NEW LINE
            DB      'INTEL DELIVERS'  ;MESSAGE
            DB      ESC             ;ESCAPE CHARACTER
;
;           (CONTINUATION OF PROGRAM)
;
;           ...     ......
XSTRING:    POP     DPH             ;LOAD DPTR WITH FIRST CHARACTER
            POP     DPL
XSTR_1:     CLR     A               ;(ZERO OFFSET)
            MOVC    A,@A+DPTR       ;FETCH FIRST CHARACTER OF STRING
XSTR_2:     JNB     TI,$            ;WAIT UNTIL TRANSMITTER READY
            CLR     TI              ;MARK AS NOT READY
            MOV     SBUF,A          ;OUTPUT NEXT CHARACTER
            INC     DPTR            ;BUMP POINTER
            CLR     A
            MOVC    A,@A+DPTR       ;GET NEXT OUTPUT CHARACTER
            CJNE    A,#ESC,XSTR_2   ;LOOP UNTIL ESCAPE READ
            MOV     A,#1
            JMP     @A+DPTR         ;RETURN TO CODE AFTER ESCAPE
```

Recognizing and Processing Special Cases

Before operating on the data it receives, a subroutine might give "special handling" to certain input values. Consider a word processing device which receives ASCII characters through the 8051 serial port and drives a thermal hard-copy printer. A standard routine translates most printing characters to bit patterns, but certain control characters (<DEL>, <CR>, <LF>, <BEL>,<ESC>, or <SP>) must invoke corresponding special routines. Any other character with an ASCII code less than 20H should be translated into the <NUL> value, 00H, and processed with the printing characters. The CJNE operation provides essentially a one-instruction CASE statement.

```
;
CHAR        EQU     R7              ;CHARACTER CODE VARIABLE
;
INTERP:     CJNE    CHAR,#7FH, INTP_1   ;SKIP UNLESS RUBOUT
;           ...     .....               (SPECIAL ROUTINE FOR RUBOUT CODE)
            RET
INTP_1:     CJNE    CHAR,#07H,INTP_2    ;SKIP UNLESS BELL
;           ...     .....               (SPECIAL ROUTINE FOR BELL CODE)
            RET
INTP_2:     CJNE    CHAR,#0AH,INTP_3    ;SKIP UNLESS LFEED
;           ...     .....               (SPECIAL ROUTINE FOR LFEED CODE)
            RET
INTP_3:     CJNE    CHAR,#0DH,INTP_4    ;SKIP UNLESS RETURN
;           ...     .....               (SPECIAL ROUTINE FOR RETURN CODE)
            RET
INTP_4:     CJNE    CHAR,#1BH,INTP_5    ;SKIP UNLESS ESCAPE
;           ...     .....               (SPECIAL ROUTINE FOR ESCAPE CODE)
            RET
INTP_5:     CJNE    CHAR,#20H,INTP_6    ;SKIP UNLESS SPACE
;           ...     .....               (SPECIAL ROUTINE FOR SPACE CODE)
            RET
INTP_6:     JC      PRINTC              ;JUMP IF CODE > 20 H
            MOV     CHAR,#0             ;REPLACE CONTROL CHARACTER WITH
                                        ;NULL CODE
```

```
PRINTC:                                    ;PROCESS STANDARD PRINTING
;           ...         .....              ;CHARACTER
            RET
;
```

Buffering Serial Port Output Characters

It is not always efficient to transmit characters through the serial port one-at-a-time. Most applications generate a short burst of characters all at once (English words or multi-digit numbers, for instance), with the bursts themselves occurring at longer intervals. Instead of waiting while the UART outputs each character, it would be more efficient if the background program could enter all the characters into a first-in first-out (FIFO) data structure, and continue about its business, letting an interrupt routine transmit each character as the serial port becomes available.

Assume there is a 16-byte output data buffer starting at 70H. QHEAD and QTAIL keep track of the head and tail portion of the buffer being used. The subroutine ENTERQ waits until there is space in the queue, then copies a character code from the accumulator to the queue.

```
QHEAD       DATA    6EH                    ;LAST BYTE ENTERED INTO QUEUE
QTAIL       DATA    6FH                    ;LAST BYTE READ FROM QUEUE.
BOTLIM      EQU     70H
TOPLIM      EQU     7FH
;
;           QUEUE IS EMPTY WHEN QHEAD = QTAIL AND
;           FULL WHEN QHEAD + 1 (WITHIN RANGE) = QTAIL.
            MOV     QHEAD,#TOPLIM
            MOV     QTAIL,#TOPLIM
;
;
ENTERQ:     MOV     R0,A                   ;SAVE ACC DATA
            MOV     A,QHEAD                ;LOAD HEAD POINTER
            INC     A                      ;PRE-INCREMENT POINTER
            CJNE    A,#TOPLIM+1,ENTQ_1
            MOV     A,#BOTLIM              ;RELOAD ON OVERFLOW
ENTQ_1:     CJNE    A,QTAIL,ENTQ_2         ;TEST IF QUEUE FULL
            SJMP    ENTQ_1                 ;LOOP UNTIL SPACE AVAILABLE
ENTQ_2:     XCH     A,R0                   ;STORE POINTER AND RELOAD ACC
            MOV     @R0,A                  ;ENTER INTO QUEUE
            MOV     QHEAD,R0               ;UPDATE HEAD POINTER
            SETB    ES                     ;ENABLE SERIAL PORT INTERRUPTS
            RET
```

The interrupt routine DQUEUE is invoked when the transmitter is ready for another character. First it determines if any characters are available for transmission, indicated by QHEAD and QTAIL being not equal. If more data is available, it is written to the transmit buffer (SBUF) and the pointers are updated. If not, DQUEUE disables serial port interrupts and returns to the background program. ENTERQ will re-enable such interrupts as more data is available. (This example does not consider interrupt-driven serial input.)

```
            ORG     0023H
            PUSH    ACC                    ;SAVE CPU STATUS
            PUSH    PSW
            MOV     PSW,#30Q               ;SELECT BANK 3
DQUEUE:     MOV     A,QTAIL
            CJNE    A,QHEAD,DQ_1           ;TEST IF QUEUE EMPTY
            CLR     ES                     ;IF SO, CLEAR ENABLE BIT AND RETURN
            SJMP    TI_RET
```

```
DQ_1:   CLR   TI                  ;ELSE ACKNOWLEDGE REQUEST
        INC   A                   ;COMPUTE NEXT BYTE'S ADDRESS
        CJNE  A,#TOPLIM+1,DQ_2
        MOV   A,#BOTLIM           :REVISE ACC IF POINTER OVERFLOWED
DQ_2:   MOV   R0,A                ;LOAD INDEX REGISTER
        MOV   SBUF,@R0            ;RELOAD TRANSMITTER
        MOV   QTAIL,A             ;SAVE LAST POINTER USED.
TI_RET: POP   PSW                 ;RESTORE STATUS AND RETURN
        POP   A
        RETI
```

Synchronizing Timer Overflows

8051 timer overflows automatically generate an internal interrupt request, which will vector program execution to the appropriate interrupt service routine if interrupts are enabled and no other service routines are in progress at the time. However, it is not predictable *exactly* how long it will take to reach the service routine. The service routine call takes two instruction cycles, but 1, 2, or 4 additional cycles may be needed to complete the instruction in progress. If the background program ever disables interrupts, the response latency could further increase by a few instruction cycles. (Critical sections generally involve simple instruction sequences – rarely multiplies or divides.) Interrupt response delay is generally negligible, but certain time-critical application must take the exact delay into account. For example, generating interrupts with timer 1 every millisecond (1000 instruction cycles) or so would normally call for reloading it with the value –1000 (0FC30H). But if the interrupt interval (averaged over time) must be accurate to 1 instruction cycle, the 16-bit value reload into the timer must be computed, taking into account when the timer actually overflowed.

This simply requires reading the appropriate timer, which has been incremented each cycle since the overflow occurred. A sequence like the one below can stop the timer, compute how much time should elapse before the next interrupt, and reload and restart the timer. The double-precision calculation shown here compensates for any amount of timer overrun within the maximum interval. Note that it also takes into account that the timer is stopped for seven instruction cycles in the process. All interrupts are disabled, so a higher priority request will not be able to disrupt the time-critical code section.

```
;       ...     .....
        CLR     EA              ;DISABLE ALL INTERRUPTS
        CLR     TR1             ;STOP TIMER 1
        MOV     A,#LOW(-1000+7) ;LOAD LOW-ORDER DESIRED COUNT
        ADD     A,TL1           ;CORRECT FOR TIMER OVERRUN
        MOV     TL1,A           ;RELOAD LOW-ORDER BYTE.
        MOV     A,#HIGH(-1000+7);REPEAT FOR HIGH-ORDER BYTE.
        ADDC    A,TH1
        MOV     TH1,A
        SETB    TH1             ;RESTART TIMER
;       ...     .....
```

Reading a Timer/Counter "On-the-Fly"

The preceding example simply stopped the timer before changing its contents. This is normally done when reloading a timer so that the time at which the timer is started (i.e. the "run" flag is set) can be exactly controlled. There are situations, though, when it is desired to read the current count without disrupting the timing process. The 8051 timer/counter registers can all be read or written while they are running, but a few precautions must be taken.

Suppose the subroutine RDTIME should return in <R1> <R0> a sixteen-bit value indicating the count in timer 0. The instant at which the count was sampled is not as critical as the fact that the value returned must have been valid at some point while the routine was in progress. There is a potential problem that between reading the two halves, a low-order register overflow might increment the high-order register, and the two data bytes returned would be "out of phase." The solution is to read the high-order byte first, then the low-order byte, and then confirm that the high-order byte has not changed. If it has, repeat the whole process.

```
RDTIME:  MOV   A,TH0          ;SAMPLE TIMER0 (HIGH)
         MOV   R0,TL0         ;SAMPLE TIMER0 (LOW)
         CJNE  A,TH0,RDTIME   ;REPEAT IF NECESSARY
         MOV   R1,A           ;STORE VALID READ
         RET
```

Device Specifications

6

PRELIMINARY

8031/8051/8751
SINGLE-COMPONENT 8-BIT MICROCOMPUTER

- 8031 - Control Oriented CPU With RAM and I/O
- 8051 - An 8031 With Factory Mask-Programmable ROM
- 8751 - An 8031 With User Programmable/Erasable EPROM

- 4K x 8 ROM/EPROM
- 128 x 8 RAM
- Four 8-Bit Ports, 32 I/O Lines
- Two 16-Bit Timer/Event Counters
- High-Performance Full-Duplex Serial Channel
- External Memory Expandable to 128K
- Compatible with MCS-80® /MCS-85® Peripherals
- Boolean Processor
- MCS-48® Architecture Enhanced with:
  - Non-Paged Jumps
  - Direct Addressing
  - Four 8-Register Banks
  - Stack Depth Up to 128-Bytes
  - Multiply, Divide, Subtract, Compare
- Most Instructions Execute in 1μs
- 4μs Multiply and Divide

The Intel® 8031/8051/8751 is a stand-alone, high-performance single-chip computer fabricated with Intel's highly-reliable + 5 Volt, depletion-load, N-Channel, silicon-gate HMOS technolgy and packaged in a 40-pin DIP. It provides the hardware features, architectural enhancements and new instructions that are necessary to make it a powerful and cost effective controller for applications requiring up to 64K bytes of program memory and/or up to 64K bytes of data storage.

The 8051/8751 contains a non-volatile 4K x 8 read only program memory; a volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and clock circuits. The 8031 is indentical, except that it lacks the program memory. For systems that require extra capability, the 8051 can be expanded using standard TTL compatible memories and the byte oriented MCS-80 and MCS-85 peripherals.

The 8051 microcomputer, like its 8048 predecessor, is efficient both as a controller and as an arithmetic processor. The 8051 has extensive facilities for binary and BCD arithmetic and excels in bit-handling capabilities. Efficient use of program memory results from an instruction set consisting of 44% one-byte, 41% two-byte, and 15% three-byte instructions. With a 12 MHz crystal, 58% of the instructions execute in 1 μs, 40% in 2 μs and multiply and divide require only 4 μs. Among the many instructions added to the standard 8048 instruction set are multiply, divide, subtract and compare.

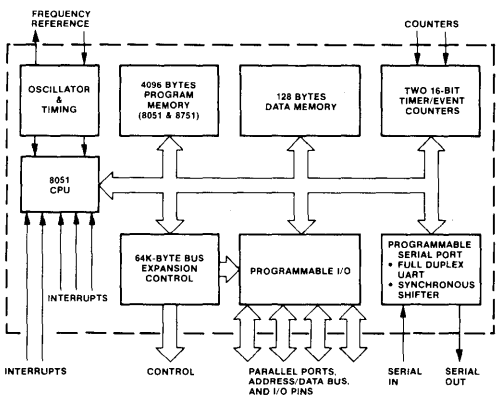


Figure 1.
Block Diagram

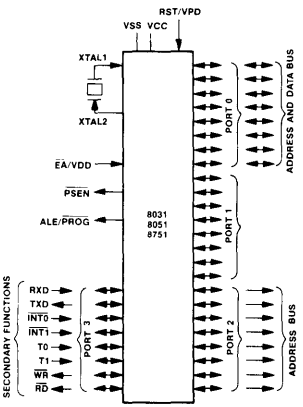


Figure 2.
Logic Symbol

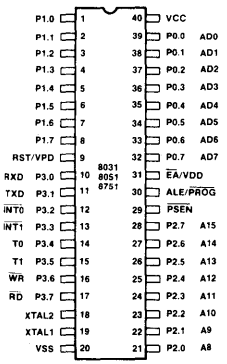


Figure 3. Pin
Configuration

8051 FAMILY PIN DESCRIPTION

V_{SS}
Circuit ground potential.

V_{CC}
+5V power supply during operation, programming and verification.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus when using external memory. It is used for data input and output during programming and verification. Port 0 can sink/source two TTL loads.

Port 1
Port 1 is an 8-bit quasi-bidirectional I/O port. It is used for the low-order address byte during programming and verification. Port 1 can sink/source one TTL load.

Port 2
Port 2 is an 8-bit quasi-bidirectional I/O port. It also emits the high-order 8 bits of address when accessing external memory. It is used for the high-order address and the control signals during programming and verification. Port 2 can sink/source one TTL load.

Port 3
Port 3 is an 8-bit quasi-bidirectional I/O port. It also contains the interrupt, timer, serial port and $\overline{RD}$ and $\overline{WR}$ pins that are used by various options. The output latch corresponding to a special function must be programmed to a one (1) for that function to operate. Port 3 can sink/source one TTL load. The special functions are assigned to the pins of Port 3, as follows:

- —RXD/data (P3.0). Serial port's receiver data input (asynchronous) or data input/output (synchronous).
- — TXD/clock (P3.1). Serial port's transmitter data output (asynchronous) or clock output (synchronous).
- — $\overline{INT0}$ (P3.2). Interrupt 0 input or gate control input for counter 0.
- — $\overline{INT1}$ (P3.3). Interrupt 1 input or gate control input for counter 1.
- — T0 (P3.4). Input to counter 0.
- — T1 (P3.5). Input to counter 1.
- — $\overline{WR}$ (P3.6). The write control signal latches the data byte from Port 0 into the External Data Memory.
- — $\overline{RD}$ (P3.7). The read control signal enables External Data Memory to Port 0.

RST/V_{PD}
A low to high transition on this pin (at approximately 3V) resets the 8051. If V_{PD} is held within its spec (approximately +5V), while V_{CC} drops below spec, V_{PD} will provide standby power to the RAM. When V_{PD} is low, the RAM's current is drawn from V_{CC}.

ALE/$\overline{PROG}$
Provides Address Latch Enable output used for latching the address into external memory during normal operation. Receives the program pulse input during EPROM programming.

$\overline{PSEN}$
The Program Store Enable output is a control signal that enables the external Program Memory to the bus during normal fetch operations.

$\overline{EA}/V_{DD}$
When held at a TTL high level, the 8051 executes instructions from the internal ROM/EPROM when the PC is less than 4096. When held at a TTL low level, the 8051 fetches all instuctions from external Program Memory. The pin also receives the 21V EPROM programming supply voltage.

XTAL1
Input to the oscillator's high gain amplifier. A crystal or external source can be used.

XTAL2
Output from the oscillator's amplifier. Required when a crystal is used.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0° C to 70° C
Storage Temperature ..... -65° C to +150° C
Voltage on Any Pin With
Respect to Ground (V_{SS}) ..... -0.5V to +7V
Power Dissipation ..... 2 Watts

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS T_A = 0° C TO 70° C; V_{CC} = 5V ± 5%; V_{SS} = 0V

Symbol	Parameter	Min.	Typ.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage (All except XTAL1)	-0.5		0.8	V	
V_{IL1}	Input Low Voltage (XTAL1)	-0.5		TBD	V	
V_{IH}	Input High Voltage (All Except XTAL1, RST/V_{PD})	2.0		V_{CC}+0.5	V	
V_{IH1}	Input High Voltage (XTAL1)	TBD		V_{CC}+0.5	V	
V_{IH2}	Input High Voltage (RST)	3.0		V_{CC} + 0.5	V	
V_{IH3}	Input High Voltage (V_{PD})	4.5		5.5	V	Power Down Only (V_{CC} = 0)
V_{OL}	Output Low Voltage (All Outputs Except Port 0)			0.45	V	1.6 mA
V_{OL1}	Output Low Voltage (Port 0)			0.45	V	3.2 mA
V_{OH}	Output High Voltage (All Outputs Except Port 0, ALE and $\overline{PSEN}$)	2.4			V	I_{OH} = -100 μA
V_{OH1}	Output High Voltage (ALE and $\overline{PSEN}$, Port 0 In External Bus Mode)	2.4			V	I_{OH} = -400 μA
I_{LO}	Pullup Resistor Current (P1, P2, P3)			-500	μA	.45V ≤ V_{IN} ≤ V_{CC}
I_{LO1}	Output Leakage Current (P0)			±10	μA	.45V ≤ V_{IN} ≤ V_{CC}
I_{CC}	Power Supply Current (All Outputs Disconnected)			150	mA	T_A = 25° C
I_{PD}	Power Down Supply Current			20	mA	T_A = 25° C, V_{PD} = 5V, V_{CC} = 0V
C_{IO}	Capacitance Of I/O Buffer			10	pF	fc = 1MHz

A.C. CHARACTERISTICS

T_A = 0°C TO 70°C; V_{CC}=5V±5% Port 0, ALE and $\overline{PSEN}$ Outputs - C_L = 150 PF; All Other Outputs - C_L = 80PF

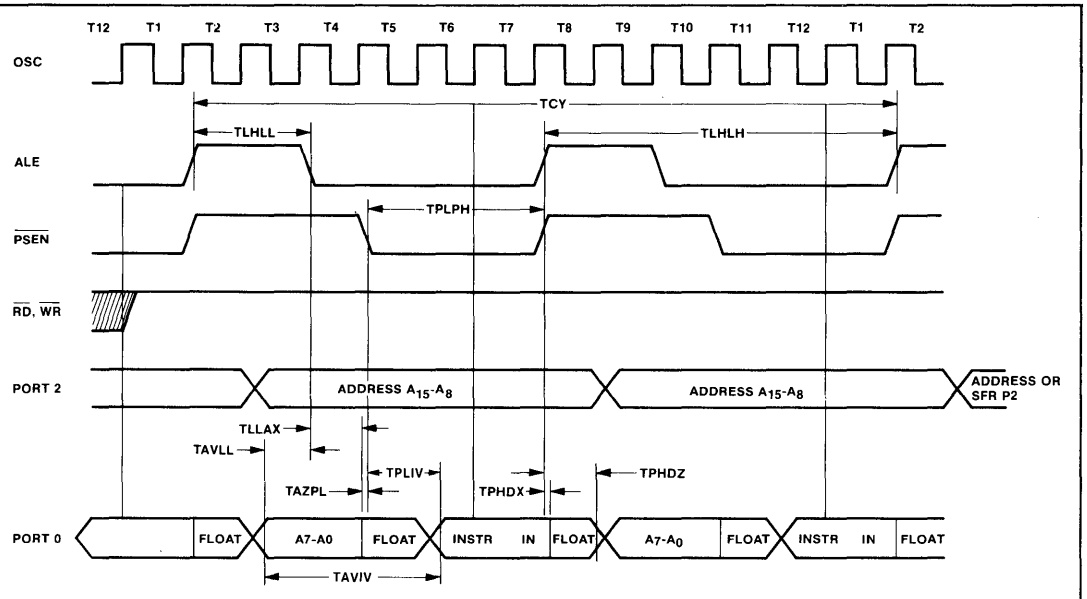
Program Memory Characteristics

		12MHz Clock			Variable Clock 1/TCLCL=1.2 MHz to 12 MHz		
Symbol	**Parameter**	**Min.**	**Max.**	**Units**	**Min.**	**Max.**	**Units**
TCLCL	Oscillator Period	83		ns			ns
TCY	Min Instruction Cycle Time	1.0		μs	12TCLCL	12TCLCL	ns
TLHLL	ALE Pulse Width	140		ns	2TCLCL-30		ns
TAVLL	Address Set Up To ALE	60		ns	TCLCL-25		ns
TLLAX	Address Hold After ALE	50		ns	TCLCL-35		ns
TPLPH	$\overline{PSEN}$ Width	230		ns	3TCLCL-20		ns
TLHLH	$\overline{PSEN}$, ALE Cycle Time	500		ns	6TCLCL		ns
TPLIV	$\overline{PSEN}$ To Valid Data In		150	ns		3TCLCL-100	ns
TPHDX	Input Data Hold After $\overline{PSEN}$	0		ns	0		ns
TPHDZ	Input Data Float After $\overline{PSEN}$		75	ns		TCLCL-10	ns
TAVIV,	Address To Valid Data In		320	ns		5TCLCL-100	ns
TAZPL	Address Float To $\overline{PSEN}$	0		ns	0		ns

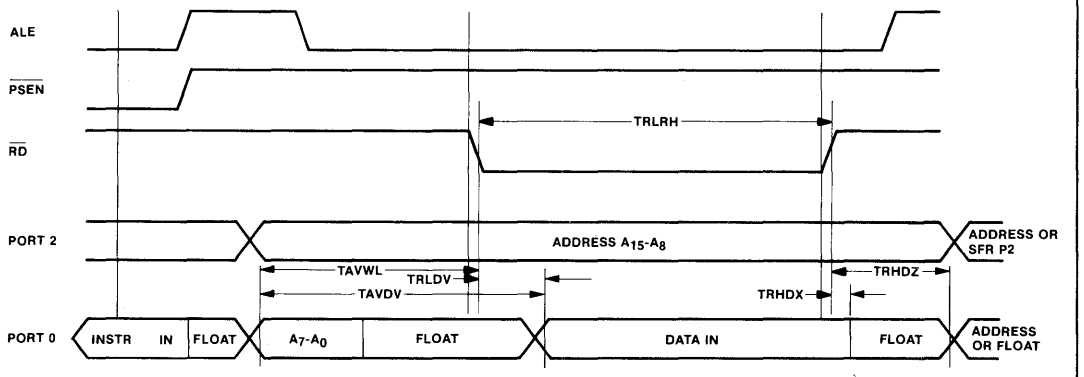
External Data Memory Characteristics

		12MHz Clock			Variable Clock		
Symbol	**Parameter**	**Min.**	**Max.**	**Units**	**Min.**	**Max.**	**Units**
TRLRH	$\overline{RD}$ Pulse Width	400		ns	6TCLCL-100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		ns	6TCLCL-100		ns
TRLDV	$\overline{RD}$ To Valid Data In		250	ns		5TCLCL-170	ns
TRHDX	Data Hold After $\overline{RD}$	0		ns	0		ns
TRHDZ	Data Float After $\overline{RD}$		100	ns		2TCLCL-70	ns
TAVDV	Address To Valid Data In		600	ns		9TCLCL-150	ns
TAVWL	Address To $\overline{WR}$ or $\overline{RD}$	200		ns	4TCLCL-130		ns
TQVWH	Data Setup Before $\overline{WR}$	400		ns	7TCLCL-180		ns
TWHQX	Data Held After $\overline{WR}$	80		ns	2TCLCL-90		ns

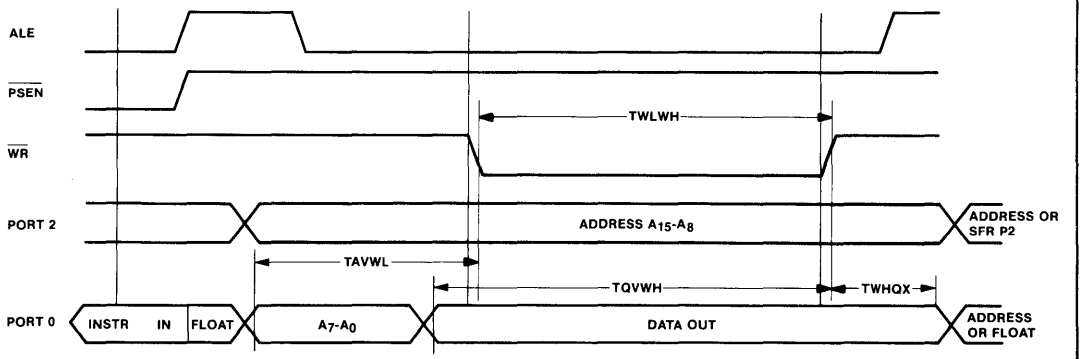
NOTE:
There are 2 to 8 ALE cycles per instruction. Clocks and state timing are shown on the timing diagram for reference purposes only. They are not accessible outside the package. TCY is the minimum instruction cycle time which consists of 12 oscillator clocks or two ALE cycles. Address setup and hold time from ALE are the same for data and program memory.



Program Memory Read Cycle



Data Memory Read Cycle

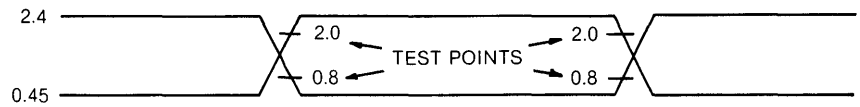


Data Memory Write Cycle

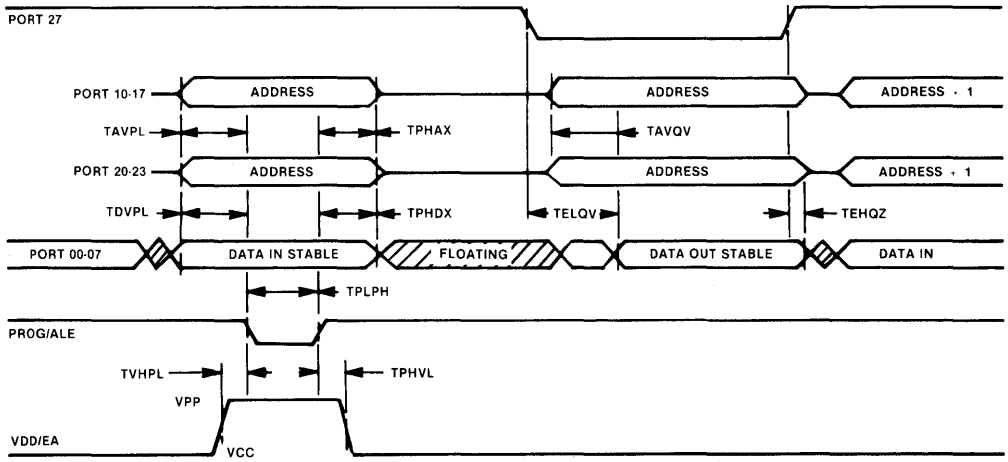
Serial Port Characteristics

Symbol	Parameter	12 MHz Clock		Variable Clock	
		Min.	Max.	Min.	Max.
TDVPL	Data Setup to PROG	10μs		3 TCY + 10μs	
TPHDX	Data Hold from PROG	10μs		3 TCY + 10μs	
TAVQV	Address to Data Valid		10μs		3 TCY + 10μs
TELQV	Output Enable (P27) to Data Valid		10μs		3 TCY + 10μs
TEHQZ	Output Enable Off to Data Float	0	10μs	0	3 TCY + 10μs
TVHPL	VDD Setup to PROG	10μs		10μs	
TPHVL	VDD Hold after PROG	10μs		10μs	
TPLPH	PROG Width	49ms	51ms	49ms	51ms

Input and Output Waveforms for A.C. Tests



PROGRAM VERIFY



NOTES: 1) PSEN = VIL, VPD/RST = VIH1
2) MSB of address is Port 23, LSB is Port 0
3) All levels are VIL, VIH, VOL, VOH except VDD/EA.

Component Data Sheets

7

8021

SINGLE COMPONENT 8-BIT MICROCOMPUTER

- 8-Bit CPU, ROM, RAM, I/O in Single 28-Pin Package
- Single 5V Supply (+4.5V to 6.5V)
- 8.38 μsec Cycle With 3.58 MHz XTAL; All Instructions 1 or 2 Cycles
- Instructions—8048 Subset
- High Current Drive Capability—2 Pins
- 1K x 8 ROM
  64 x 8 RAM
  21 I/O Lines
- Interval Timer/Event Counter
- Clock Generated With Single Inductor or Crystal
- Zero-Cross Detection Capability
- Easily Expandable I/O

The Intel® 8021 is a totally self-sufficient 8-bit parallel computer fabricated on a single silicon chip using Intel's N-channel silicon gate MOS process. The features of the 8021 include a subset of the 8048 optimized for low cost, high volume applications, plus additional I/O flexibility and power.

The 8021 contains 1K X 8 program memory, a 64 X 8 data memory, 21 I/O lines, and an 8-bit timer/event counter, in addition to on-board oscillator and clock circuits. For systems that require extra I/O capability, the 8021 can be expanded using the 8243 or discrete logic.

This microprocessor is designed to be an efficient controller as well as an arithmetic processor. The 8021 has bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting mostly of single byte instructions and no instructions over two bytes in length.

To minimize the development problems and maximize flexibility, an 8021 system can be easily designed using the 8021 emulation board, the EM-1. The EM-1 contains a 40-pin socket which can accommodate either the 8748 shipped with the board or an ICE-49 plug. Also, the necessary discrete logic to reproduce the 8021's additional I/O features is included.

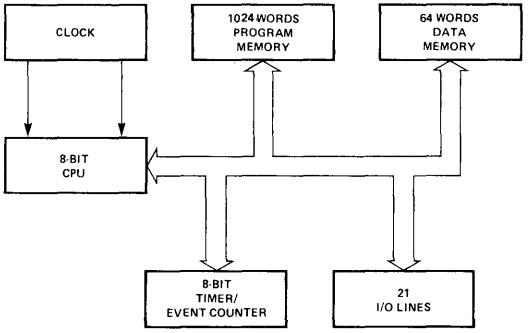


Figure 1.
Block Diagram

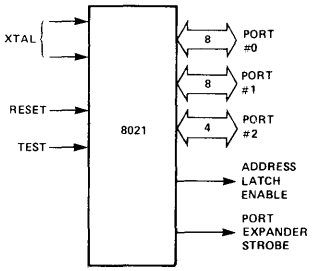


Figure 2.
Logic Symbol

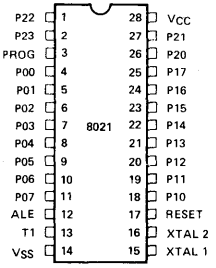


Figure 3. Pin
Configuration

AFN-01567A-01

ABSOLUTE MAXIMUM RATINGS*

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 W

**NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage	-0.5		0.8	V	
V_{IH}	Input High Voltage (All except XTAL 1 & 2, T1 RESET)	3.0		V_{CC}	V	
V_{IH1}	Input High Voltage (XTAL 1 & 2, T1 RESET)	3.8		V_{CC}	V	
$V_{IH(10\%)}$	Input high voltage (All except XTAL 1 & 2, T1, RESET)	2.0		V_{CC}	V	V_{CC} = 5.0V ± 10%
$V_{IH1(10\%)}$	Input high voltage (XTAL 1 & 2, T1, RESET)	3.5		V_{CC}	V	V_{CC} = 5.0V ± 10%
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (P10, P11)			2.5	V	I_{OL} = 7 mA
V_{OH}	Output High Voltage (All unless Open Drain)	2.4			V	I_{OH} = 40 μA
I_{LO}	Output Leakage Current (Open Drain Option—Port 0)			± 10	μA	V_{SS}+0.45≤V_{IN}≤V_{CC}
I_{CC}	V_{CC} Supply Current		40	75	mA	

T1 ZERO CROSS CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V, C_L = 80pF

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
VZX	Zero-Cross Detection Input (T1)	1	3	VPP	AC Coupled, C = .2μF
AZX	Zero-Cross Accuracy		± 135	mV	60 Hz Sine Wave
FZX	Zero-Cross Detection Input Frequency (T1)	0.05	1	kHZ	

Table 1. Pin Description

Desig-nation	Pin No.	Function
VSS	14	Circuit GND potential
VCC	28	+5V power supply
PROG	3	Output strobe for 8243 I/O Expander
P00-P07 Port 0	4-11	8-bit quasi-bidirectional port
P10-P17 Port 1	18-25	8-bit quasi-bidirectional port
P20-P23 Port 2	26-27 1-2	4-bit quasi-bidirectional port P20-P23 also serve as a 4-bit I/O expander bus for 8243
T1	13	Input pin testable using the JT1 and JNT1 instructions. Can be designated the timer/event counter input using the STRT CNT instruction. Also allows zero-crossover sensing of slowly moving inputs.
RESET	17	Input used to initialize the processor by clearing status flip-flops and setting program counters to zero.
ALE	12	Address Latch Enable. Signal occurring once every 30 input clocks, used as an output clock.
XTAL1	15	One side of crystal or inductor input for internal oscillator. Also input for external source. (Not TTL compatible.)
XTAL2	16	Other side of timing control element.

Table 2. Instruction Set Summary

	Mnemonic	Description	Bytes	Cycle	Hexadecimal Opcode
Accumulator	ADD A,R_r	Add register to A	1	1	68-6F
	ADD A,@ R	Add data memory to A	1	1	60-61
	ADD A,#data	Add immediate to A	2	2	03
	ADDC A,R_r	Add register with carry	1	1	78-7F
	ADDC A,@ R	Add data memory with carry	1	1	70-71
	ADDC A,#data	Add immediate with carry	2	2	13
	ANL A,R_r	And register to A	1	1	58-5F
	ANL A,@ R	And data memory to A	1	1	50-51
	ANL A,#data	And immediate to A	2	2	53
	ORL A,R_r	Or register to A	1	1	48-4F
	ORL A,@ R	Or data memory to A	1	1	40-41
	ORL A,#data	Or immediate to A	2	2	43
	XRL A,R_r	Exclusive Or register to A	1	1	D8-DF
	XRL A,@ R	Exclusive Or data memory to A	1	1	D0-D1
	XRL A,#data	Exclusive Or immediate to A	2	2	D3
	INC A	Increment A	1	1	17
	DEC A	Decrement A	1	1	07
	CLR A	Clear A	1	1	27
	CPL A	Complement A	1	1	37
	DA A	Decimal adjust A	1	1	57
	SWAP A	Swap nibbles of A	1	1	47
	RL A	Rotate A left	1	1	E7
	RLC A	Rotate A left through carry	1	1	F7
	RR A	Rotate A right	1	1	77
	RRC A	Rotate A right through carry	1	1	67
Input/Output	IN A, P_p	Input port to A	1	2	08,09,0A
	OUTL P_p,A	Output A to port	1	2	90,39,3A
	MOVD A,P_p	Input expander port to A	1	2	0C-0F
	MOVD P_p,A	Output A to expander port	1	2	3C-3F
	ANLD P_p,A	And A to expander port	1	2	9C-9F
	ORLD P_p,A	Or A to expander port	1	2	8C-8F
Registers	INC R_r	Increment register	1	1	18-1F
	INC @ R	Increment data memory	1	1	10-11
Branch	JMP addr	Jump unconditional	2	2	04,24,44,64,
	JMPP @ A	Jump indirect	1	2	B3
	DJNZ R_r, addr	Decrement register and jump on R not zero	2	2	E8-EF
	JC addr	Jump on carry=1	2	2	F6
	JNC addr	Jump on carry=0	2	2	E6
	JZ addr	Jump on A zero	2	2	C6
	JNZ addr	Jump on A not zero	2	2	96
	JT1 addr	Jump on T1=1	2	2	56
	JNT1 addr	Jump on T1=0	2	2	46
	JTF addr	Jump on timer flag	2	2	16
Subroutine	CALL addr	Jump to subroutine	1	2	14,34,54,74
	RET	Return	1	2	83
Flags	CLR C	Clear carry	1	1	97
	CPL C	Complement carry	1	1	A7
Data Moves	MOV A,R_r	Move register to A	1	1	F8-FF
	MOV A,@ R	Move data memory to A	1	1	F0-F1
	MOV A,#data	Move immediate to A	2	2	23
	MOV R_r,A	Move A to register	1	1	A8-AF
	MOV @ R,A	Move A to data memory	1	1	A0-A1
	MOV R_r,#data	Move immediate to register	2	2	B8-BF
	MOV@R,#data	Move immediate to data memory	2	2	B0-B1
	XCH A,R_r	Exchange A and register	1	1	28-2F
	XCH A,@ R	Exchange A and data memory	1	1	20-21
	XCHD A,@ R	Exchange nibble of A and register	1	1	30-31
	MOVP A,@ A	Move to A from current page	1	2	A3
Timer/Counter	MOV A,T	Read timer/counter	1	1	42
	MOV T,A	Load timer/counter	1	1	62
	STRT T	Start timer	1	1	55
	STRT CNT	Start counter	1	1	45
	STOP TCNT	Stop timer/counter	1	1	65
	NOP	No operation	1	1	00

8021L
SINGLE COMPONENT 8-BIT MICROCOMPUTER
LOW POWER 10mA

- **8-Bit CPU, ROM, RAM, I/O in Single 28-Pin Package**
- **Single 5V Supply (+4.5V to 8V)**
- **8.38 μsec Cycle With 3.58 MHz XTAL; All instructions 1 or 2 Cycles**
- **Instructions—8048 Subset**
- **High Current Drive Capability—2 Pins**
- **1K x 8 ROM**
  64 x 8 RAM
  21 I/O Lines
- **Interval Timer/Event Counter**
- **Clock Generated With Single Inductor or Crystal**
- **Zero-Cross Detection Capability**
- **Easily Expandable I/O**

The Intel® 8021L is a totally self-sufficient 8-bit parallel computer fabricated on a single silicon chip using Intel's N-channel silicon gate MOS process. The features of the 8021L include a subset of the 8048 optimized for low cost, high volume applications, plus additional I/O flexibility and power.

The 8021L contains 1K X 8 program memory, a 64 X 8 data memory, 21 I/O lines, and an 8-bit timer/event counter, in addition to on-board oscillator and clock circuits. For systems that require extra I/O capability, the 8021L can be expanded using the 8243 or discrete logic.

This microprocessor is designed to be an efficient controller as well as an arithmetic processor. The 8021L has bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting mostly of single byte instructions and no instructions over two bytes in length.

To minimize the development problems and maximize flexibility, an 8021L system can be easily designed using the 8021L emulation board, the EM-1. The EM-1 contains a 40-pin socket which can accommodate either the 8748 shipped with the board or an ICE-49 plug. Also, the necessary discrete logic to reproduce the 8021L's additional I/O features is included.

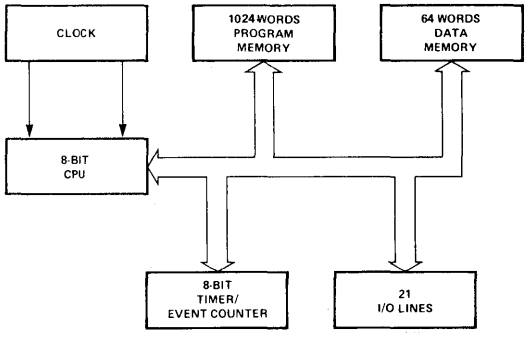


Figure 1. Block Diagram

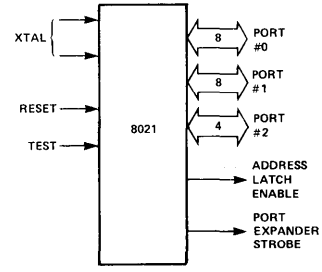


Figure 2. Logic Symbol

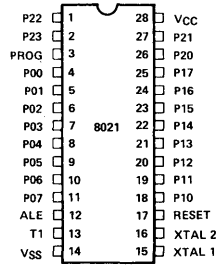


Figure 3. Pin Configuration

ABSOLUTE MAXIMUM RATINGS*

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 W

**NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage	-0.5		0.8	V	
V_{IH}	Input High Voltage (All except XTAL 1 & 2, T1 RESET)	3.0		VCC	V	
V_{IH1}	Input High Voltage (XTAL 1 & 2, T1 RESET)	3.8		VCC	V	
$V_{IH(10\%)}$	Input high voltage (All except 1 & 2, T1, RESET)	2.0		VCC	V	VCC = 5.0V ± 10%
$V_{IH1(10\%)}$	Input high voltage (XTAL 1 & 2, T1, RESET)	3.5		VCC	V	VCC = 5.0V ± 10%
V_{OL}	Output Low Voltage			0.45	V	I_{OL} + 1.6 mA
V_{OL1}	Output Low Voltage (P10, P11)			2.5	V	I_{OL} = 7 mA
V_{OH}	Output High Voltage (All unless Open Drain)	2.4			V	I_{OH} = 40 μA
I_{LO}	Output Leakage Current (Open Drain Option—Port 0)			± 10	μA	V_{SS}+0.45≤V_{IN}≤V_{CC}
I_{CC}	VCC Supply Current		40	75	mA	

T1 ZERO CROSS CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V, C_L = 80 pF

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
VZX	Zero-Cross Detection Input (T1)	1	3	VPP	AC Coupled, C = .2μF
AZX	Zero-Cross Accuracy		± 135	mV	60 Hz Sine Wave
FZX	Zero-Cross Detection Input Frequency (T1)	0.05	1	kHZ	
tCY	Cycle Time	8.38	50.0		3.58 MHz XTAL = 8.38 μs tCY

8022

SINGLE COMPONENT 8-BIT MICROCOMPUTER WITH ON-CHIP A/D CONVERTER

- 8-Bit CPU, ROM, RAM, I/O in Single 40-Pin Package
- On-Chip 8-Bit A/D Converter; Two Input Channels
- 8 Comparator Inputs (Port 0)
- Zero-Cross Detection Capability
- Single 5V Supply (4.5V to 6.5V)
- High Current Drive Capability—2 Pins
- Two Interrupts—External and Timer
- 2K x 8 ROM, 64 x 8 RAM, 28 I/O Lines
- 8.38 μsec Cycle; All Instructions 1 or 2 Cycles
- Instructions—8048 Subset
- Interval Timer/Event Counter
- Clock Generated with Single Inductor or Crystal
- Easily Expanded I/O

The Intel® 8022 is the newest member of the MCS-48™ family of single chip 8-bit microcomputers. It is designed to satisfy the requirements of low cost, high volume applications which involve analog signals, capacitive touchpanel keyboards, and/or large ROM space. The 8022 addresses these applications by integrating many new functions on-chip, such as A/D conversion, comparator inputs and zero-cross detection.

The features of the 8022 include 2K bytes of program memory (ROM), 64 bytes of data memory (RAM), 28 I/O lines, an on-chip A/D converter with two input channels, an 8-bit port with comparator inputs for interfacing to low voltage capacitive touchpanels or other non-TTL interfaces, external and timer interrupts, and zero-cross detection capability. In addition, it contains the 8-bit interval timer/event counter, on-board oscillator and clock circuitry, single 5V power supply requirement, and easily expandable I/O structure common to all members of the MCS-48 family.

The 8022 is designed to be an efficient controller as well as an arithmetic processor. It has bit handling capability plus facilities for both binary and BCD arithmetic. Efficient use of program memory results from using the MCS-48 instruction set which consists mostly of single byte instructions and has extensive conditional jump and direct table lookup capability. Program memory usage is further reduced via the 8022's hardware implementation of the A/D

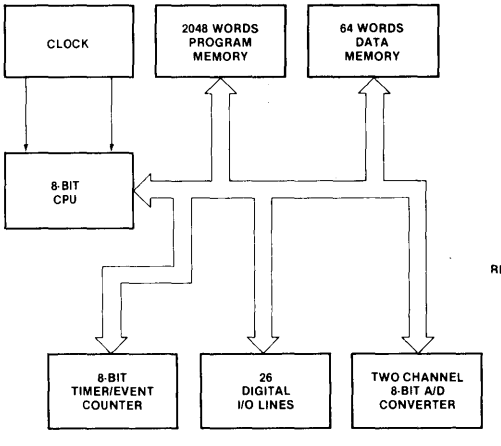


Figure 1. Block Diagram

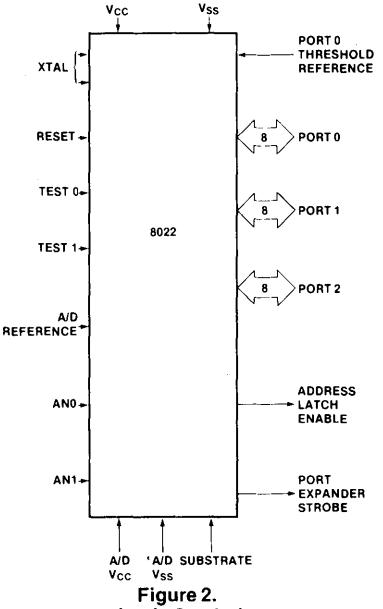


Figure 2. Logic Symbol

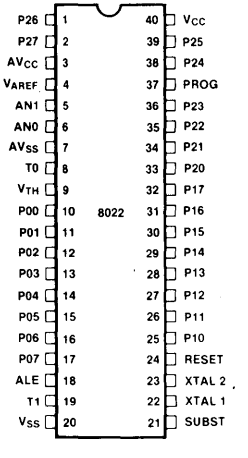


Figure 3. Pin Configuration

Table 1. Pin Description

Designation	Pin No.	Function
VSS	20	Circuit GND potential.
VCC	40	+ 5V circuit power supply.
PROG	37	Output strobe for Intel® 8243 I/O expander.
P00-P07 Port 0	10-17	8-bit open-drain port with comparator inputs. The switching threshold is set externally by VTH. Optional pull-up resistors may be added via ROM mask selection.
VTH	9	Port 0 threshold reference pin.
P10-P17 Port 1	25-32	8-bit quasi-bidirectional port.
P20-P27 Port 2	33-36 38-39 1-2	8-bit quasi-bidirectional port. P20-23 also serve as a 4-bit I/O expander for Intel® 8243.
T0	8	Interrupt input and input pin testable using the conditional transfer instructions JT0 and JNT0. Initiates an interrupt following a low level input if interrupt is enabled. Interrupt is disabled after a reset.
T1	19	Input pin testable using the JT1 and JNT1 conditional transfer instructions. Can be designated the timer/event counter input using the STRT CNT instruction. Also serves as the zero-cross detection input to allow zero-crossover sensing of slowly moving AC inputs. Optional pull-up resistor may be added via ROM mask selection.

Designation	Pin No.	Function
RESET	24	Input used to initialize the processor by clearing status flip-flops and setting the program counter to zero.
AVSS	7	A/D converter GND Potential. Also establishes the lower limit of the conversion range.
AVCC	3	A/D + 5V power supply.
SUBST	21	Substrate pin used with a bypass capacitor to stabilize the substrate voltage and improve A/D accuracy.
VAREF	4	A/D converter reference voltage. Establishes the upper limit of the conversion range.
AN0, AN1	6,5	Analog inputs to A/D converter. Software selectable on-chip via SEL AN0 and SEL AN1 instructions.
ALE	18	Address Latch Enable. Signal occurring once every 30 input clocks (once every cycle), used as an output clock.
XTAL 1	22	One side of crystal or inductor input for internal oscillator. Also input for external frequency source. (Not TTL compatible.)
XTAL 2	23	Other side of timing control element. This pin is not connected when an external frequency source is used.

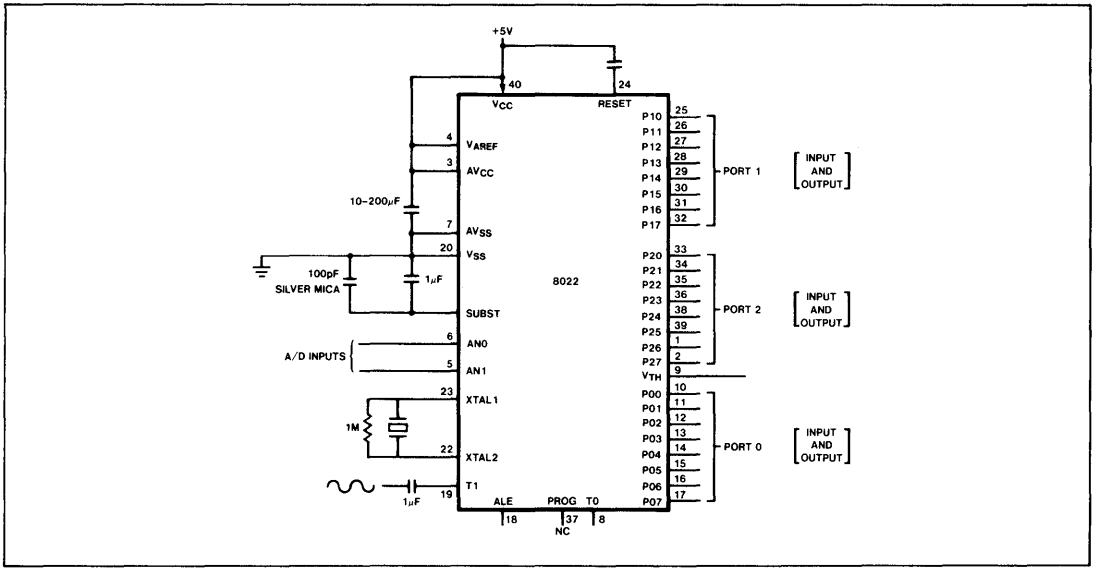


Figure 3.The Stand Alone 8022

ABSOLUTE MAXIMUM RATINGS*

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 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS T_A = 0° C to 70° C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage	-0.5		0.8	V	VTH Floating
V_{IH1}	Input Low Voltage (Port 0)	-0.5		VTH-0.1	V	
V_{IH}	High Voltage (All except XTAL 1, RESET)	2.0		VCC	V	VCC = 5.0V ± 10% VTH Floating
V_{IH1}	Input High Voltage (All except XTAL 1, RESET)	3.0		VCC	V	VCC = 5.5V ± 1V VTH Floating
V_{IH2}	Input High Voltage (Port 0)	VTH+0.1		VCC	V	
V_{IH3}	Input High Voltage (RESET, XTAL 1)	3.0		VCC	V	VCC = 5.0V ± 10%
VTH	Port 0 Threshold Reference Voltage	0		.4VCC	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (P10, P11)			2.5	V	I_{OL} = 7 mA
V_{OH}	Output High Voltage (all unless Open Drain Option—Port 0)	2.4			V	I_{OH} = 50 μA
I_{LI}	Input Current (T1)			± 200	μA	VCC ≥ VIN ≥ VSS+.45V
I_{LO}	Output Leakage Current (Open Drain Option—Port 0)			± 10	μA	VCC ≥ VIN ≥ VSS+0.45V
I_{CC}	VCC Supply Current		50	100	mA	

A.C. CHARACTERISTICS T_A = 0° C to 70° C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V

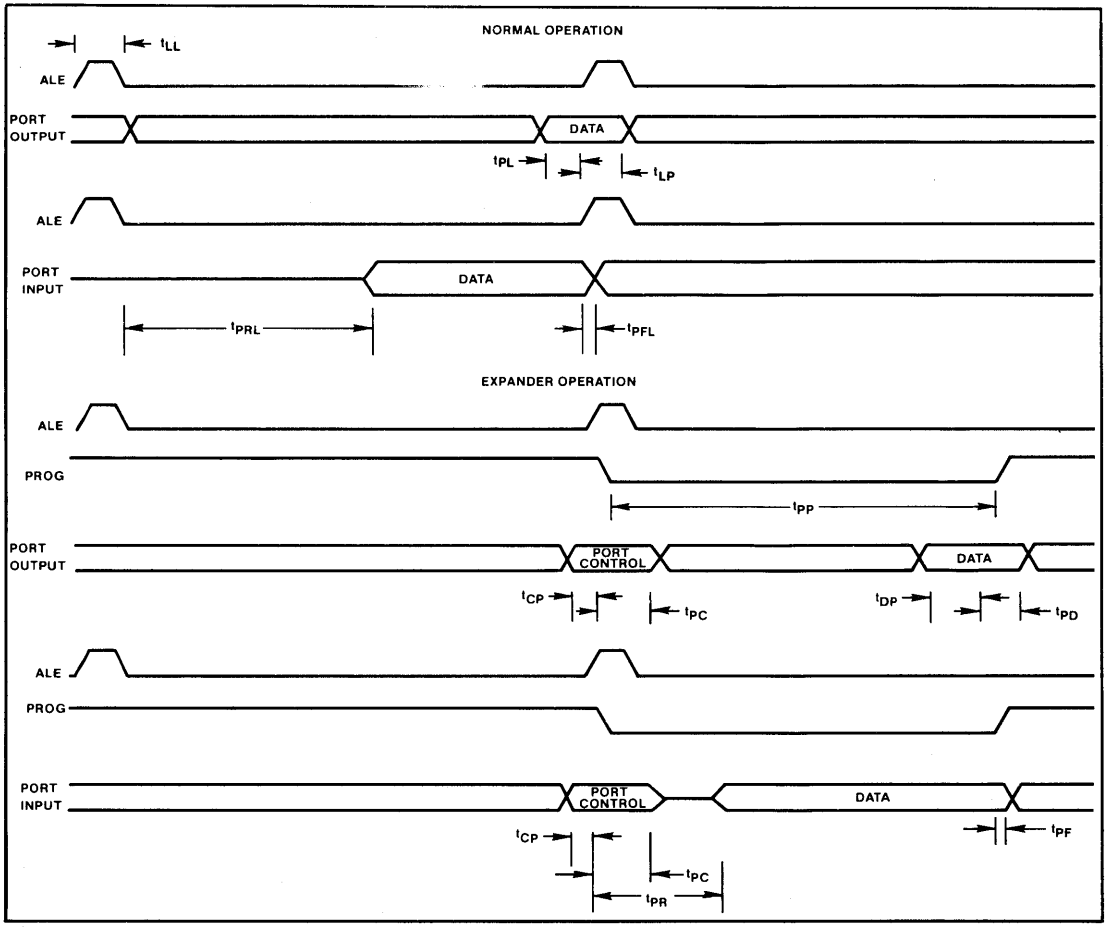
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
tCY	Cycle Time	8.38	50.0	μS	3 MHz XTAL = 10 μs tCY
VZX	Zero-Cross Detection Input (T1)	1	3	VACpp	AC Coupled
AZX	Zero-Cross Accuracy		± 135	mV	60 Hz Sine Wave
FZX	Zero-Cross Detection Input Frequency (T1)	0.05	1	kHz	

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5.5V \pm 1V$, $V_{SS} = 0V$

Test Conditions: $C_L = 80$ pF $t_{CY} = 8.38\ \mu s$

	Symbol	Parameter	Min.	Max.	Unit	Notes
Expander Operation	t_{CP}	Port Control Setup Before Falling Edge of PROG	0.5		μs	
	t_{PC}	Port Control Hold After Falling Edge of PROG	0.8		μs	
	t_{PR}	PROG to Time P2 Input Must Be Valid		1.0	μs	
	t_{DP}	Output Data Setup Time	7.0		μs	
	t_{PD}	Output Data Hold Time	8.3		μs	
	t_{PF}	Input Data Hold Time	0	.150	μs	
	t_{PP}	PROG Pulse Width	8.3		μs	
Normal Operation	t_{PRL}	ALE to Time P2 Input Must Be Valid		3.6	μs	
	t_{PL}	Output Data Setup Time	0.8		μs	
	t_{LP}	Output Data Hold Time	1.6		μs	
	t_{PFL}	Input Data Hold Time	0		μs	
	t_{LL}	ALE Pulse Width	3.9	23.0	μs	$t_{CY} = 8.38\ \mu s$ for min

Port 2 Timing

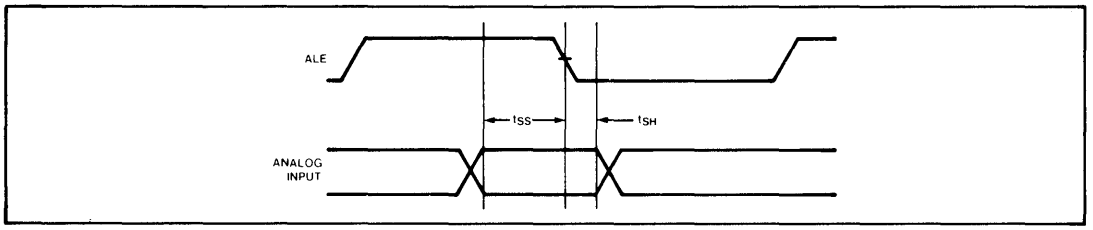


A/D CONVERTER CHARACTERISTICS

T_A = 0° C to 70° C, V_{CC} = 5.5V ± 1V, V_{SS} = 0V, AV_{CC} = 5.5V ± 1V, AV_{SS} = 0V, $AV_{CC}/2 \leqslant V_{AREF} \leqslant AV_{CC}$

Parameter	Min.	Typ.	Max.	Unit	Comments
Resolution	8			Bits	
Absolute Accuracy			.8% FSR ± ½ LSB	LSB	(Note 1)
Sample Setup Before Falling Edge of ALE (t_{SS})		0.20		t_{CY}	
Sample Hold After Falling Edge of ALE (t_{SH})		0.10		t_{CY}	
Input Capacitance (AN0, AN1)		1		pF	
Conversion Time	4		4	t_{CY}	

Analog Input Timing



NOTE:

1. The analog input must be maintained at a constant voltage during the sample time ($t_{SS} + t_{SH}$).

Table 2. Instruction Set Summary

	Mnemonic	Description	Bytes	Cycle	Hexadecimal Opcode
Accumulator	ADD A,R_r	Add register to A	1	1	68-6F
	ADD A,@ R	Add data memory to A	1	1	60-61
	ADD A,#data	Add immediate to A	2	2	03
	ADDC A,R_r	Add register with carry	1	1	78-7F
	ADDC A,@ R	Add data memory with carry	1	1	70-71
	ADDC A,#data	Add immediate with carry	2	2	13
	ANL A,R_r	And register to A	1	1	58-5F
	ANL A,@ R	And data memory to A	1	1	50-51
	ANL A,#data	And immediate to A	2	2	53
	ORL A,R_r	Or register to A	1	1	48-4F
	ORL A,@ R	Or data memory to A	1	1	40-41
	ORL A,#data	Or immediate to A	2	2	43
	XRL A,R_r	Exclusive Or register to A	1	1	D8-DF
	XRL A,@ R	Exclusive Or data memory to A	1	1	D0-D1
	XRL A,#data	Exclusive Or immediate to A	2	2	D3
	INC A	Increment A	1	1	17
	DEC A	Decrement A	1	1	07
	CLR A	Clear A	1	1	27
	CPL A	Complement A	1	1	37
	DA A	Decimal adjust A	1	1	57
	SWAP A	Swap nibbles of A	1	1	47
	RL A	Rotate A left	1	1	E7
	RLC A	Rotate A left through carry	1	1	F7
	RR A	Rotate A right	1	1	77
	RRC A	Rotate A right through carry	1	1	67
Input/Output	IN A, P_p	Input port to A	1	2	08,09,0A
	OUTL P_p,A	Output A to port	1	2	90,39,3A
	MOVD A,P_p	Input expander port to A	1	2	0C-0F
	MOVD P_p,A	Output A to expander port	1	2	3C-3F
	ANLD P_p,A	And A to expander port	1	2	9C-9F
	ORLD P_p,A	Or A to expander port	1	2	8C-8F
Registers	INC R_r	Increment register	1	1	18-1F
	INC @ R	Increment data memory	1	1	10-11
Branch	JMP addr	Jump unconditional	2	2	04,24,44,64, 84,A4,C4,E4
	JMPP @ A	Jump indirect	1	2	B3
	DJNZ R,addr	Decrement register and jump on R not zero	2	2	E8-EF
	JC addr	Jump on carry=1	2	2	F6
	JNC addr	Jump on carry=0	2	2	E6
	JZ addr	Jump on A zero	2	2	C6
	JNZ addr	Jump on A not zero	2	2	96

	Mnemonic	Description	Bytes	Cycle	Hexadecimal Opcode
	JTO addr	Jump on T0=1	2	2	36
	JNTO addr	Jump on T0=0	2	2	26
	JT1 addr	Jump on T1=1	2	2	56
	JNT1 addr	Jump on T1=0	2	2	46
	JTF addr	Jump on timer flag	2	2	16
Subroutine	CALL addr	Jump to subroutine	1	2	14,34,54,74 94,B4,D4,F4
	RET	Return	1	2	83
Flags	CLR C	Clear carry	1	1	97
	CPL C	Complement carry	1	1	A7
Data Moves	MOV A,R_r	Move register to A	1	1	F8-FF
	MOV A,@ R	Move data memory to A	1	1	F0-F1
	MOV A,#data	Move immediate to A	2	2	23
	MOV R_r,A	Move A to register	1	1	A8-AF
	MOV @ R,A	Move A to data memory	1	1	A0-A1
	MOV R_r,#data	Move immediate to register	2	2	B8-BF
	MOV@R,#data	Move immediate to data memory	2	2	B0-B1
	XCH A,R_r	Exchange A and register	1	1	28-2F
	XCH A,@ R	Exchange A and data memory	1	1	20-21
	XCHD a,@ R	Exchange nibble of A and register	1	1	30-31
	MOVP A,@ A	Move to A from current page	1	2	A3
Timer/Counter	MOV A,T	Read timer/counter	1	1	42
	MOV T,A	Load timer/counter	1	1	62
	STRT T	Start timer	1	1	55
	STRT CNT	Start counter	1	1	45
	STOP TCNT	Stop timer/counter	1	1	65
A/D Converter	RAD	Move conversion resuli register to A	1	2	80
	SEL AN0	Select analog input zero	1	1	85
	SEL AN1	Select analog input one	1	1	95
Interrupts	EN I	Enable external interrupt	1	1	05
	DIS I	Disable external interrupt	1	1	15
	EN TCNTI	Enable timer/counter interrupt	1	1	25
	DIS TCNTI	Disable timer/counter interrupt	1	1	35
	RET I	Return from interrupt	1	2	93
	NOP	No operation	1	1	00

SYMBOLS AND ABBREVIATIONS USED

A	Accumulator
addr	11-Bit Program Memory Address
AN0, AN1	Analog Input 0, Analog Input 1
CNT	Event Counter
data	8-Bit Number or Expression
I	Interrupt
P	Mnemonic for "in-page" Operation
P_p	Port Designator (P=0, 1, 2 or 4-7)
R_r	Register Designator (r=0-7)
T	Timer
T0, T1	Test 0, Test 1
#	Immediate Data Prefix
@	Indirect Address Prefix

8022H
HIGH PERFORMANCE SINGLE COMPONENT 8-BIT MICROCOMPUTER WITH ON-CHIP A/D CONVERTER

- 8-Bit CPU, ROM, RAM, I/O in Single 40-Pin Package
- On-Chip 8-Bit A/D Converter; Two Input Channels
- 8 Comparator Inputs (Port 0)
- Zero-Cross Detection Capability
- Single 5V Supply (4.5V to 6.5V)
- Two Interrupts—External and Timer
- 2K x 8 ROM, 64 x 8 RAM, 28 I/O Lines
- 5 μsec Cycle; All Instructions 1 or 2 Cycles (6 MHz Clock)
- Instructions—8048 Subset
- Interval Time/Event Counter
- Clock Generated with Single Inductor or Crystal
- Easily Expanded I/O

The Intel® 8022H is designed to satisfy the requirements of low cost, high volume applications which involve analog signals, capacitive touchpanel keyboards, and/or large ROM space. The 8022H addresses these applications by integrating many new functions on-chip, such as A/D conversion, comparator inputs and zero-cross detection.

The features of the 8022H include 2K bytes of program memory (ROM), 64 bytes of data memory (RAM), 28 I/O lines, an on-chip A/D converter with two input channels, an 8-bit port with comparator inputs for interfacing to low voltage capacitive touchpanels or other non-TTL interfaces, external timer interrupts, and zero-cross detection capability. In addition, it contains the 8-bit interval timer/event counter, on-board oscillator and clock circuitry, single 5V power supply requirement, and easily expandable I/O structure common to all members of the MCS-48 family.

The 8022H is designed to be an efficient controller as well as an arithmetic processor. It has bit handling capability plus facilities for both binary and BCD arithmetic. Efficient use of program memory results from using the MCS-48 instruction set which consists mostly of single byte instructions and has extensive conditional jump and direct table lookup capability. Program memory usage is further reduced via the 8022H's hardware implementation of the A/D converter which simplifies interfacing to analog signals.

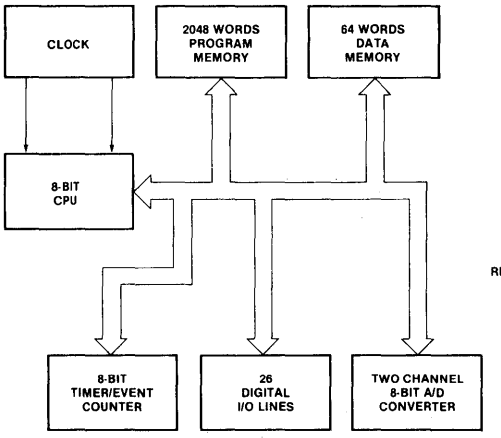


Figure 1. Block Diagram

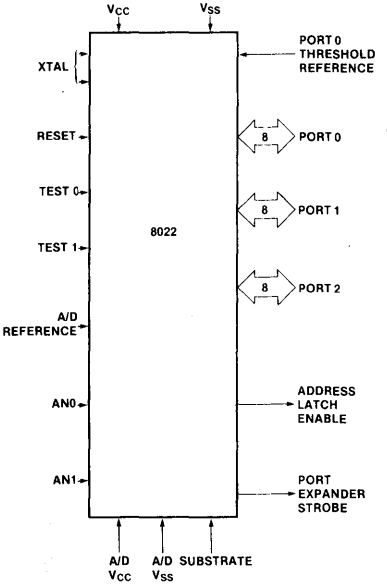


Figure 2. Logic Symbol

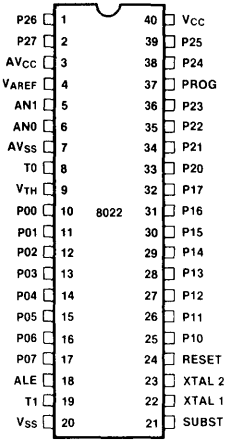


Figure 3. Pin Configuration

8048H/8048H-1/8035HL/8035HL-1
HMOS SINGLE COMPONENT 8-BIT MICROCOMPUTER

- 8048H/8048H-1 Mask Programmable ROM
- 8035HL/8035HL-1 CPU Only with Power Down Mode

- 8-BIT CPU, ROM, RAM, I/O in Single Package
- High Performance HMOS
- Reduced Power Consumption
- 1.4 µsec and 1.9 µsec Cycle Versions All Instructions 1 or 2 Cycles
- Over 90 Instructions: 70% Single Byte
- 1K x 8 ROM 64 x RAM 27 I/O Lines
- Interval Timer/Event Counter
- Easily Expandable Memory and I/O
- Compatible with 8080/8085 Series Peripherals
- Two Single Level Interrupts

The Intel® 8048H/8048H-1/8035HL/8035HL-1 are totally self-sufficient, 8-bit parallel computers fabricated on single silicon chips using Intel's advanced N-channel silicon gate HMOS process.

The 8048H contains a 1K X 8 program memory, a 64 X 8 RAM data memory, 27 I/O lines, and an 8-bit timer/counter in addition to on-board oscillator and clock circuits. For systems that require extra capability the 8048H can be expanded using standard memories and MCS-80® /MCS-85® peripherals. The 8035HL is the equivalent of the 8048H without program memory and can be used with external ROM and RAM.

To reduce development problems to a minimum and provide maximum flexibility, a logically and functionally pin compatible version of the 8048H with UV-erasable user-programmable EPROM program memory is available. The 8748 will emulate the 8048H up to 6 MHz clock frequency with minor differences.

The 8048H is fully compatible with the 8048 when operated at 6MHz.

These microcomputers are designed to be efficient controllers as well as arithmetic processors. They have extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting mostly of single byte instructions and no instructions over 2 bytes in length.

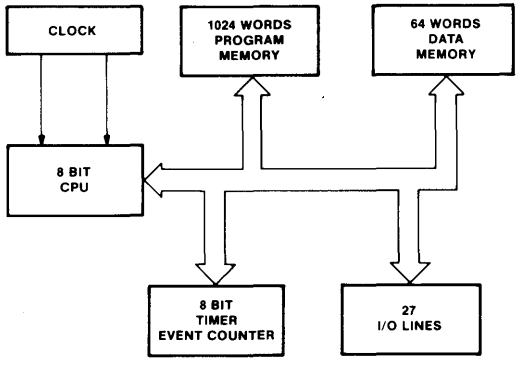


Figure 1. Block Diagram

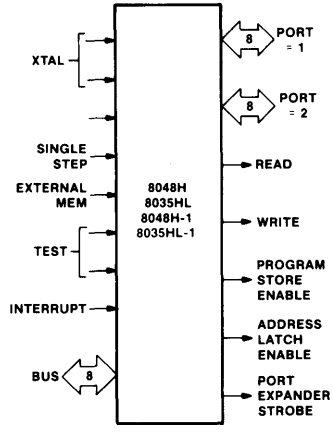


Figure 2. Logic Symbol

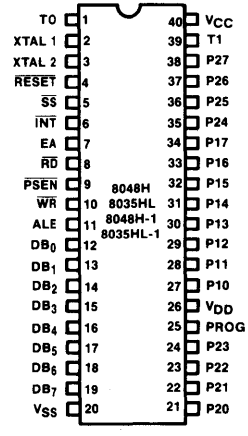


Figure 3. Pin Configuration (top view)

AFN-01491B-01

Table 1. Pin Description

Symbol	Pin No.	Function
V_{SS}	20	Circuit GND potential
V_{DD}	26	Low power standby pin
V_{CC}	40	Main power supply; +5V during operation.
PROG	25	Output strobe for 8243 I/O expander.
P10-P17 Port 1	27-34	8-bit quasi-bidirectional port.
P20-P27 Port 2	21-24	8-bit quasi-bidirectional port.
	35-38	P20-P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
DB0-DB7 BUS	12-19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched.
		Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.
T0	1	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction.
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset.
		Also testable with conditional jump instruction. (Active low)
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the bus from an external device.
		Used as a read strobe to external data memory. (Active low)
$\overline{RESET}$	4	Input which is used to initialize the processor. (Active low) (Non TTL V_{IH})
$\overline{WR}$	10	Output strobe during a bus write. (Active low)
		Used as write strobe to external data memory.
ALE	11	Address latch enable. This signal occurs once during each cycle and is useful as a clock output.
		The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	Program store enable. This output occurs only during a fetch to external program memory. (Active low)
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction. (Active low)
EA	7	External access input which forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification. (Active high)
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source. (Non TTL V_{IH})
XTAL2	3	Other side of crystal input.

AFN-01491B-02

Table 2. Instruction Set

Accumulator

Mnemonic	Description	Bytes	Cycles
ADD A, R	Add register to A	1	1
ADD A, @R	Add data memory to A	1	1
ADD A, # data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, # data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, # data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A @R	Or data memory to A	1	1
ORL A, # data	Or immediate to A	2	2
XRL A, R	Exclusive or register to A	1	1
XRL A, @R	Exclusive or data memory to A	1	1
XRL, A, # data	Exclusive or immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1

Input/Output

Mnemonic	Description	Bytes	Cycles
IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, # data	And immediate to port	2	2
ORL P, # data	Or immediate to port	2	2
INS A, BUS	Input BUS to A	1	2
OUTL BUS, A	Output A to BUS	1	2
ANL BUS, # data	And immediate to BUS	2	2
ORL BUS, # data	Or immediate to BUS	2	2
MOVD A,P	Input expander port to A	1	2
MOVD P, A	Output A to expander port	1	2
ANLD P, A	And A to expander port	1	2
ORLD P, A	Or A to expander port	1	2

Registers

Mnemonic	Description	Bytes	Cycles
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1

Branch

Mnemonic	Description	Bytes	Cycles
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and skip	2	2
JC addr	Jump on carry = 1	2	2
JNC addr	Jump on carry = 0	2	2
JZ addr	Jump on A zero	2	2
JNZ addr	Jump on A not zero	2	2
JTO addr	Jump on TO = 1	2	2
JNTO addr	Jump on TO = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag	2	2
JN1 addr	Jump on INT = 0	2	2
JBb addr	Jump on accumulator bit	2	2

Subroutine

Mnemonic	Description	Bytes	Cycles
CALL addr	Jump to subroutine	2	2
RETR	Return	1	2
RETR	Return and restore status	1	2

Flags

Mnemonic	Description	Bytes	Cycles
CLR C	Clear carry	1	1
CPL C	Complement carry	1	1
CLR F0	CLear flag 0	1	1
CPL F0	Complement flag 0	1	1
CLR F1	Clear flag 1	1	1
CPL F1	Complement flag 1	1	1

Data Moves

Mnemonic	Description	Bytes	Cycles
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, # data	Move immediate to A	2	2
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, # data	Move immediate to register	2	2
MOV @R, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1
XCHD A, @R	Exchange nibble of A and register	1	
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @	Move to A from page 3	1	2

Timer/Counter

Mnemonic	Description	Bytes	Cycles
MOV A, T	Read timer/counter	1	1
MOV T, A	Load timer/counter	1	1
STRT T	Start timer	1	1
STRT CNT	Start counter	1	1
STOP TCNT	Stop timer/counter	1	1
EN TCNT1	Enable timer/counter interrupt	1	1
DIS TCNT1	Disable timer/counter interrupt	1	1

Control

Mnemonic	Description	Bytes	Cycles
EN 1	Enable external interrupt	1	1
DIS 1	Disable external interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENT 0 CLK	Enable clock output on T0	1	1

Mnemonic	Description	Bytes	Cycles
NOP	No operation	1	1

AFN-01491B-03

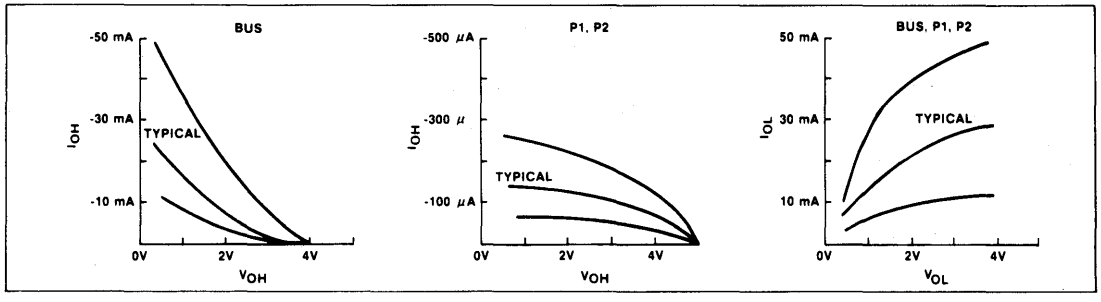
ABSOLUTE MAXIMUM RATINGS*

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 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS (TA = 0°C to 70°C, V_{CC} = V_{DD} = 5V + 10%, V_{SS} = 0V)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except $\overline{RESET}$, X1, X2)	-.5		.8	V	
V_{IL1}	Input Low Voltage ($\overline{RESET}$, X1, X2)	-.5		.6	V	
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.0		V_{CC}	V	
V_{IH1}	Input High Voltage (X1, X2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (BUS)			.45	V	I_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			.45	V	I_{OL} = 1.8 mA
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0 mA
V_{OL3}	Output Low Voltage (All Other Outputs)			.45	V	I_{OL} = 1.6 mA
VOH	Output High Voltage (BUS)	2.4			V	I_{OH} = -400μA
V_{OH1}	Output High Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	2.4			V	I_{OH} = -100μA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = -40μA
I_{L1}	Input Leakage Current (T1, $\overline{INT}$)			± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LI1}	Input Leakage Current (P10-P17, P20-P27, EA, $\overline{SS}$)			-500	μA	V_{SS} + .45$\leq V_{IN} \leq V_{CC}$
I_{L0}	Output Leakage Current (BUS, TO) (High Impedance State)			± 10	μA	V_{SS} + .45$\leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		4	8	mA	
I_{DD} + I_{CC}	Total Supply Current		40	80	mA	
V_{DD}	RAM Standby Pin Voltage	2.2		5.5	V	Standby Mode, Reset $\leq$0.6V



AFN-01491B-04

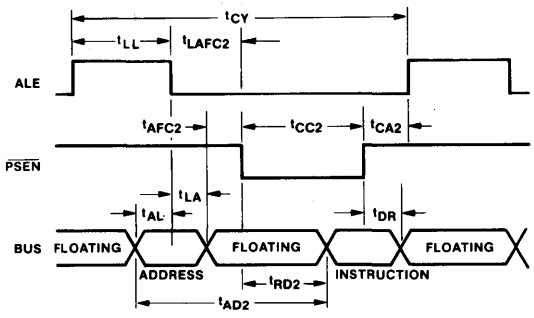
A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = V_{DD} = 5V ± 10%, V_{SS} = OV)

Symbol	Parameter	F (t_{CY})	8048H 8035HL 6 MHz Min.	8048H 8035HL 6 MHz Max.	8048H 8035HL 8 MHz Min.	8048H 8035HL 8 MHz Max.	8048H-1 8035HL-1 11 MHz Min.	8048H-1 8035HL-1 11 MHz Max.	Unit	Conditions (Note 1)
t_{LL}	ALE Pulse Width	7/30 t_{CY} -170	410		260		150			
t_{AL}	Addr Setup to ALE	1/5 t_{CY} -110	390		260		160			
t_{LA}	Addr Hold from ALE	1/15 t_{CY} -40	120		80		50			
t_{CC1}	Control Pulse Width ($\overline{RD}$, $\overline{WR}$)	1/2 t_{CY} -200	1050		730		480			
t_{CC2}	Control Pulse Width ($\overline{PSEN}$)	2/5 t_{CY} -200	800		550		350			
t_{DW}	Data Setup before $\overline{WR}$	13/30 t_{CY} -200	880		610		390			
t_{WD}	Data Hold after $\overline{WR}$	1/5 t_{CY} -150	350		220		120			(Note 2)
t_{DR}	Data Hold ($\overline{RD}$, $\overline{PSEN}$)	1/10 t_{CY} -30	0	220	0	160	0	110		
t_{RD1}	$\overline{RD}$ to Data in	2/5 t_{CY} -200		800		550		350		
t_{RD2}	$\overline{PSEN}$ to Data in	3/10 t_{CY} -200		550		360		210		
t_{AW}	Addr Setup to $\overline{WR}$	2/5 t_{CY} -150	850		600		300			
t_{AD1}	Addr Setup to Data ($\overline{RD}$)	23/30 t_{CY} -250		1670		1190		750		
t_{AD2}	Addr Setup to Data ($\overline{PSEN}$)	3/5 t_{CY} -250		1250		880		480		
t_{AFC1}	Addr Float to $\overline{RD}$, $\overline{WR}$	2/15 t_{CY} -40	290		210		140			
t_{AFC2}	Addr Float to $\overline{PSEN}$	1/30 t_{CY} -40	40		20		10			
t_{LAFC1}	ALE to Control ($\overline{RD}$, $\overline{WR}$)	1/5 t_{CY} -75	420		300		200			
t_{LAFC2}	ALE to Control ($\overline{PSEN}$)	1/10 t_{CY} -75	170		110		60			
t_{CA1}	Control to ALE ($\overline{RD}$, $\overline{WR}$, $\overline{PROG}$)	1/15 t_{CY} -40	120		80		50			
t_{CA2}	Control to ALE ($\overline{PSEN}$)	4/15 t_{CY} -40	620		460		320			
t_{CP}	Port Control Setup to $\overline{PROG}$	1/10 t_{CY} -40	210		140		100			
t_{PC}	Port Control Hold to $\overline{PROG}$	4/15 t_{CY} -200	460		300		160			
t_{PR}	$\overline{PROG}$ to P2 Input Valid	17/30 t_{CY} -120		1300		940		650		
t_{PF}	Input Data Hold from $\overline{PROG}$	1/10 t_{CY}		250	0	190	0	140		
t_{DP}	Output Data Setup	2/5 t_{CY} -150	850		600		400			
t_{PD}	Output Data Hold	1/10 t_{CY} -50	200		130		90			
t_{PP}	$\overline{PROG}$ Pulse Width	7/10 t_{CY} -250	1500		1060		700			
t_{PL}	Port 2 I/O Setup to ALE	4/15 t_{CY} -200	460		300		160			
t_{LP}	Port 2 I/O Hold to ALE	1/10 t_{CY} -100	150		80		40			
t_{PV}	Port Output from ALE	3/10 t_{CY} +100		850		660		510		
t_{CY}	Cycle Time		2.5		1.875		1.36			
t_{0PRR}	T0 Rep Rate	3/15 t_{CY}	500		370		270			

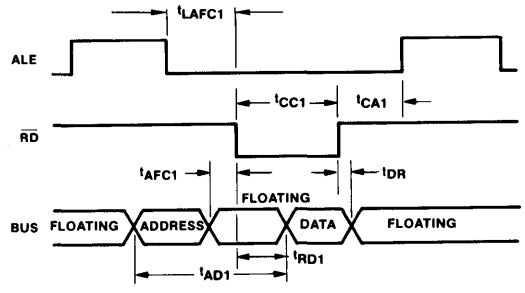
Notes:
1. Control Outputs CL = 80pF
   BUS Outputs CL = 150pF
2. BUS High Impedance Load 20pF

AFN-01491B-05

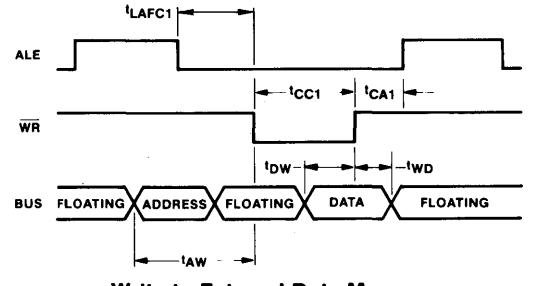
WAVEFORMS



Instruction Fetch From External Program Memory



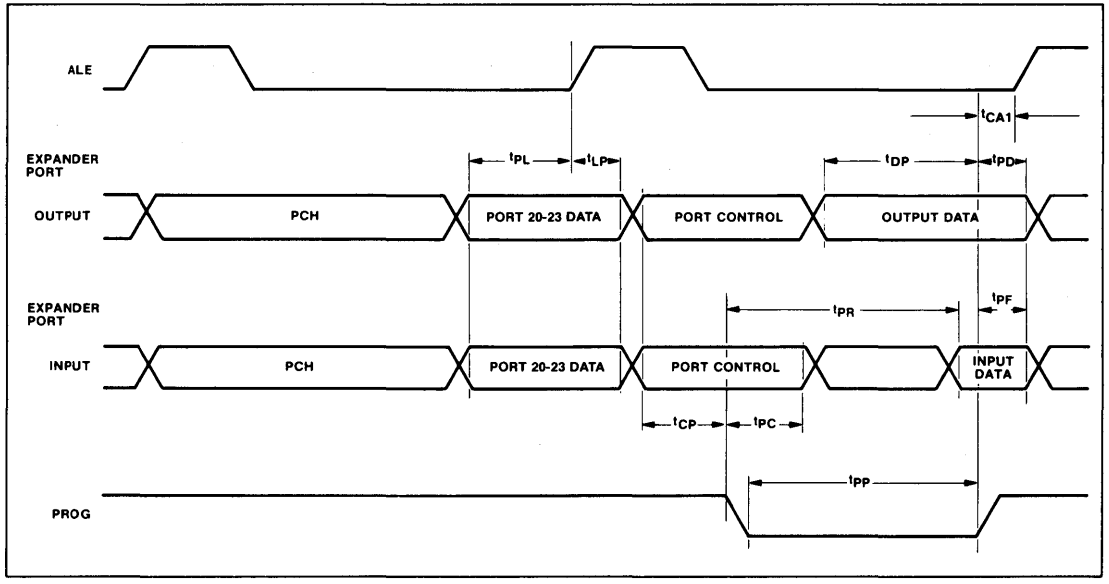
Read From External Data Memory



Write to External Data Memory

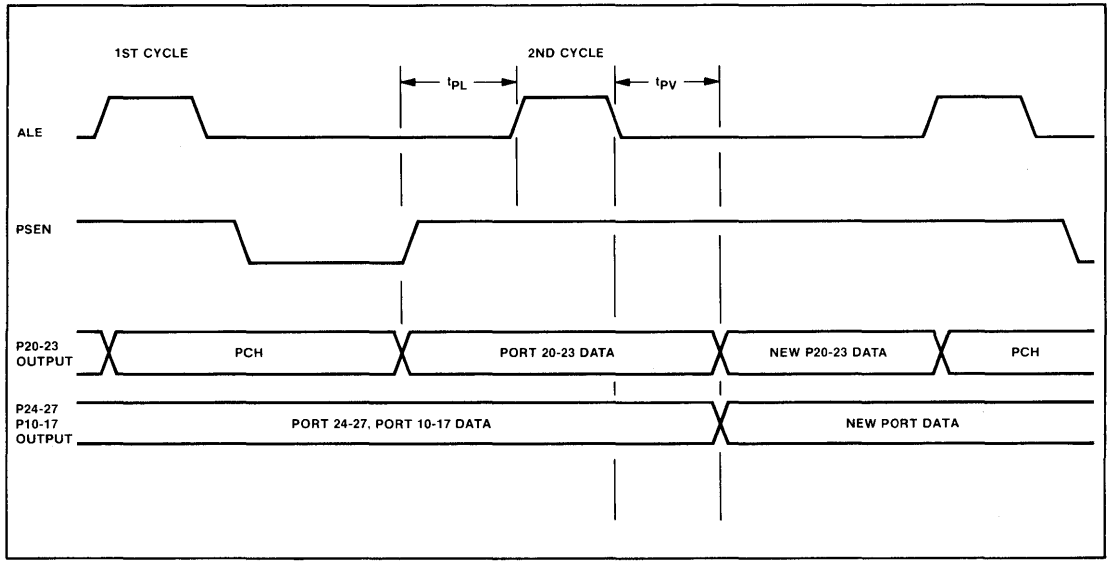
Input and Output for A.C.Tests

PORT 2 TIMING

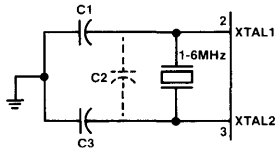


AFN-01491B-06

I/O PORT TIMING

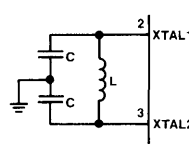


Crystal Oscillator Mode



C1 = 5pF + 1/2pF + STRAY
C2 = CRYSTAL + STRAY
C3 = 20pF + 1pF + STRAY

LC Oscillator Mode



$$f \approx \frac{1}{2\pi\sqrt{LC'}}$$

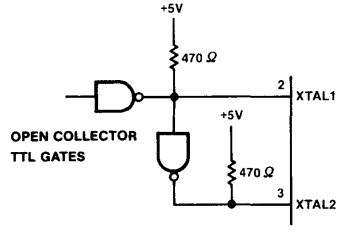
$$C' = \frac{C + 3C_{pp}}{2}$$

$C_{pp} \approx$ 5-10 pF PIN-TO-PIN CAPACITANCE

L	C	NOMINAL f
45 μH	20pF	5.2 MHz
120 μH	20pF	3.2 MHz

EACH C SHOULD BE APPROXIMATELY 20pF, INCLUDING STRAY CAPACITANCE.

Driving From External Source



AFN-01491B-07

PRELIMINARY

8048L
SPECIAL LOW POWER CONSUMPTION SINGLE COMPONENT 8-BIT MICROCOMPUTER

- **Typical Power Consumption 100mW**
- **Typical Standby Power 10mW**
  V_{DD} minimum of 2.2V
- **8-Bit CPU, ROM, RAM, I/O in Single Package**
- **4.17 μsec Instruction Cycle.**
  All Instructions 1 or 2 Cycles.
- **Over 90 Instructions: 70% Single Byte**
- **1K x 8 ROM**
  64 x 8 RAM
  27 I/O Lines
- **Interval Timer/Event Counter**
- **Easily Expandable Memory and I/O**
- **Compatible with 8080/8085 Series Peripherals**
- **Two Single Level Interrupts**

The Intel® 8048L is a totally self-sufficient 8-bit parallel computer fabricated on a single silicon chip using Intel's advanced N-channel silicon gate HMOS process, using special techniques to reduce operating and standby power consumption. The 8048L contains a 1K X 8 program memory, a 64 X 8 RAM data memory, 27 I/O lines, and an 8-bit timer/counter in addition to on-board oscillator and clock circuits. For systems that require extra capability the 8048L can be expanded using standard memories and MCS-80®/MCS-85® peripherals. The 8048L can be used with external ROM and RAM.

To reduce development problems to a minimum and provide maximum flexibility, a logically and functionally pin compatible version of the 8048L with UV-erasable user-programmable EPROM program memory is available. The 8748 will emulate the 8048L with greater power and other minor differences.

This microcontroller is designed to be an efficient controller as well as an arithmetic processor. The 8048L has extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting mostly of single byte instructions and no instructions over two bytes in length.

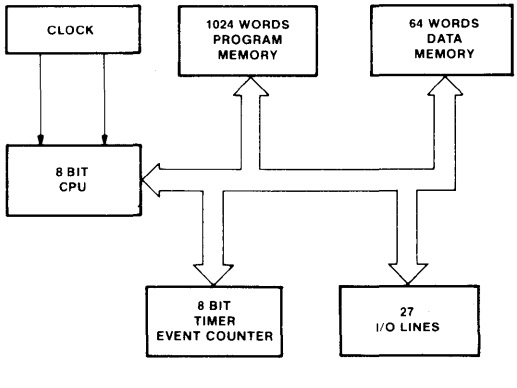


Figure 1.
8048L Block Diagram

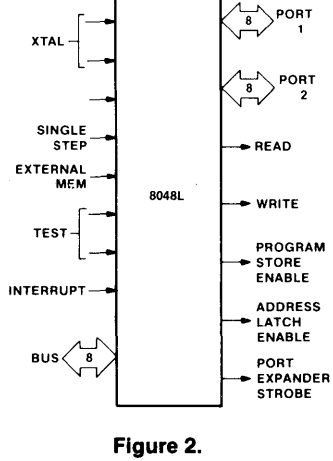


Figure 2.
8048L Logic Symbol

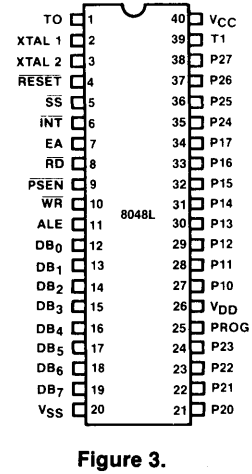


Figure 3.
8048L Pin Configuration

PIN DESCRIPTION

Designation	Pin =	Function
V_{SS}	20	Circuit GND potential
V_{DD}	26	Low power standby pin
V_{CC}	40	Main power supply; +5V during operation.
PROG	25	Output strobe for 8243 I/O expander.
P10-P17 Port 1	27-34	8-bit quasi-bidirectional port.
P20-27 Port 2	21-24	8-bit quasi-bidirectional port.
	35-38	P20-P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
DB0-DB7 BUS	12-19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched.
		Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.
T0	1	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction.
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. (Active low)
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the bus from an external device.
		Used as a read strobe to external data memory. (Active low)
$\overline{RESET}$	4	Input which is used to initialize the processor. (Active low) (Non TTL V_{IH})
$\overline{WR}$	10	Output strobe during a bus write. (Active low)
		Used as write strobe to external data memory.
ALE	11	Address latch enable. This signal occurs once during each cycle and is useful as a clock output.
		The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	Program store enable. This output occurs only during a fetch to external program memory. (Active low)
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction. (Active low)
EA	7	External access input which forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification. (Active high)
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source. (Non TTL V_{IH})
XTAL2	3	Other side of crystal input.

INSTRUCTION SET

Accumulator

Mnemonic	Description	Bytes	Cycles
ADD A, R	Add register to A	1	1
ADD A, @R	Add data memory to A	1	1
ADD A, # data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, # data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, # data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A @R	Or data memory to A	1	1
ORL A, # data	Or immediate to A	2	2
XRL A, R	Exclusive or register to A	1	1
XRL A, @R	Exclusive or data memory to A	1	1
XRL, A, # data	Exclusive or immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1

Input/Output

Mnemonic	Description	Bytes	Cycles
IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, # data	And immediate to port	2	2
ORL P, # data	Or immediate to port	2	2
INS A, BUS	Input BUS to A	1	2
OUTL BUS, A	Output A to BUS	1	2
ANL BUS, # data	And immediate to BUS	2	2
ORL BUS, # data	Or immediate to BUS	2	2
MOVD A,P	Input expander port to A	1	2
MOVD P, A	Output A to expander port	1	2
ANLD P, A	And A to expander port	1	2
ORLD P, A	Or A to expander port	1	2

Registers

Mnemonic	Description	Bytes	Cycles
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1

Branch

Mnemonic	Description	Bytes	Cycles
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and skip	2	2
JC addr	Jump on carry = 1	2	2
JNC addr	Jump on carry = 0	2	2
JZ addr	Jump on A zero	2	2
JNZ addr	Jump on A not zero	2	2
JTO addr	Jump on TO = 1	2	2
JNTO addr	Jump on TO = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag	2	2
JN1 addr	Jump on INT = 0	2	2
JBb addr	Jump on accumulator bit	2	2

Subroutine

Mnemonic	Description	Bytes	Cycles
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2

Flags

Mnemonic	Description	Bytes	Cycles
CLR C	Clear carry	1	1
CPL C	Complement carry	1	1
CLR F0	CLear flag 0	1	1
CPL F0	Complement flag 0	1	1
CLR F1	Clear flag 1	1	1
CPL F1	Complement flag 1	1	1

Data Moves

Mnemonic	Description	Bytes	Cycles
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, # data	Move immediate to A	2	2
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, # data	Move immediate to register	2	2
MOV @R, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1
XCHD A, @R	Exchange nibble of A and register	1	
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @	Move to A from page 3	1	2

Timer/Counter

Mnemonic	Description	Bytes	Cycles
MOV A, T	Read timer/counter	1	1
MOV T, A	Load timer/counter	1	1
STRT T	Start timer	1	1
STRT CNT	Start counter	1	1
STOP TCNT	Stop timer/counter	1	1
EN TCNTI	Enable timer/counter interrupt	1	1
DIS TCNTI	Disable timer/counter interrupt	1	1

Control

Mnemonic	Description	Bytes	Cycles
EN 1	Enable external interrupt	1	1
DIS 1	Disable external interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENT 0 CLK	Enable clock output on T0	1	1

Mnemonic	Description	Bytes	Cycles
NOP	No operation	1	1

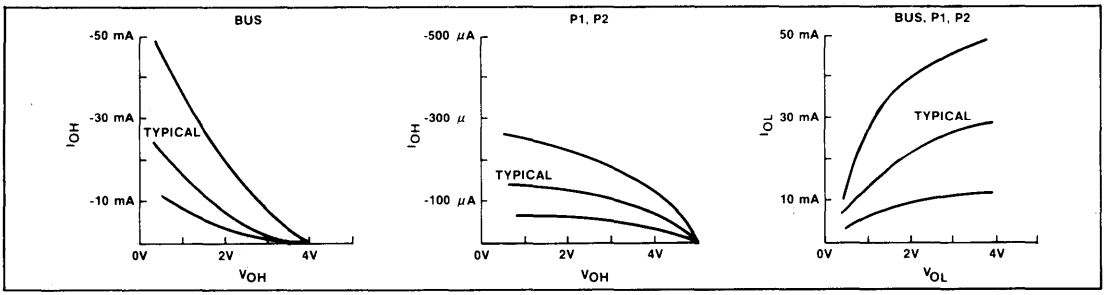
ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... -65°C to + 125°C
Voltage On Any Pin With Respect to Ground ..... -0.5V to +7V
Power Dissipation ..... 1.5 Watt

** COMMENT Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of device at these or any other conditions above those indicated in the operational sections of this specification is not implied.*

D.C. AND OPERATING CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = V_{DD} = 5V ± 10%, V_{SS} = 0V

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except $\overline{RESET}$, X1, X2)	-.5		.8	V	
V_{IL1}	Input Low Voltage ($\overline{RESET}$, X1, X2)	-.5		.6	V	
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.0		V_{CC}	V	
V_{IH1}	Input High Voltage (X1, X2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (BUS)			.45	V	V_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			.45	V	I_{OL} = 1.8 mA
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0 mA
V_{OL3}	Output Low Voltage (All Other Outputs)			.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage (BUS)	2.4			V	I_{OH} = −400 μA
V_{OH1}	Output High Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	2.4			V	I_{OH} = −100 μA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = −40 μA
I_{L1}	Input Leakage Current (T1, $\overline{INT}$)			± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LI1}	Input Leakage Current (P10-P17, P20-P27, EA, $\overline{SS}$)			-500	μA	V_{SS} + .45 $\leq V_{IN} \leq V_{CC}$
I_{L0}	Output Leakage Current (BUS, TO) (High Impedance State)			± 10	μA	V_{SS} + .45 $\leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		2	4	mA	
I_{DD} + I_{CC}	Total Supply Current		20	40	mA	
V_{DD}	Ram Standby Pin Voltage	2.2		5.5	V	Standby Mode, Reset $\leq$ 0.6V



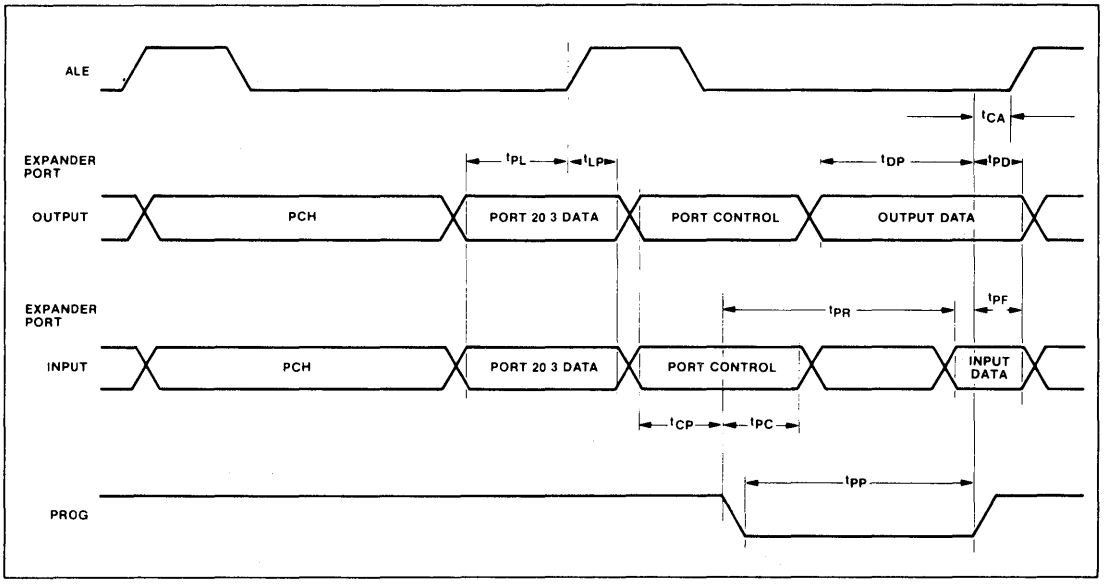
A.C. CHARACTERISTICS (PORT 2 TIMING)

T_A = 0° C to 70° C, V_{CC} = 5V + 10%, V_{SS} = 0V

TCY = 4.17 μS

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CP}	Port Control Setup Before Falling Edge of PROG	185		ns	
t_{PC}	Port Control Hold After Falling Edge of PROG	160		ns	
t_{PR}	PROG to Time P2 Input Must Be Valid		1.35	μs	
t_{PF}	Input Data Hold Time	0	250	ns	
t_{DP}	Output Data Setup Time	420		ns	
t_{PD}	Output Data Hold Time	110		ns	
t_{PP}	PROG Pulse Width	2.0		μs	
t_{PL}	Port 2 I/O Data Setup	585		ns	
t_{LP}	Port 2 I/O Data Hold	250		ns	

PORT 2 TIMING



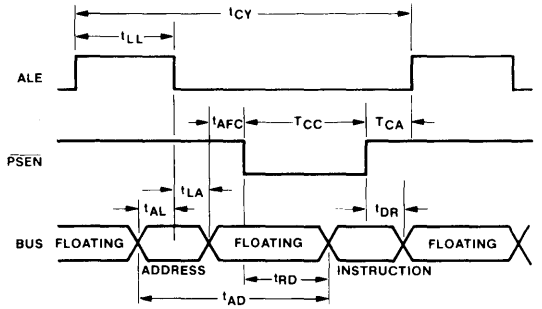
BUS TIMING AS A FUNCTION OF TCY *

SYMBOL	FUNCTION OF TCY			
T_{LL}	7/30	T_{CY}	MIN	
T_{AL}	2/15	T_{CY}	MIN	
T_{LA}	1/15	T_{CY}	MIN	
T_{CC} (1)	1/2	T_{CY}	MIN	T_{CC} (1) : $\overline{RD}/\overline{WR}$
T_{CC} (2)	2/5	T_{CY}	MIN	T_{CC} (2) : $\overline{PSEN}$
T_{DW}	13/30	T_{CY}	MIN	
T_{WD}	1/15	T_{CY}	MIN	
T_{DR}	0		MIN	

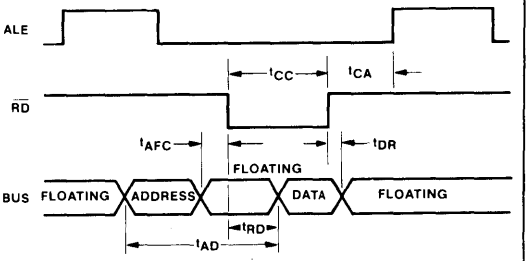
SYMBOL	FUNCTION OF TCY			
T_{RD} (1)	2/5	T_{CY}	MAX	T_{RD} (1) : $\overline{RD}$
T_{RD} (2)	3/10	T_{CY}	MAX	T_{RD} (2) : $\overline{PSEN}$
T_{AW}	1/3	T_{CY}	MIN	
T_{AD} (1)	11/15	T_{CY}	MAX	T_{AD} (1) : $\overline{RD}$
T_{AD} (2)	8/15	T_{CY}	MAX	T_{AD} (2) : $\overline{PSEN}$
T_{AFC} (1)	2/15	T_{CY}	MIN	T_{AFC} (1): $\overline{RD}$
T_{AFC} (2)	1/30	T_{CY}	MIN	T_{AFC} (2): $\overline{PSEN}$
T_{CA} (1)	1/15	T_{CY}	MIN	T_{CA} (1) : $\overline{RD}$, $\overline{WR}$
T_{CA} (2)	2/15	T_{CY}	MIN	T_{CA} (2) : $\overline{PSEN}$

*** APPROXIMATE VALUES NOT INCLUDING GATE DELAYS.**

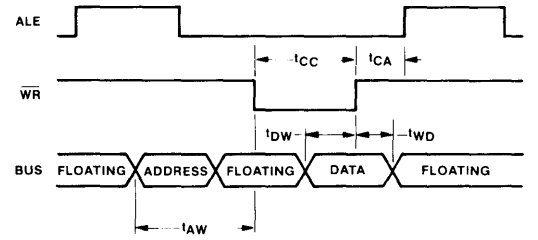
WAVEFORMS



Instruction Fetch From External Program Memory



Read From External Data Memory



Write to External Data Memory

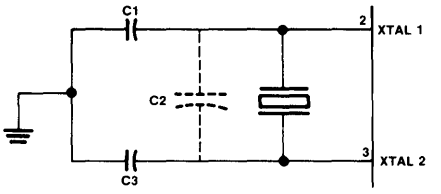
Input and Output for A.C.Tests

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = 5V + 10\%$, $V_{SS} = 0V$

Symbol	Parameter	Min.	Max.	Unit	Conditions (Note 1)
t_{LL}	ALE Pulse Width	600		ns	
t_{AL}	Address Setup to ALE	150		ns	
t_{LA}	Address Hold from ALE	80		ns	
t_{CC}	Control Pulse Width (PSEN, RD, WR)	1500		ns	
t_{DW}	Data Setup before WR	640		ns	
t_{WD}	Data Hold After WR	120		ns	C_L = 20pF
t_{CY}	Cycle Time	4.17	15.0	μs	
t_{DR}	Data Hold	0	200	ns	
t_{RD}	PSEN, RD to Data In		750	ns	
t_{AW}	Address Setup to WR	260		ns	
t_{AD}	Address Setup to Data In		1450	ns	
t_{AFC}	Address Float to RD, PSEN	0		ns	
t_{CA}	Control Pulse to ALE	20		ns	

Note 1: Control outputs: C_L = 80 pF
BUS Outputs: C_L = 150 pF

CRYSTAL OSCILLATOR MODE

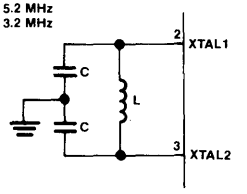


C1 = 5pF ± 1/2pF + STRAY < 5pF
C2 = CRYSTAL + STRAY < 8pF
C3 = 20pF ± 1pF + STRAY < 5pF

CRYSTAL SERIES RESISTANCE SHOULD BE LESS THAN 75Ω AT 6 MHz LESS THAN 180Ω AT 3.6MHz

LC OSCILLATOR MODE

L	C	NOMINAL 1
45pH	20pF	5.2 MHz
120pH	20pF	3.2 MHz



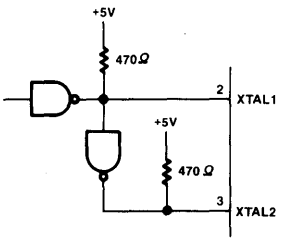
$$F = \frac{1}{2\pi\sqrt{LC}}$$

$$C = \frac{C + Cpp}{2}$$

Cpp = 5-10 pF PIN TO PIN CAPACITANCE

EACH C SHOULD BE APPROXIAMTELY 20pF. INCLUDING STRAY CAPACITANCE

DRIVING FROM EXTERNAL SOURCE



XTAL 1 MUST BE HIGH 35-65% OF THE PERIOD AND XTAL 2 MUST BE HIGH 35-65% OF THE PERIOD. RISE AND FALL TIMES MUST NOT EXCEED 20ns.

8049H/8039HL
HMOS SINGLE COMPONENT 8-BIT MICROCOMPUTER

- **8049H Mask Programmable ROM**
- **8039HL CPU Only with Power Down Mode**

- **8-BIT CPU, ROM, RAM, I/O in Single Package**
- **High Performance HMOS**
- **Reduced Power Consumption**
- **1.4 usec and 1.9 µsec Cycle Versions All Instructions 1 or 2 Cycles.**
- **Over 90 instructions: 70% Single Byte**
- **1K x 8 ROM**
  64 x 8 RAM
  27 I/O Lines
- **Interval Timer/Event Counter**
- **Easily Expandable Memory and I/O**
- **Compatible with 8080/8085 Series Peripherals**
- **Two Single Level Interrupts**

The Intel® 8049H/8039HL are totally self-sufficient, 8-bit parallel computers fabricated on single silicon chips using Intel's advanced N-channel silicon gate HMOS process.

The 8049H contains a 2K X 8 program memory, a 128 X 8 RAM data memory, 27 I/O lines, and an 8-bit timer/counter in addition to on-board oscillator and clock circuits. For systems that require extra capability the 8049H can be expanded using standard memories and MCS-80®/MCS-85® peripherals. The 8039HL is the equivalent of the 8049H without program memory and can be used with external ROM and RAM.

To reduce development problems to a minimum and provide maximum flexibility, a logically and functionally pin compatible version of the 8049H with UV-erasable user-programmable EPROM program memory will soon be available. The 8749 will emulate the 8049H up to 1 MHz clock frequency with minor differences.

The 8049H is fully compatible with the 8049.

These microcomputers are designed to be efficient controllers as well as arithmetic processors. They have extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting mostly of single byte instructions and no instructions over 2 bytes in length.

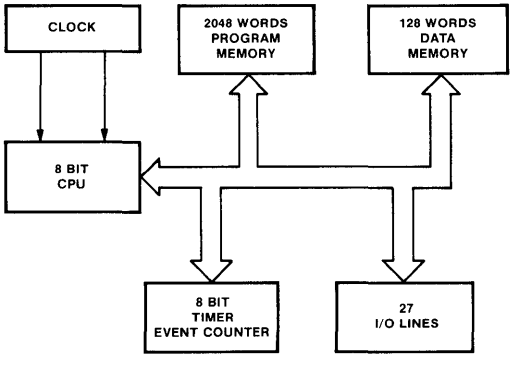


Figure 1. Block Diagram

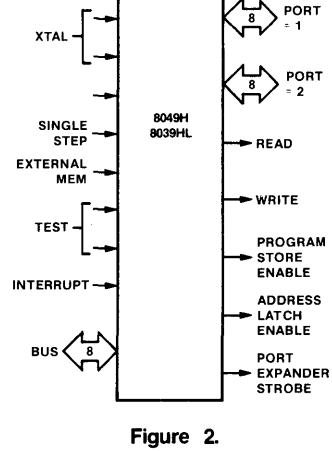


Figure 2. Logic Symbol

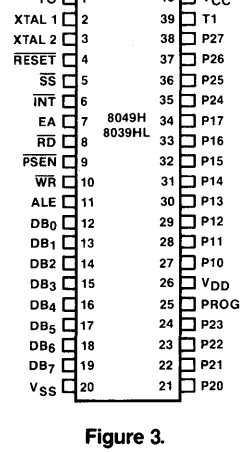


Figure 3. Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Function
V_{SS}	20	Circuit GND potential
V_{DD}	26	Low power standby pin
V_{CC}	40	Main power supply; +5V during operation.
PROG	25	Output strobe for 8243 I/O expander.
P10-P17 Port 1	27-34	8-bit quasi-bidirectional port.
P20-P27 Port 2	21-24	8-bit quasi-bidirectional port.
	35-38	P20-P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
DB0-DB7 BUS	12-19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched.
		Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.
TO	1	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction.
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. (Active low)

Symbol	Pin No.	Function
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the bus from an external device.
		Used as a read strobe to external data memory. (Active low)
$\overline{RESET}$	4	Input which is used to initialize the processor. (Active low) (Non TTL V_{IH})
$\overline{WR}$	10	Output strobe during a bus write. (Active low)
		Used as write strobe to external data memory.
ALE	11	Address latch enable. This signal occurs once during each cycle and is useful as a clock output.
		The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	Program store enable. This output occurs only during a fetch to external program memory. (Active low)
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction. (Active low)
EA	7	External access input which forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification. (Active high)
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source. (Non TTL V_{IH})
XTAL2	3	Other side of crystal input.

Table 2. Instruction Set

Accumulator

Mnemonic	Description	Bytes	Cycles
ADD A, R	Add register to A	1	1
ADD A, @R	Add data memory to A	1	1
ADD A, # data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, # data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, # data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A @R	Or data memory to A	1	1
ORL A, # data	Or immediate to A	2	2
XRL A, R	Exclusive or register to A	1	1
XRL A, @R	Exclusive or data memory to A	1	1
XRL, A, # data	Exclusive or immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1

Input/Output

Mnemonic	Description	Bytes	Cycles
IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, # data	And immediate to port	2	2
ORL P, # data	Or immediate to port	2	2
INS A, BUS	Input BUS to A	1	2
OUTL BUS, A	Output A to BUS	1	2
ANL BUS, # data	And immediate to BUS	2	2
ORL BUS, # data	Or immediate to BUS	2	2
MOVD A,P	Input expander port to A	1	2
MOVD P, A	Output A to expander port	1	2
ANLD P, A	And A to expander port	1	2
ORLD P, A	Or A to expander port	1	2

Registers

Mnemonic	Description	Bytes	Cycles
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1

Branch

Mnemonic	Description	Bytes	Cycles
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and skip	2	2
JC addr	Jump on carry = 1	2	2
JNC addr	Jump on carry = 0	2	2
JZ addr	Jump on A zero	2	2
JNZ addr	Jump on A not zero	2	2
JTO addr	Jump on TO = 1	2	2
JNTO addr	Jump on TO = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag	2	2
JN1 addr	Jump on INT = 0	2	2
JBb addr	Jump on accumulator bit	2	2

Subroutine

Mnemonic	Description	Bytes	Cycles
CALL addr	Jump to subroutine	2	2
RETR	Return	1	2
RETR	Return and restore status	1	2

Flags

Mnemonic	Description	Bytes	Cycles
CLR C	Clear carry	1	1
CPL C	Complement carry	1	1
CLR F0	CLear flag 0	1	1
CPL F0	Complement flag 0	1	1
CLR F1	Clear flag 1	1	1
CPL F1	Complement flag 1	1	1

Data Moves

Mnemonic	Description	Bytes	Cycles
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, # data	Move immediate to A	2	2
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, # data	Move immediate to register	2	2
MOV @R, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1
XCHD A, @R	Exchange nibble of A and register	1	1
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @	Move to A from page 3	1	2

Timer/Counter

Mnemonic	Description	Bytes	Cycles
MOV A, T	Read timer/counter	1	1
MOV T, A	Load timer/counter	1	1
STRT T	Start timer	1	1
STRT CNT	Start counter	1	1
STOP TCNT	Stop timer/counter	1	1
EN TCNT1	Enable timer/counter interrupt	1	1
DIS TCNT1	Disable timer/counter interrupt	1	1

Control

Mnemonic	Description	Bytes	Cycles
EN 1	Enable external interrupt	1	1
DIS 1	Disable external interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENT 0 CLK	Enable clock output on T0	1	1

Mnemonic	Description	Bytes	Cycles
NOP	No operation	1	1

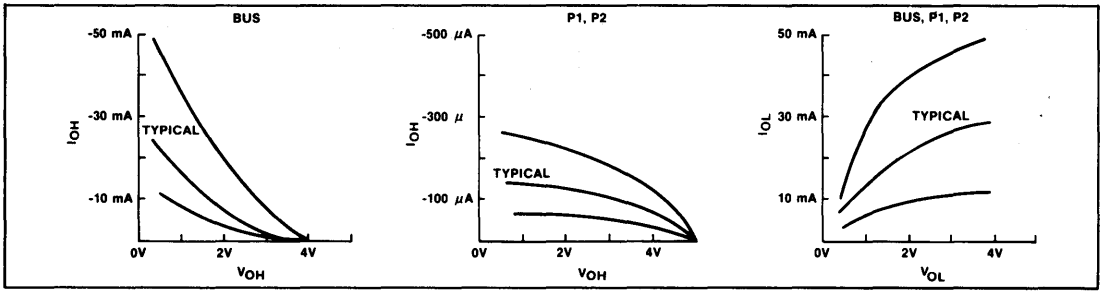
ABSOLUTE MAXIMUM RATINGS*

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 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = V_{DD} = 5V ± 10%, V_{SS} = 0V)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except $\overline{RESET}$, X1, X2)	-.5		.8	V	
V_{IL1}	Input Low Voltage ($\overline{RESET}$, X1, X2)	-.5		.6	V	
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.0		V_{CC}	V	
V_{IH1}	Input High Voltage (X1, X2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (BUS)			.45	V	I_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			.45	V	I_{OL} = 1.8 mA
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0 mA
V_{OL3}	Output Low Voltage (All Other Outputs)			.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage (BUS)	2.4			V	I_{OH} = -400 μA
V_{OH1}	Output High Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	2.4			V	I_{OH} = -100 μA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = -40 μA
I_{L1}	Input Leakage Current (T1, $\overline{INT}$)			± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LI1}	Input Leakage Current (P10-P17, P20-P27, EA, $\overline{SS}$)			-500	μA	V_{SS} + .45 $\leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current (BUS, TO) (High Impedance State)			± 10	μA	V_{SS} + .45 $\leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		5	10	mA	
I_{DD} + I_{CC}	Total Supply Current		50	100	mA	



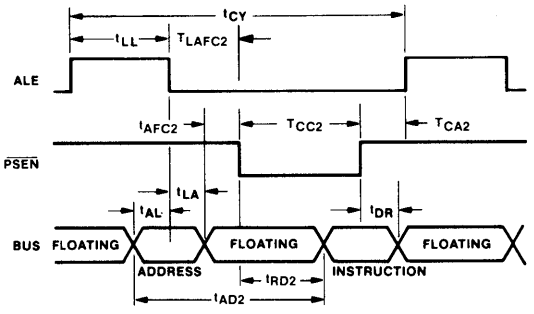
AFN-01784A-04

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = V_{DD} = 5V ± 10%, V_{SS} = 0V)

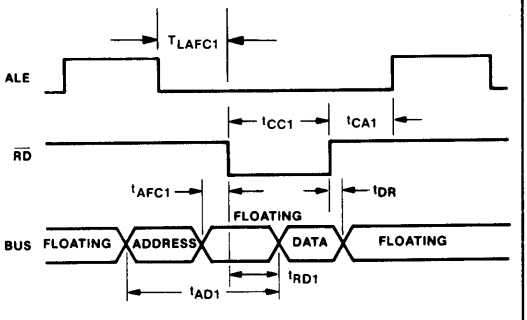
Symbol	Parameter	f (t_{CY}) (Note 3)	11 MHz Min.	11 MHz Max.	Unit	Conditions (Note 1)
t_{LL}	ALE Pulse Width	7/30 t_{CY} -170	150		ns	
t_{AL}	Addr Setup to ALE	1/5 t_{CY} -110	160		ns	
t_{LA}	Addr Hold from ALE	1/15 t_{CY} -40	50		ns	
t_{CC1}	Control Pulse Width ($\overline{RD}$, $\overline{WR}$)	1/2 t_{CY} -200	480		ns	
t_{CC2}	Control Pulse Width ($\overline{PSEN}$)	2/5 t_{CY} -200	350		ns	
t_{DW}	Data Setup before $\overline{WR}$	13/30 t_{CY} -200	390		ns	
t_{WD}	Data Hold after $\overline{WR}$	1/5 t_{CY} -150	120		ns	(Note 2)
t_{DR}	Data Hold ($\overline{RD}$, $\overline{PSEN}$)	1/10 t_{CY} -30	0	110	ns	
t_{RD1}	$\overline{RD}$ to Data in	2/5 t_{CY} -200		350	ns	
t_{RD2}	$\overline{PSEN}$ to Data in	3/10 t_{CY} -200		210	ns	
t_{AW}	Addr Setup to $\overline{WR}$	2/5 t_{CY} -150	300		ns	
t_{AD1}	Addr Setup to Data ($\overline{RD}$)	23/30 t_{CY} -250		750	ns	
t_{AD2}	Addr Setup to Data ($\overline{PSEN}$)	3/5 t_{CY} -250		480	ns	
t_{AFC1}	Addr Float to $\overline{RD}$, $\overline{WR}$	2/15 t_{CY} -40	140		ns	
t_{AFC2}	Addr Float to $\overline{PSEN}$	1/30 t_{CY} -40	10		ns	
t_{LAFC1}	ALE to Control, ($\overline{RD}$, $\overline{WR}$)	1/5 t_{CY} -75	200		ns	
t_{LAFC2}	ALE to Control ($\overline{PSEN}$)	1/10 t_{CY} -75	60		ns	
t_{CA1}	Control to ALE ($\overline{RD}$, $\overline{WR}$, $\overline{PROG}$)	1/15 t_{CY} -40	50		ns	
t_{CA2}	Control to ALE ($\overline{PSEN}$)	4/15 t_{CY} -40	320		ns	
t_{CP}	Port Control Setup to $\overline{PROG}$	1/10 t_{CY} -40	100		ns	
t_{PC}	Port Control Hold to $\overline{PROG}$	4/15 t_{CY} -200	160		ns	
t_{PR}	$\overline{PROG}$ to P2 Input Valid	17/30 t_{CY} -120		650	ns	
t_{PF}	Input Data Hold from $\overline{PROG}$	1/10 t_{CY}	0	140	ns	
t_{DP}	Output Data Setup	2/5 t_{CY} -150	400		ns	
t_{PD}	Output Data Hold	1/10 t_{CY} -50	90		ns	
t_{PP}	$\overline{PROG}$ Pulse Width	7/10 t_{CY} -250	700		ns	
tPL	Port 2 I/O Setup to ALE	4/15 t_{CY} -200	160		ns	
t_{LP}	Port 2 I/O Hold to ALE	1/10 t_{CY} -100	40		ns	
t_{PV}	Port Output from ALE	3/10 t_{CY} +100		510	ns	
t_{CY}	Cycle Time		1.36		µs	
t_{0PRR}	t0 Rep Rate	3/15 t_{CY}	270		ns	

Notes:
1. Control Outputs CL = 80pF
   BUS Outputs CL = 150pF
2. BUS High Impedance Load 20pF
3. Calculated values will be equal to or better than published 8049 values.

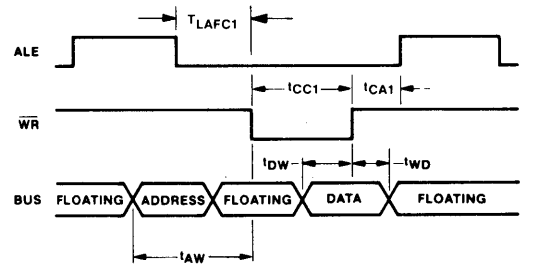
WAVEFORMS



Instruction Fetch From External Program Memory



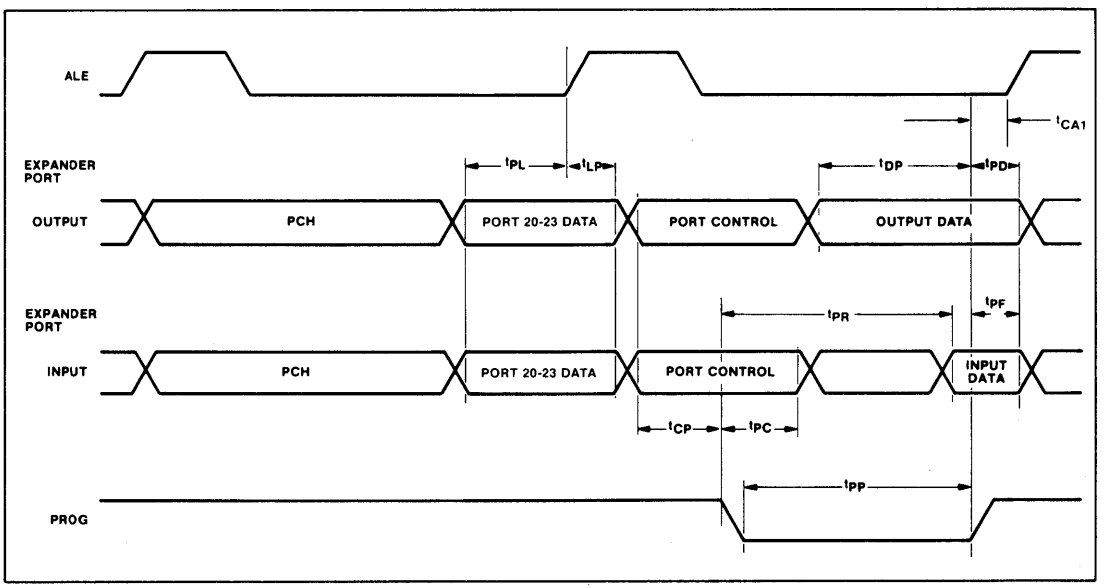
Read From External Data Memory



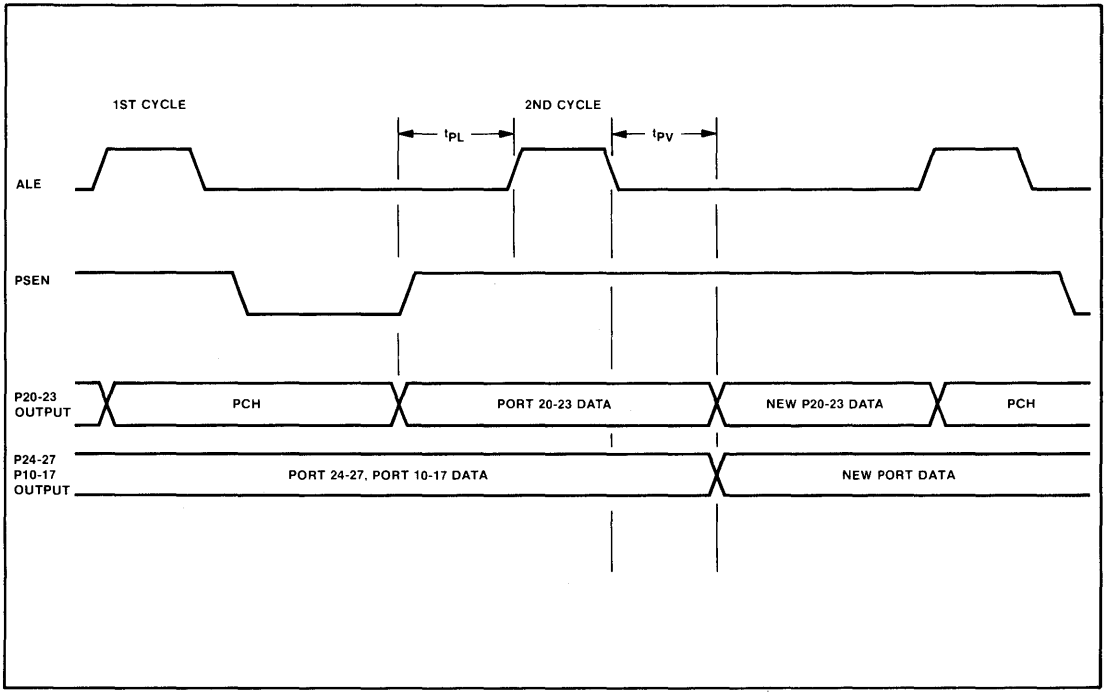
Write to External Data Memory

Input and Output for A.C. Tests

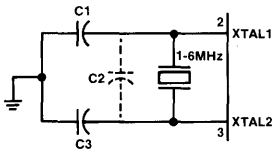
PORT 2 EXPANDER TIMING



I/O PORT TIMING



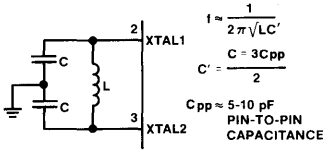
OSCILLATOR MODE



C1 = 5pF ± 1/2pF + STRAY < 5pF
C2 = CRYSTAL ± STRAY < 8pF
C3 = 20pF + 1pF ± STRAY < 5pF

CRYSTAL SERIES RESISTANCE SHOULD BE LESS THAN 75 Ω AT 6MHz; LESS THAN 180 Ω AT 3.6MHz.

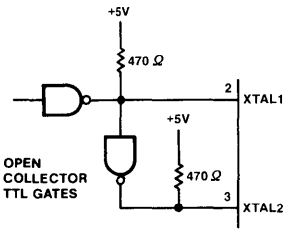
LC OSCILLATOR MODE



L	C	NOMINAL f
45 μH	20pF	5.2 MHz
120 μH	20pF	3.2 MHz

EACH C SHOULD BE APPROXIMATELY 20pF, INCLUDING STRAY CAPACITANCE.

DRIVING FROM EXTERNAL SOURCE



FOR THE 8048, XTAL1 MUST BE HIGH 35-65% OF THE PERIOD AND XTAL2 MUST BE HIGH 35-65% OF THE PERIOD.

RISE AND FALL TIMES MUST NOT EXCEED 20ns.

8243
MCS-48® INPUT/OUTPUT EXPANDER

- **Low Cost**
- **Simple Interface to MCS-48® Microcomputers**
- **Four 4-Bit I/O Ports**
- **AND and OR Directly to Ports**
- **24-Pin DIP**
- **Single 5V Supply**
- **High Output Drive**
- **Direct Extension of Resident 8048 I/O Ports**

The Intel® 8243 is an input/output expander designed specifically to provide a low cost means of I/O expansion for the MCS-48® family of single chip microcomputers. Fabricated in 5 volts NMOS, the 8243 combines low cost, single supply voltage and high drive current capability.

The 8243 consists of four 4-bit bidirectional static I/O ports and one 4-bit port which serves as an interface to the MCS-48 microcomputers. The 4-bit interface requires that only 4 I/O lines of the 8048 be used for I/O expansion, and also allows multiple 8243's to be added to the same bus.

The I/O ports of the 8243 serve as a direct extension of the resident I/O facilities of the MCS-48 microcomputers and are accessed by their own MOV, ANL, and ORL instructions.

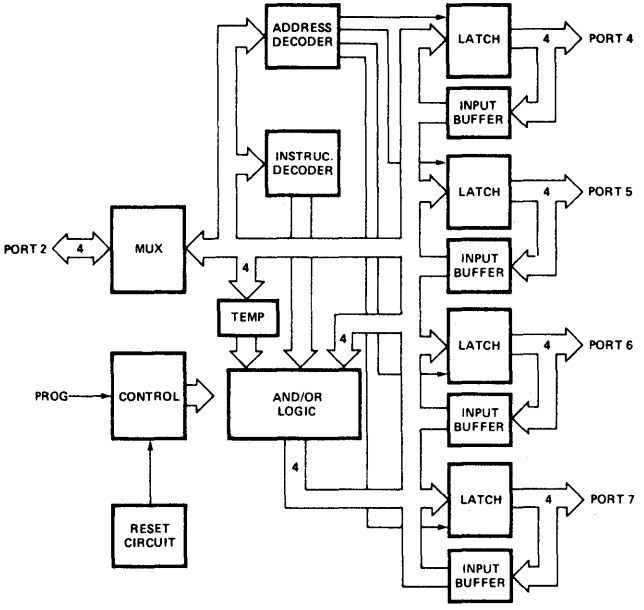


Figure 1. 8243 Block Diagram

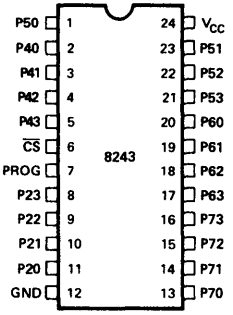


Figure 2. 8243 Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Function
PROG	7	Clock Input. A high to low transition on PROG signifies that address and control are available on P20-P23, and a low to high transition signifies that data is available on P20-P23.
$\overline{CS}$	6	Chip Select Input. A high on CS inhibits any change of output or internal status.
P20-P23	11-8	Four (4) bit bi-directional port contains the address and control bits on a high to low transition of PROG. During a low to high transition contains the data for a selected output port if a write operation, or the data from a selected port before the low to high transition if a read operation.
GND	12	0 volt supply.
P40-P43 P50-P53 P60-P63 P70-P73	2-5 1, 23-21 20-17 13-16	Four (4) bit bi-directional I/O ports. May be programmed to be input (during read), low impedance latched output (after write), or a tri-state (after read). Data on pins P20-P23 may be directly written, ANDed or ORed with previous data.
V_{CC}	24	+5 volt supply.

FUNCTIONAL DESCRIPTION

General Operation

The 8243 contains four 4-bit I/O ports which serve as an extension of the on-chip I/O and are addressed as ports 4-7. The following operations may be performed on these ports:

- Transfer Accumulator to Port.
- Transfer Port to Accumulator.
- AND Accumulator to Port.
- OR Accumulator to Port.

All communication between the 8048 and the 8243 occurs over Port 2 (P20-P23) with timing provided by an output pulse on the PROG pin of the processor. Each transfer consists of two 4-bit nibbles:

The first containing the "op code" and port address and the second containing the actual 4-bits of data. A high to low transition of the PROG line indicates that address is present while a low to high transition indicates the presence of data. Additional 8243's may be added to the 4-bit bus and chip selected using additional output lines from the 8048/8748/8035.

Power On Initialization

Initial application of power to the device forces input/output ports 4, 5, 6, and 7 to the tri-state and port 2 to the input mode. The PROG pin may be either high or low when power is applied. The first high to low transition of PROG causes device to exit power on mode. The power on sequence is initiated if VCC drops below 1V.

P21	P20	Address Code	P23	P22	Instruction Code
0	0	Port 4	0	0	Read
0	1	Port 5	0	1	Write
1	0	Port 6	1	0	ORLD
1	1	Port 7	1	1	ANLD

Write Modes

The device has three write modes. MOVD Pi, A directly writes new data into the selected port and old data is lost. ORLD Pi, A takes new data, OR's it with the old data and then writes it to the port. ANLD Pi, A takes new data, AND's it with the old data and then writes it to the port. Operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. On the low to high transition of PROG data on port 2 is transferred to the logic block of the specified output port.

After the logic manipulation is performed, the data is latched and outputed. The old data remains latched until new valid outputs are entered.

Read Mode

The device has one read mode. The operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. As soon as the read operation and port address are decoded, the appropriate outputs are tri-stated, and the input buffers switched on. The read operation is terminated by a low to high transition of the PROG pin. The port (4, 5, 6 or 7) that was selected is switched to the tri-stated mode while port 2 is returned to the input mode.

Normally, a port will be in an output (write mode) or input (read mode). If modes are changed during operation, the first read following a write should be ignored; all following reads are valid. This is to allow the external driver on the port to settle after the first read instruction removes the low impedance drive from the 8243 output. A read of any port will leave that port in a high impedance state.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin
With Respect to Ground ..... -0.5 V to +7V
Power Dissipation ..... 1 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

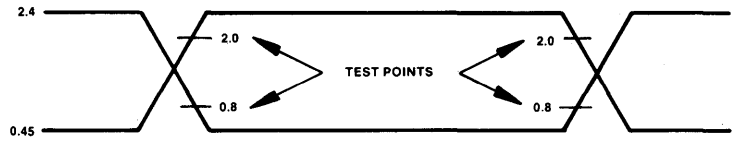
D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V 10%

Symbol	Parameter	Min.	Typ.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+0.5	V	
V_{OL1}	Output Low Voltage Ports 4-7			0.45	V	I_{OL} = 4.5 mA*
V_{OL2}	Output Low Voltage Port 7			1	V	I_{OL} = 20 mA
V_{OH1}	Output High Voltage Ports 4-7	2.4			V	I_{OH} = 240μA
I_{IL1}	Input Leakage Ports 4-7	-10		20	μA	Vin = V_{CC} to OV
I_{IL2}	Input Leakage Port 2, CS, PROG	-10		10	μA	Vin = V_{CC} to OV
V_{OL3}	Output Low Voltage Port 2			.45	V	I_{OL} = 0.6 mA
I_{CC}	V_{CC} Supply Current		10	20	mA	
V_{OH2}	Output Voltage Port 2	2.4				I_{OH} = 100μA
I_{OL}	Sum of all I_{OL} from 16 Outputs			72	mA	4.5 mA Each Pin

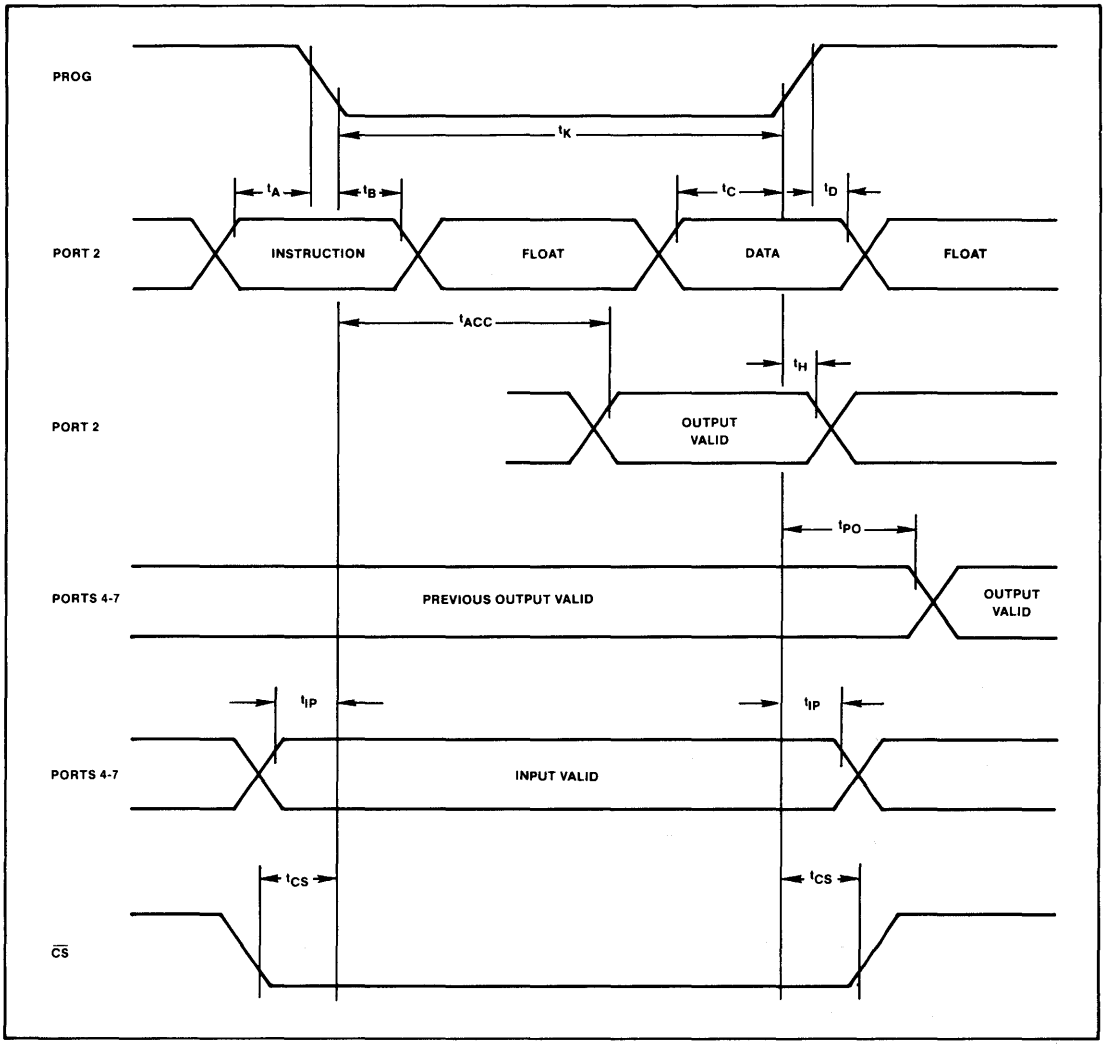
*See following graph for additional sink current capability

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V 10%

Symbol	Parameter	Min.	Max.	Units	Test Conditions
tA	Code Valid Before PROG	100		ns	80 pF Load
tB	Code Valid After PROG	60		ns	20 pF Load
tC	Data Valid Before PROG	200		ns	80 pF Load
tD	Data Valid After PROG	20		ns	20 pF Load
tH	Floating After PROG	0	150	ns	20 pF Load
tK	PROG Negative Pulse Width	700		ns	
tCS	CS Valid Before/After PROG	50		ns	
tPO	Ports 4-7 Valid After PROG		700	ns	100 pF Load
tLP1	Ports 4-7 Valid Before/After PROG	100		ns	
tACC	Port 2 Valid After PROG		650	ns	80 pF Load



WAVEFORMS


PROG
t_K
t_A
t_B
t_C
t_D
PORT 2
INSTRUCTION
FLOAT
DATA
FLOAT
t_{ACC}
t_H
PORT 2
OUTPUT VALID
t_{PO}
PORTS 4-7
PREVIOUS OUTPUT VALID
OUTPUT VALID
t_{IP}
t_{IP}
PORTS 4-7
INPUT VALID
t_{CS}
t_{CS}
$\overline{CS}$

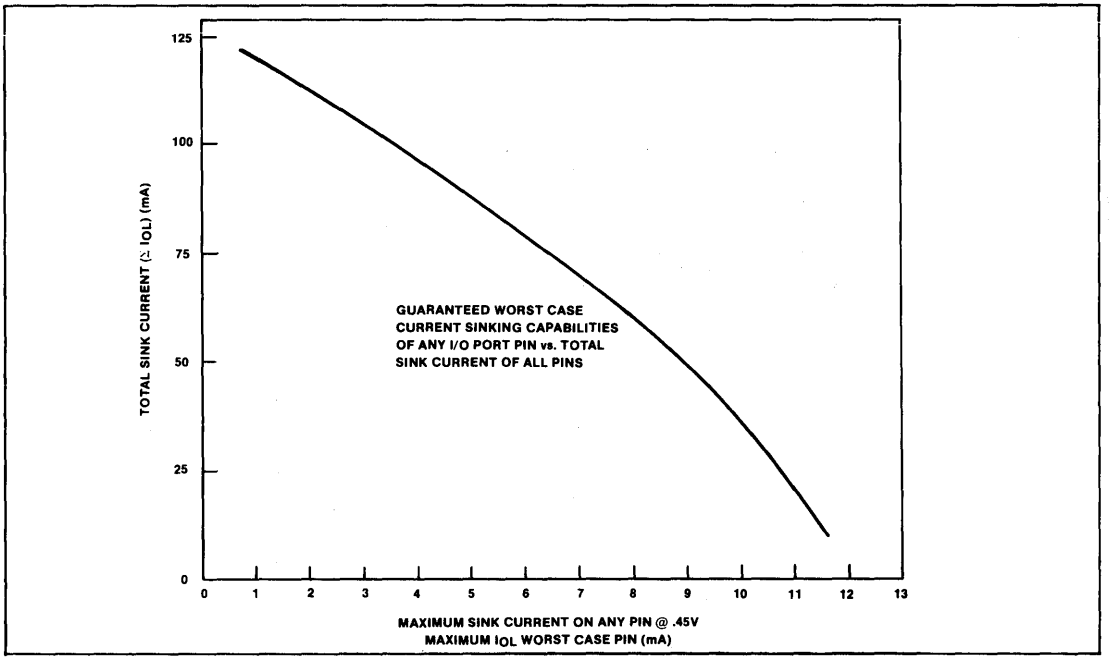


Figure 3

Sink Capability

The 8243 can sink 5 mA @ .45V on each of its 16 I/O lines simultaneously. If, however, all lines are not sinking simultaneously or all lines are not fully loaded, the drive capability of any individual line increases as is shown by the accompanying curve.

For example, if only 5 of the 16 lines are to sink current at one time, the curve shows that each of those 5 lines is capable of sinking 9 mA @ .45V (if any lines are to sink 9 mA the total I_{OL} must not exceed 45 mA or five 9 mA loads).

Example: How many pins can drive 5 TTL loads (1.6 mA) assuming remaining pins are unloaded?

I_{OL} = 5 x 1.6 mA = 8 mA
ϵI_{OL} = 60 mA from curve
pins = 60 mA ÷ 8 mA/pin = 7.5 = 7

In this case, 7 lines can sink 8 mA for a total of 56mA. This leaves 4 mA sink current capability which can be divided in any way among the remaining 8 I/O lines of the 8243.

Example: This example shows how the use of the 20 mA sink capability of Port 7 affects the sinking capability of the other I/O lines.

An 8243 will drive the following loads simultaneously.

2 loads—20 mA @ 1V (port 7 only)
8 loads—4 mA @ .45V
6 loads—3.2 mA @ .45V
Is this within the specified limits?

ϵI_{OL} = (2 x 20) + (8 x 4) + (6 x 3.2) = 91.2 mA.
From the curve: for I_{OL} = 4 mA, $\epsilon I_{OL} \approx$ 93 mA.
since 91.2 mA < 93 mA the loads are within specified limits.

Although the 20 mA @ 1V loads are used in calculating ϵI_{OL}, it is the largest current required @ .45V which determines the maximum allowable ϵI_{OL}.

NOTE: A10 to 50KΩ pullup resistor to +5V should be added to 8243 outputs when driving to 5V CMOS directly.

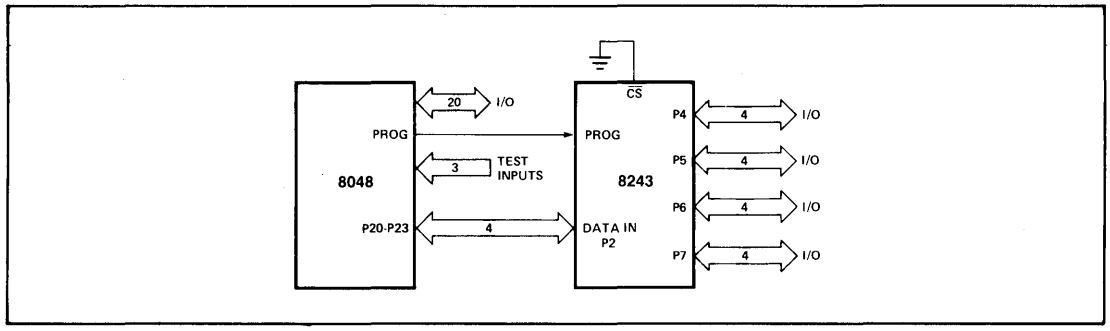


Figure 4. Expander Interface

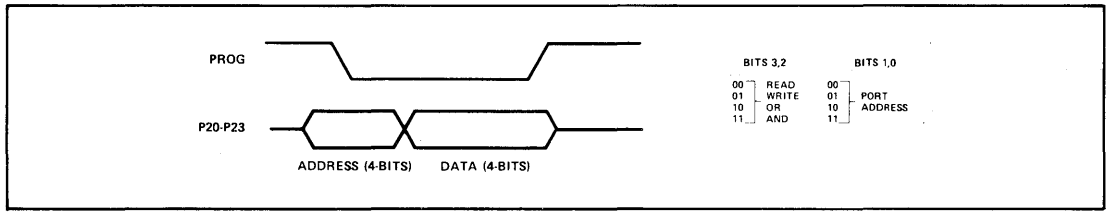


Figure 5. Output Expander Timing

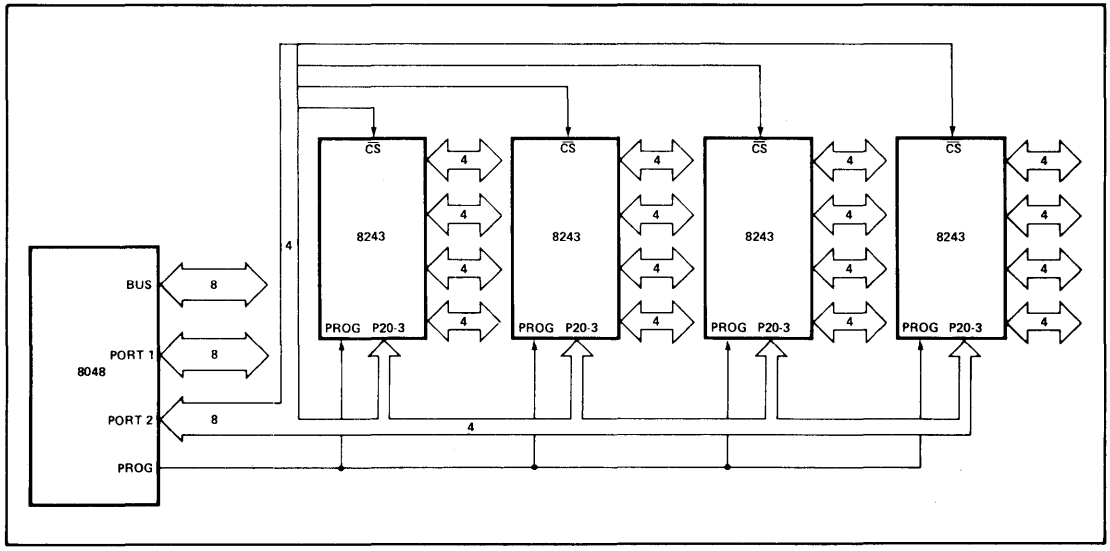


Figure 6. Using Multiple 8243's

8155/8156/8155-2/8156-2
2048 BIT STATIC MOS RAM WITH I/O PORTS AND TIMER

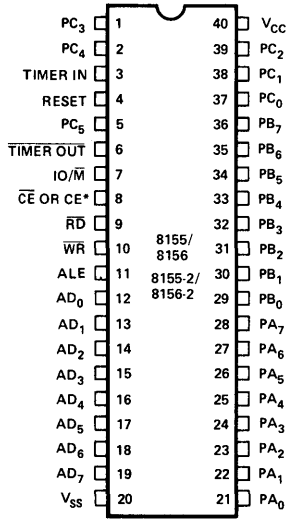
- 256 Word x 8 Bits
- Single +5V Power Supply
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8 Bit I/O Ports
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/ Timer
- Compatible with 8085A and 8088 CPU
- Multiplexed Address and Data Bus
- 40 Pin DIP

The 8155 and 8156 are RAM and I/O chips to be used in the 8085A and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085A CPU. The 8155-2 and 8156-2 have maximum access times of 330 ns for use with the 8085A-2 and the full speed 5 MHz 8088 CPU.

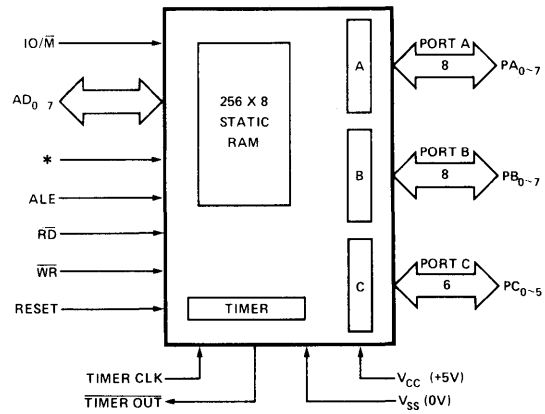
The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

PIN CONFIGURATION



BLOCK DIAGRAM



*: 8155/8155-2 = $\overline{CE}$, 8156/8156-2 = CE

8155/8156 PIN FUNCTIONS

Symbol	Function
RESET (input)	Pulse provided by the 8085A to initialize the system (connect to 8085A RESET OUT). Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulse should typically be two 8085A clock cycle times.
AD_{0-7} (input/output)	3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus. The 8-bit address is latched into the address latch inside the 8155/56 on the falling edge of ALE. The address can be either for the memory section or the I/O section depending on the IO/$\overline{M}$ input. The 8-bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal.
CE or $\overline{CE}$ (input)	Chip Enable: On the 8155, this pin is $\overline{CE}$ and is ACTIVE LOW. On the 8156, this pin is CE and is ACTIVE HIGH.
$\overline{RD}$ (input)	Read control: Input low on this line with the Chip Enable active enables and AD_{0-7} buffers. If IO/$\overline{M}$ pin is low, the RAM content will be read out to the AD bus. Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus.
$\overline{WR}$ (input)	Write control: Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register depending on IO/$\overline{M}$.

Symbol	Function
ALE (input)	Address Latch Enable: This control signal latches both the address on the AD_{0-7} lines and the state of the Chip Enable and IO/$\overline{M}$ into the chip at the falling edge of ALE.
IO/$\overline{M}$ (input)	Selects memory if low and I/O and command/status registers if high.
$PA_{0-7}(8)$ (input/output)	These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
$PB_{0-7}(8)$ (input/output)	These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
$PC_{0-5}(6)$ (input/output)	These 6 pins can function as either input port, output port, or as control signals for PA and PB. Programming is done through the command register. When PC_{0-5} are used as control signals, they will provide the following: PC_0 — A INTR (Port A Interrupt) PC_1 — $\overline{ABF}$ (Port A Buffer Full) PC_2 — $\overline{A\ STB}$ (Port A Strobe) PC_3 — B INTR (Port B Interrupt) PC_4 — $\overline{B\ BF}$ (Port B Buffer Full) PC_5 — B STB (Port B Strobe)
TIMER IN (input)	Input to the counter-timer.
$\overline{TIMER\ OUT}$ (output)	Timer output. This output can be either a square wave or a pulse depending on the timer mode.
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference.

ABSOLUTE MAXIMUM RATINGS*

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 1.5W

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

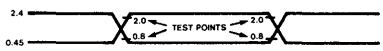
D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
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		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} to 0V
I_{LO}	Output Leakage Current		± 10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current		180	mA	
I_{IL}(CE)	Chip Enable Leakage 8155 8156		 +100 -100	 μA μA	V_{IN} = V_{CC} to 0V

A.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

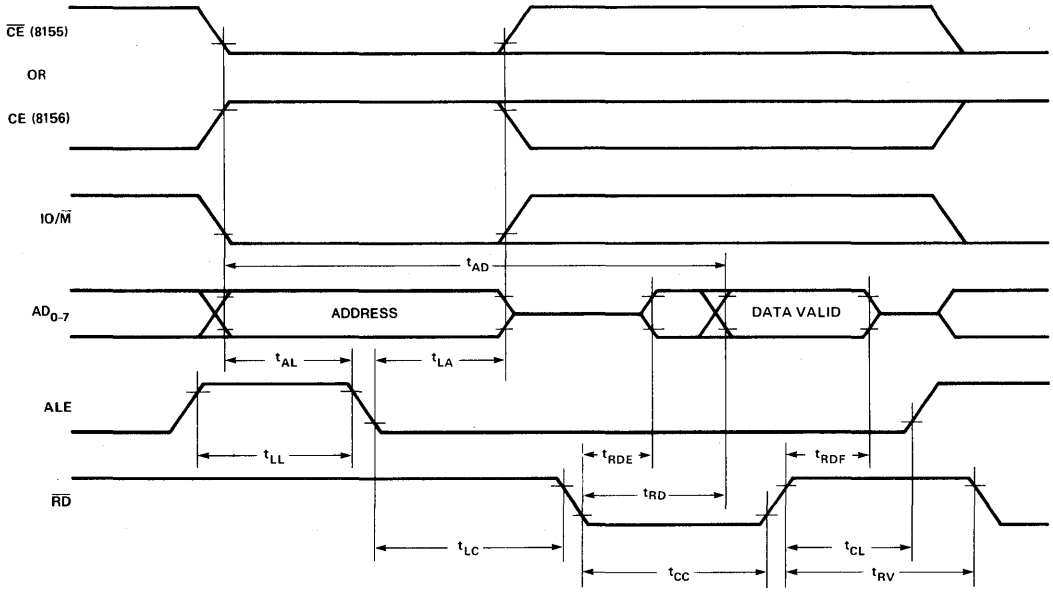
		8155/8156		8155-2/8156-2 (Preliminary)		
SYMBOL	PARAMETER	MIN.	MAX.	MIN.	MAX.	UNITS
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time after Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{AD}	Address Stable to Data Out Valid		400		330	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float After READ	0	100	0	80	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to WRITE Set Up Time	150		100		ns
t_{WD}	Data In Hold Time After WRITE	0		0		ns
t_{RV}	Recovery Time Between Controls	300		200		ns
t_{WP}	WRITE to Port Output		400		300	ns
t_{PR}	Port Input Setup Time	70		50		ns
t_{RP}	Port Input Hold Time	50		10		ns
t_{SBF}	Strobe to Buffer Full		400		300	ns
t_{SS}	Strobe Width	200		150		ns
t_{RBE}	READ to Buffer Empty		400		300	ns
t_{SI}	Strobe to INTR On		400		300	ns
t_{RDI}	READ to INTR Off		400		300	ns
t_{PSS}	Port Setup Time to Strobe Strobe	50		0		ns
t_{PHS}	Port Hold Time After Strobe	120		100		ns
t_{SBE}	Strobe to Buffer Empty		400		300	ns
t_{WBF}	WRITE to Buffer Full		400		300	ns
t_{WI}	WRITE to INTR Off		400		300	ns
t_{TL}	TIMER-IN to $\overline{\text{TIMER-OUT}}$ Low		400		300	ns
t_{TH}	TIMER-IN to $\overline{\text{TIMER-OUT}}$ High		400		300	ns
t_{RDE}	Data Bus Enable from READ Control	10		10		ns
t_1	TIMER-IN Low Time	80		40		ns
t_2	TIMER-IN High Time	120		70		ns

Input Waveform for A.C. Tests:



WAVEFORMS

a. Read Cycle



b. Write Cycle

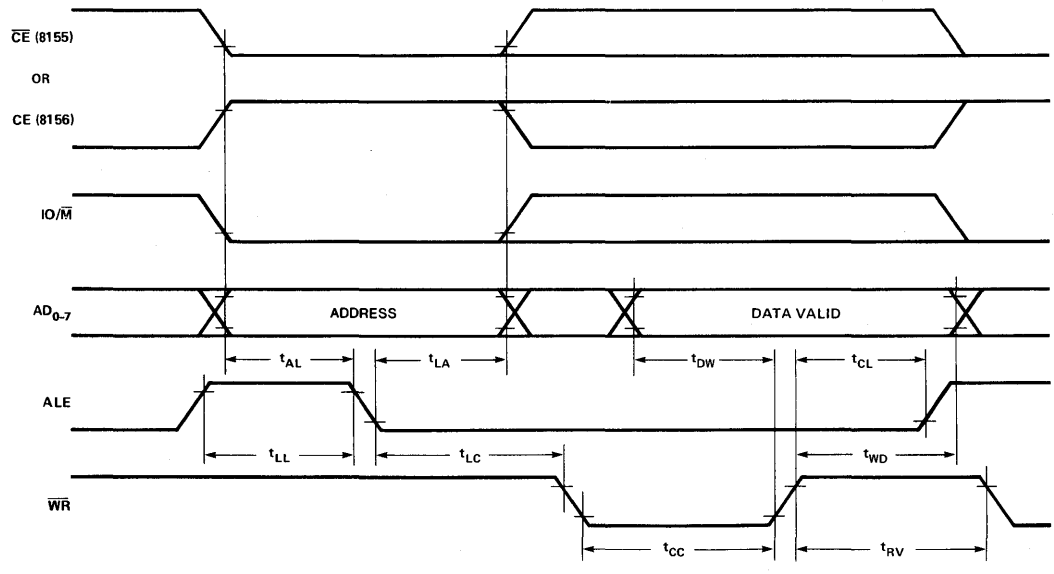
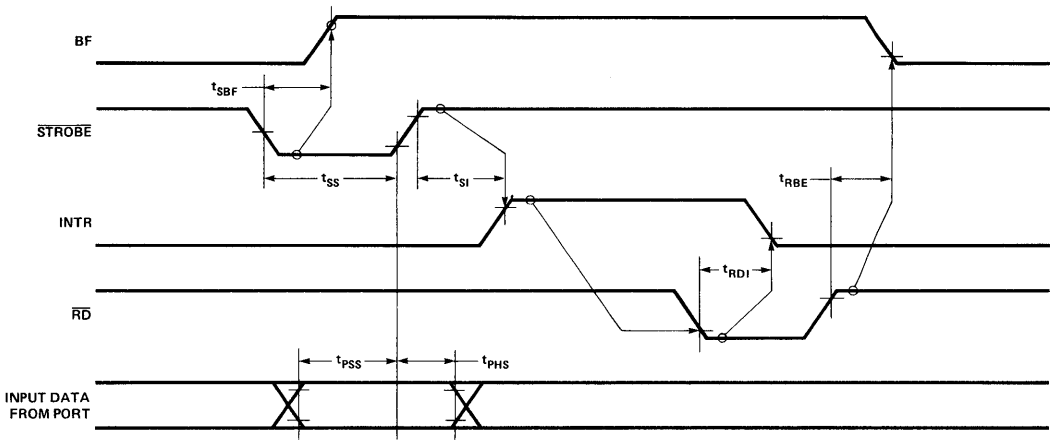


Figure 12. 8155/8156 Read/Write Timing Diagrams

a. Strobed Input Mode



b. Strobed Output Mode

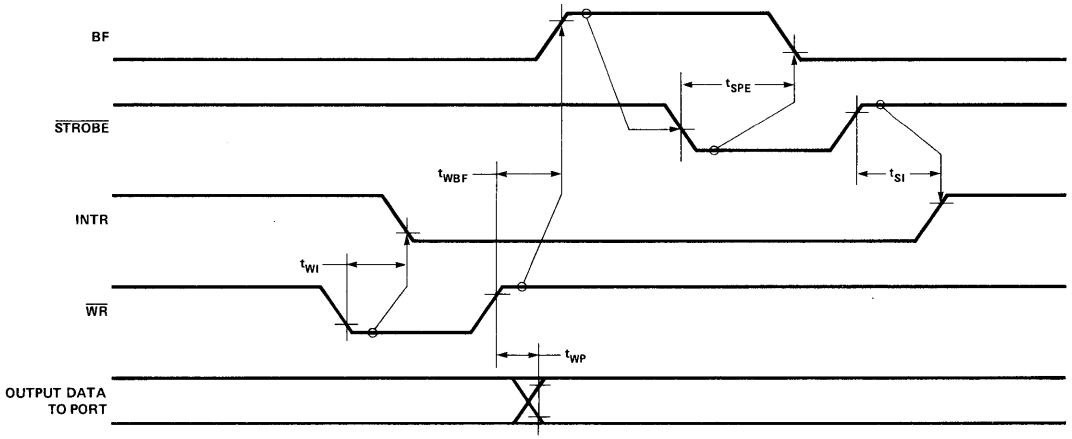
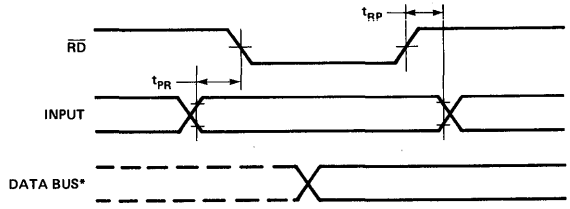
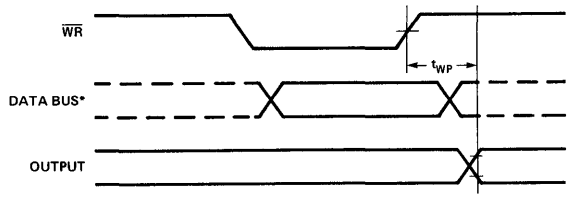


Figure 13. Strobed I/O Timing

a. Basic Input Mode

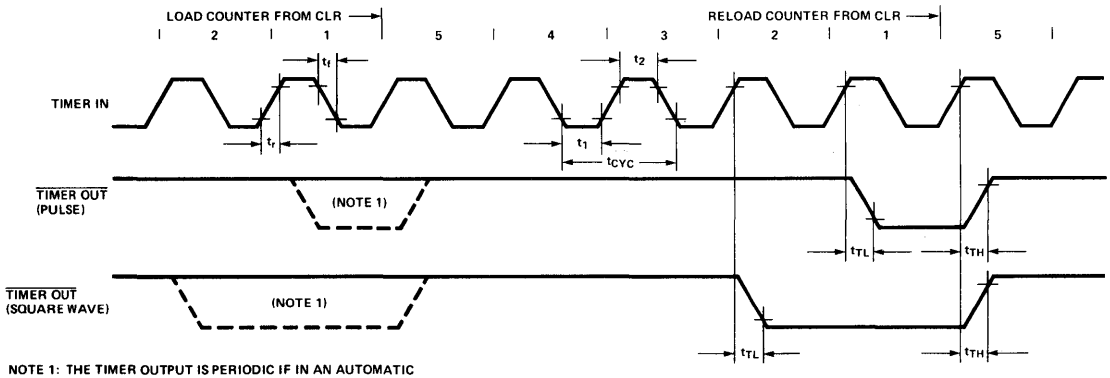


b. Basic Output Mode



*DATA BUS TIMING IS SHOWN IN FIGURE 7.

Figure 14. Basic I/O Timing Waveform



NOTE 1: THE TIMER OUTPUT IS PERIODIC IF IN AN AUTOMATIC RELOAD MODE (M_1 MODE BIT = 1)

Figure 15. Timer Output Waveform Countdown from 5 to 1

8185/8185-2
1024 x 8-BIT STATIC RAM FOR MCS-85™

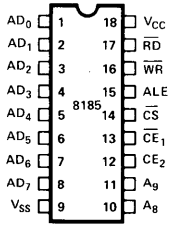
- **Multiplexed Address and Data Bus**
- **Directly Compatible with 8085A and 8088 Microprocessors**
- **Low Operating Power Dissipation**
- **Low Standby Power Dissipation**
- **Single +5V Supply**
- **High Density 18-Pin Package**

The Intel® 8185 is an 8192-bit static random access memory (RAM) organized as 1024 words by 8-bits using N-channel Silicon-Gate MOS technology. The multiplexed address and data bus allows the 8185 to interface directly to the 8085A and 8088 microprocessors to provide a maximum level of system integration.

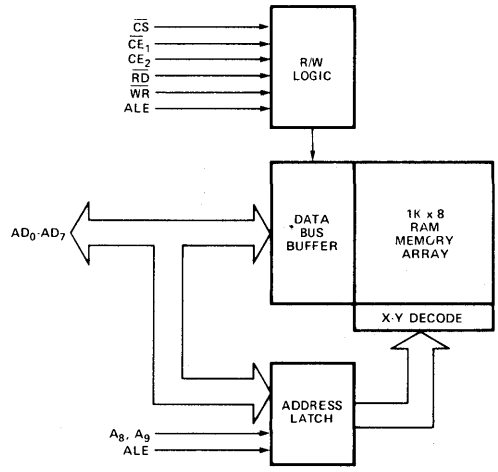
The low standby power dissipation minimizes system power requirements when the 8185 is disabled.

The 8185-2 is a high-speed selected version of the 8185 that is compatible with the 5 MHz 8085A-2 and the full speed 5 MHz 8088.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

AD_0-AD_7	ADDRESS/DATA LINES
A_8, A_9	ADDRESS LINES
$\overline{CS}$	CHIP SELECT
$\overline{CE}_1$	CHIP ENABLE (IO/$\overline{M}$)
CE_2	CHIP ENABLE
ALE	ADDRESS LATCH ENABLE
$\overline{RD}$	READ ENABLE
$\overline{WR}$	WRITE ENABLE

OPERATIONAL DESCRIPTION

The 8185 has been designed to provide for direct interface to the multiplexed bus structure and bus timing of the 8085A microprocessor.

At the beginning of an 8185 memory access cycle, the 8-bit address on AD_{0-7}, A_8 and A_9, and the status of $\overline{CE}_1$ and CE_2 are all latched internally in the 8185 by the falling edge of ALE. If the latched status of both $\overline{CE}_1$ and CE_2 are active, the 8185 powers itself up, but no action occurs until the $\overline{CS}$ line goes low and the appropriate $\overline{RD}$ or $\overline{WR}$ control signal input is activated.

The $\overline{CS}$ input is not latched by the 8185 in order to allow the maximum amount of time for address decoding in selecting the 8185 chip. Maximum power consumption savings will occur, however, only when $\overline{CE}_1$ and CE_2 are activated selectively to power down the 8185 when it is not in use. A possible connection would be to wire the 8085A's IO/$\overline{M}$ line to the 8185's $\overline{CE}_1$ input, thereby keeping the 8185 powered down during I/O and interrupt cycles.

TABLE 1.
TRUTH TABLE FOR
POWER DOWN AND FUNCTION ENABLE

$\overline{CE}_1$	CE_2	$\overline{CS}$	(CS*)[2]	8185 Status
1	X	X	0	Power Down and Function Disable[1]
X	0	X	0	Power Down and Function Disable[1]
0	1	1	0	Powered Up and Function Disable[1]
0	1	0	1	Powered Up and Enabled

Notes:
X: Don't Care.
1: Function Disable implies Data Bus in high impedance state and not writing.
2: CS* = ($\overline{CE}_1$ = 0) • (CE_2 = 1) • ($\overline{CS}$ = 0)
CS* = 1 signifies all chip enables and chip select active

TABLE 2.
TRUTH TABLE FOR
CONTROL AND DATA BUS PIN STATUS

(CS*)	$\overline{RD}$	$\overline{WR}$	AD_{0-7} During Data Portion of Cycle	8185 Function
0	X	X	Hi-Impedance	No Function
1	0	1	Data from Memory	Read
1	1	0	Data to Memory	Write
1	1	1	Hi-Impedance	Reading, but not Driving Data Bus

Note:
X: Don't Care.

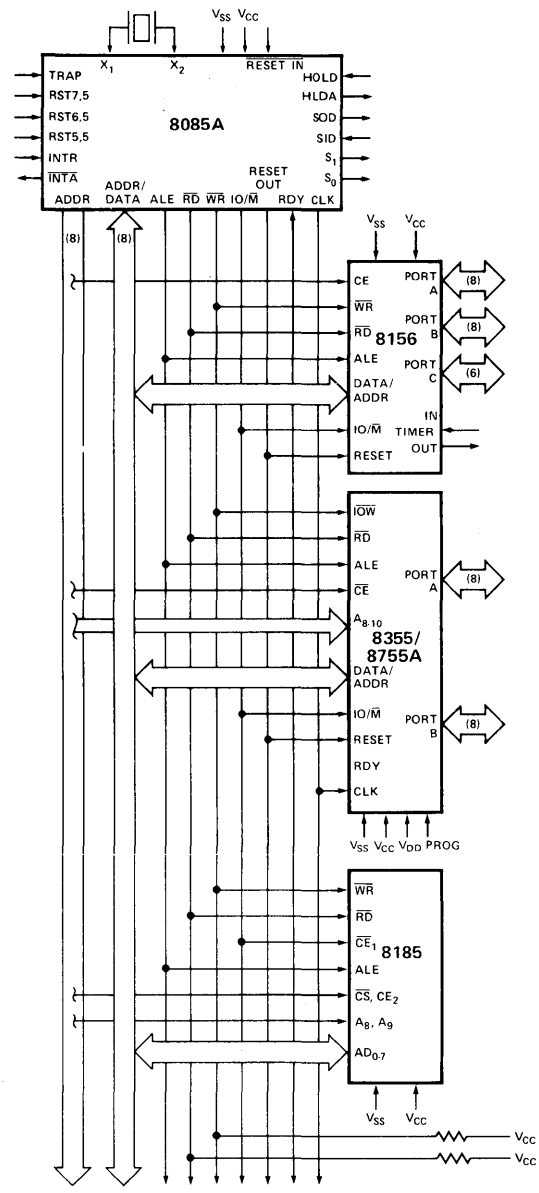


Figure 1. 8185 in an MCS-85 System.

4 Chips:
2K Bytes ROM
1.25K Bytes RAM
38 I/O Lines
1 Counter/Timer
2 Serial I/O Lines
5 Interrupt Inputs

ABSOLUTE MAXIMUM RATINGS*

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 1.5W

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

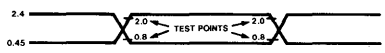
Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4			I_{OH} = – 400μA
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} to 0V
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current Powered Up		100	mA	
	Powered Down		35	mA	

A.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

Symbol	Parameter [1]	8185 Preliminary		8185-2 Preliminary		Units
		Min.	Max.	Min.	Max.	
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time After Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{LD}	ALE to Data Out Valid		300		200	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float After READ	0	100	0	80	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to WRITE Set Up Time	150		150		ns
t_{WD}	Data In Hold Time After WRITE	20		20		ns
t_{SC}	Chip Select Set Up to Control Line	10		10		ns
t_{CS}	Chip Select Hold Time After Control	10		10		ns
t_{ALCE}	Chip Enable Set Up to ALE Falling	30		10		ns
t_{LACE}	Chip Enable Hold Time After ALE	50		30		ns

Notes:
1. All AC parameters are referenced at
   a) 2.4V and .45V for inputs
   b) 2.0V and .8V for outputs.

Input Waveform for A.C. Tests:



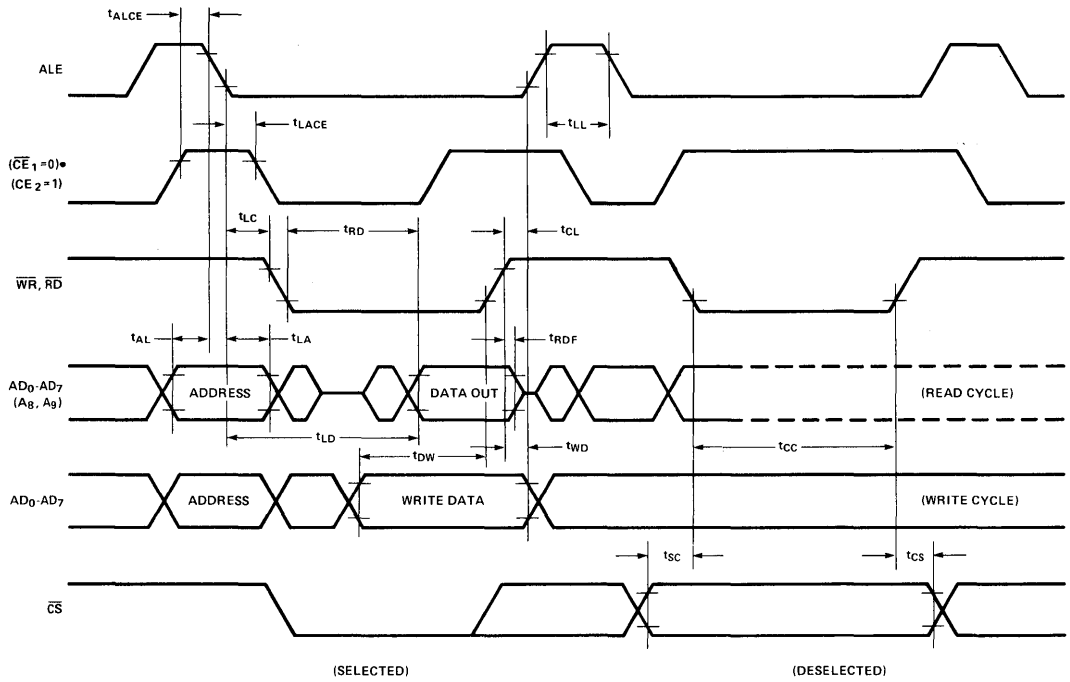


Figure 3. 8185 Timing.

8355/8355-2
16,384-BIT ROM WITH I/O

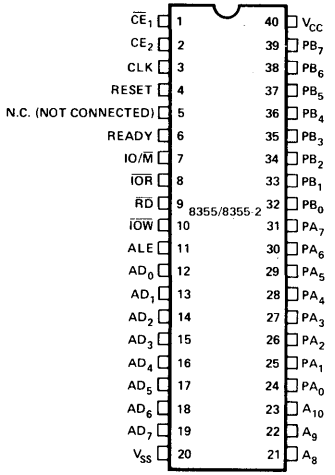
- 2048 Words x 8 Bits
- Single +5V Power Supply
- Directly compatible with 8085A and 8088 Microprocessors
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- Internal Address Latch
- 40-Pin DIP

The Intel® 8355 is a ROM and I/O chip to be used in the 8085A and 8088 microprocessor systems. The ROM portion is organized as 2048 words by 8 bits. It has a maximum acess time of 400 ns to permit use with no wait states in the 8085A CPU.

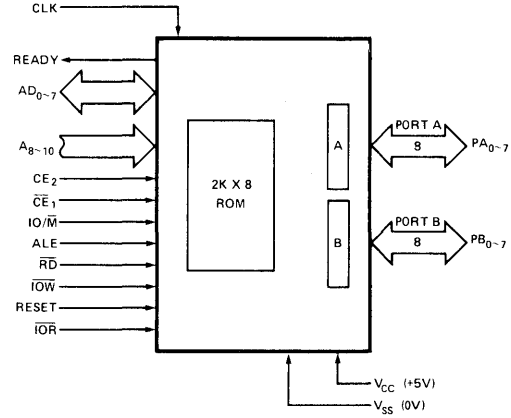
The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines and each I/O port line is individually programmable as input or output.

The 8355-2 has a 300ns access time for compatibility with the 8085A-2 and full speed 5 MHz 8088 microprocessors.

PIN CONFIGURATION



BLOCK DIAGRAM



Symbol	Function
ALE (Input)	When ALE (Address Latch Enable is high, AD_{0-7}, $IO/\overline{M}$, A_{8-10}, CE, and $\overline{CE}$ enter address latched. The signals (AD, $IO/\overline{M}$, A_{8-10}, CE, $\overline{CE}$) are latched in at the trailing edge of ALE.
AD_{0-7} (Input)	Bidirectional Address/Data bus. The lower 8-bits of the ROM or I/O address are applied to the bus lines when ALE is high. During an I/O cycle, Port A or B are selected based on the latched value of AD_0. If $\overline{RD}$ or $\overline{IOR}$ is low when the latched chip enables are active, the output buffers present data on the bus.
A_{8-10} (Input)	These are the high order bits of the ROM address. They do not affect I/O operations.
$\overline{CE_1}$ CE_2 (Input)	Chip Enable Inputs: $\overline{CE_1}$ is active low and CE_2 is active high. The 8355 can be accessed only when BOTH Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD_{0-7} and READY outputs will be in a high impedance state.
$IO/\overline{M}$ (Input)	If the latched $IO/\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the ROM.
$\overline{RD}$ (Input)	If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD_{0-7} output buffers are enabled and output either the selected ROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD_{0-7} output buffers are 3-state.
$\overline{IOW}$ (Input)	If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD_0 to be written with the data on AD_{0-7}. The state of $IO/\overline{M}$ is ignored.

Symbol	Function
CLK (Input)	The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE}$ low, CE high and ALE high.
READY (Output)	Ready is a 3-state output controlled by $\overline{CE_1}$, CE_2, ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK (see Figure 6).
PA_{0-7} (Input/ Output)	These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD_0. Read operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD_0 low, or $IO/\overline{M}$ high, $\overline{RD}$ low, active chip enables, and AD_0 low.
PB_{0-7} (Input/ Output)	This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD_0.
RESET (Input)	An input high on RESET causes all pins in Port A and B to assume input mode.
$\overline{IOR}$ (Input)	When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination $IO/\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to V_{CC} ("1").
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference.

ABSOLUTE MAXIMUM RATINGS*

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 1.5W

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

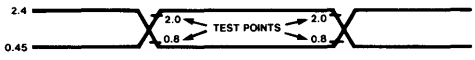
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	V_{CC} = 5.0V
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	V_{CC} = 5.0V
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{IL}	Input Leakage		10	μA	V_{IN} = V_{CC} to 0V
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤V_{OUT} ≤V_{CC}
I_{CC}	V_{CC} Supply Current		180	mA	

A.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

		8355		8355-2		
Symbol	**Parameter**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{CYC}	Clock Cycle Time	320		200		ns
T_1	CLK Pulse Width	80		40		ns
T_2	CLK Pulse Width	120		70		ns
t_f,t_r	CLK Rise and Fall Time		30		30	ns
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time after Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{AD}	Address Stable to Data Out Valid		400		330	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float after READ	0	100	0	85	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to Write Set Up Time	150		150		ns
t_{WD}	Data In Hold Time After WRITE	10		10		ns
t_{WP}	WRITE to Port Output		400		400	ns
t_{PR}	Port Input Set Up Time	50		50		ns
t_{RP}	Port Input Hold Time	50		50		ns
t_{RYH}	READY HOLD Time	0	160	0	160	ns
t_{ARY}	ADDRESS (CE) to READY		160		160	ns
t_{RV}	Recovery Time Between Controls	300		200		ns
t_{RDE}	READ Control to Data Bus Enable	10		10		ns

Note: C_{LOAD} = 150pF

Input Waveform for A.C. Tests:



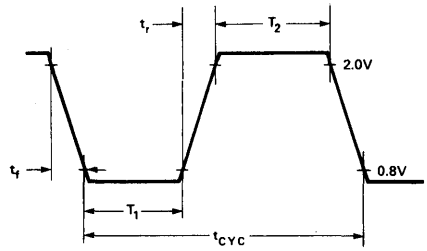


Figure 3. Clock Specification for 8355

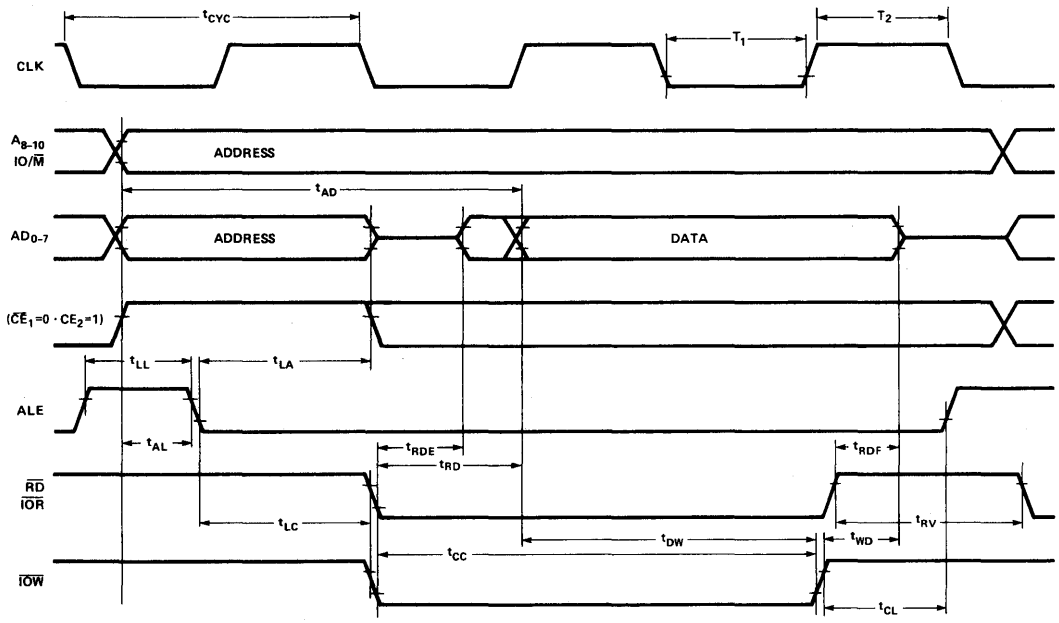
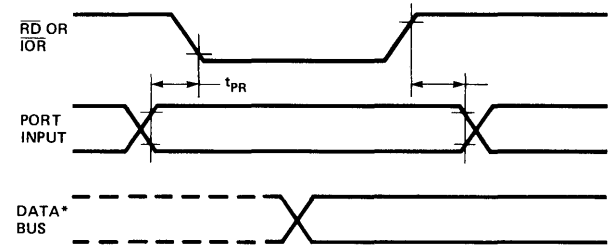
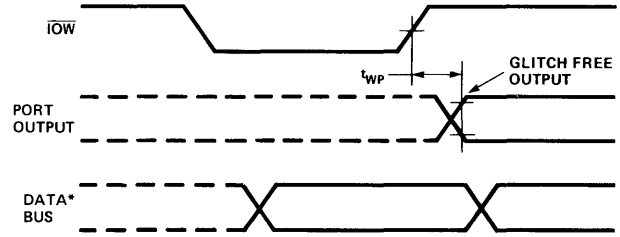


Figure 4. ROM Read and I/O Read and Write

a. Input Mode



b. Output Mode



*DATA BUS TIMING IS SHOWN IN FIGURE 4.

Figure 5. I/O Port Timing

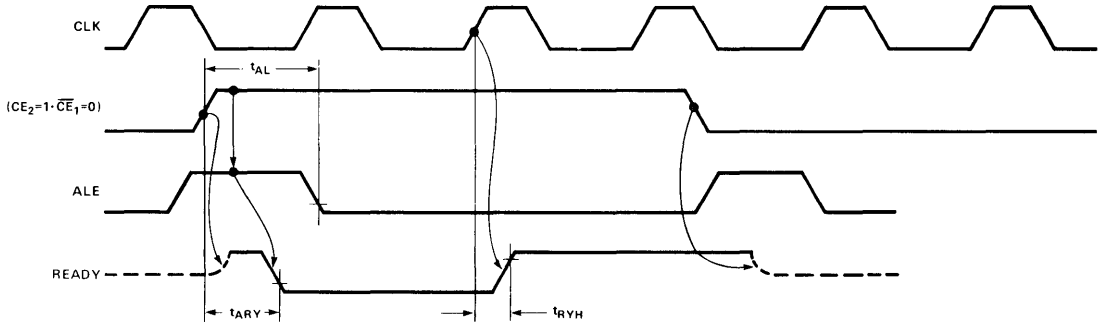


Figure 6. Wait State Timing (Ready = 0)

8755A/8755A-2
16,384-BIT EPROM WITH I/O

- 2048 Words x 8 Bits
- Single +5V Power Supply (V_{CC})
- Directly Compatible with 8085A and 8088 Microprocessors
- U.V. Erasable and Electrically Reprogrammable
- Internal Address Latch
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- 40-Pin DIP

The Intel® 8755A is an erasable and electrically reprogrammable ROM (EPROM) and I/O chip to be used in the 8085A and 8088 microprocessor systems. The EPROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 450 ns to permit use with no wait states in an 8085A CPU.

The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is individually programmable as input or output.

The 8755A-2 is a high speed selected version of the 8755A compatible with the 5 MHz 8085A-2 and the full speed 5 MHz 8088.

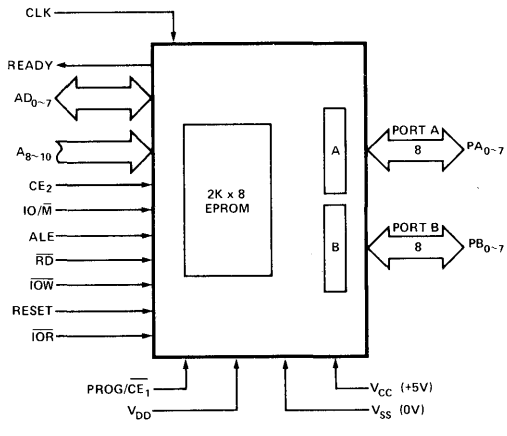


Figure 1. Block Diagram

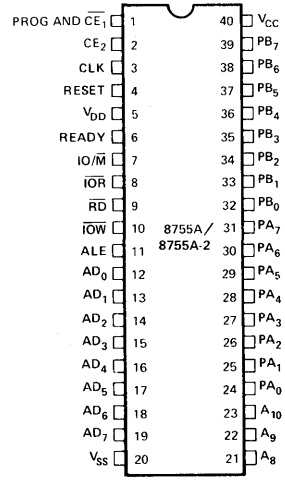


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Type	Name and Function
ALE	I	**Address Latch Enable:** When Address Latch Enable goes *high*, AD_{0-7}, IO/M, A_{8-10}, CE_2, and $\overline{CE_1}$ enter the address latches. The signals (AD, IO/M, A_{8-10}, CE) are latched in at the trailing edge of ALE.
AD_{0-7}	I	**Bidirectional Address/Data Bus:** The lower 8-bits of the PROM or I/O address are applied to the bus lines when ALE is high. During an I/O cycle, Port A or B are selected based on the latched value of AD_0. IF $\overline{RD}$ or $\overline{IOR}$ is low when the latched Chip Enables are active, the output buffers present data on the bus.
A_{8-10}	I	**Address:** These are the high order bits of the PROM address. They do not affect I/O operations.
PROG/$\overline{CE_1}$ CE_2	I	**Chip Enable Inputs:** $\overline{CE_1}$ is active low and CE_2 is active high. The 8755A can be accessed only when *both* Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD_{0-7} and READY outputs will be in a high impedance state. $\overline{CE_1}$ is also used as a programming pin. (See section on programming.)
IO/$\overline{M}$	I	**I/O Memory:** If the latched IO/$\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the PROM.
$\overline{RD}$	I	**Read:** If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD_{0-7} output buffers are enabled and output either the selected PROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD_{0-7} output buffers are 3-stated.
$\overline{IOW}$	I	**I/O Write:** If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD_0 to be written with the data on AD_{0-7}. The state of IO/$\overline{M}$ is ignored.
CLK	I	**Clock:** The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE_1}$ low, CE_2 high, and ALE high.

Symbol	Type	Name and Function
READY	O	**Ready** is a 3-state output controlled by CE_2, $\overline{CE_1}$, ALE and CLK. READY is forced low when the Chip Enables are actove during the time ALE is high, and remains low until the rising edge of the next CLK. (See Figure 6.)
PA_{0-7}	I/O	**Port A:** These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD_0, AD_1. Read Operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD_0 and AD_1 low, *or* IO/$\overline{M}$ high, $\overline{RD}$ low, active Chip Enables, and AD_0 and AD_1 low.
PB_{0-7}	I/O	**Port B:** This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD_0 and a 0 from AD_1.
RESET	I	**Reset:** In normal operation, an input high on RESET causes all pins in Ports A and B to assume input mode (clear DDR register).
$\overline{IOR}$	I	**I/O Read:** When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination of IO/$\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to V_{CC} ("1").
V_{CC}		**Power:** +5 volt supply.
V_{SS}		**Ground:** Reference.
V_{DD}		**Power Supply:** V_{DD} is a programming voltage, and must be tied to V_{CC} when the 8755A is being read. For programming, a high voltage is supplied with V_{DD} = 25V, typical. (See section on programming.)

FUNCTIONAL DESCRIPTION

PROM Section

The 8755A contains an 8-bit address latch which allows it to interface directly to MCS-48, MCS-85 and iAPX 88/10 Microcomputers without additional hardware.

The PROM section of the chip is addressed by the 11-bit address and CE. The address, $\overline{CE}_1$ and CE_2 are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/$\overline{M}$ is low when $\overline{RD}$ goes low, the contents of the PROM location addressed by the latched address are put out on the AD_{0-7} lines (provided that V_{DD} is tied to V_{CC}.)

I/O Section

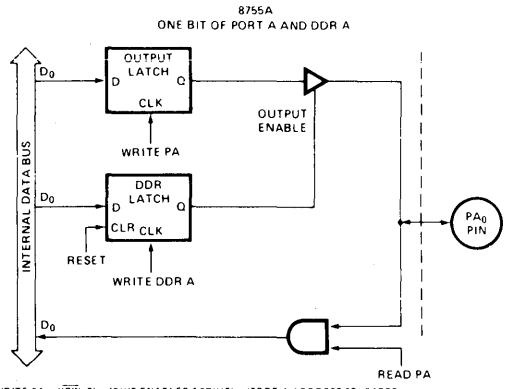
The I/O section of the chip is addressed by the latched value of AD_{0-1}. Two 8-bit Data Direction Registers DDR in 8755A determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8755A are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

AD_1	AD_0	Selection
0	0	Port A
0	1	Port B
1	0	Port A Data Direction Register (DDR A)
1	1	Port B Data Direction Register (DDR B)

When $\overline{IOW}$ goes low and the Chip Enables are active, the data on the AD is written into I/O port selected by the latched value of AD_{0-1}. During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of IO/$\overline{M}$. The actual output level does not change until $\overline{IOW}$ returns high. (glitch free output)

A port can be read out when the latched Chip Enables are active and either $\overline{RD}$ goes low with IO/$\overline{M}$ high, or $\overline{IOR}$ goes low. Both input and output mode bits of a selected port will appear on lines AD_{0-7}.

To clarify the function of the I/O Ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.



WRITE PA = ($\overline{IOW}$=0) • (CHIP ENABLES ACTIVE) • (PORT A ADDRESS SELECTED)
WRITE DDR A = ($\overline{IOW}$=0) • (CHIP ENABLES ACTIVE) • (DDR A ADDRESS SELECTED)
READ PA = {[(IO/$\overline{M}$=1) • ($\overline{RD}$=0)] + ($\overline{IOR}$=0)} • (CHIP ENABLES ACTIVE) • (PORT A ADDRESS SELECTED)

NOTE: WRITE PA IS NOT QUALIFIED BY IO/M.

Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the Output Latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.

TABLE 1. 8755A PROGRAMMING MODULE CROSS REFERENCE

MODULE NAME	USE WITH
UPP 955	UPP(4)
UPP UP2(2)	UPP 855
PROMPT 975	PROMPT 80/85(3)
PROMPT 475	PROMPT 48(1)

NOTES:
1. Described on p. 13-34 of 1978 Data Catalog.
2. Special adaptor socket.
3. Described on p. 13-39 of 1978 Data Catalog.
4. Described on p. 13-71 of 1978 Data Catalog.

ERASURE CHARACTERISTICS

The erasure characteristics of the 8755A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8755A in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8755A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8755 window to prevent unintentional erasure.

The recommended erasure procedure for the 8755A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15W-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000μW/cm^2 power rating. The 8755A should be placed within one inch from the lamp tubes during erasure. Some lamps have a filter on their tubes and this filter should be removed before erasure.

PROGRAMMING

Initially, and after each erasure, all bits of the EPROM portions of the 8755A are in the "1" state. Information is introduced by selectively programming "0" into the desired bit locations. A programmed "0" can only be changed to a "1" by UV erasure.

The 8755A can be programmed on the Intel® Universal PROM Programmer (UPP), and the PROMPT™ 80/85 and PROMPT-48™ design aids. The appropriate programming modules and adapters for use in programming both 8755A's and 8755's are shown in Table 1.

The program mode itself consists of programming a single address at a time, giving a single 50 msec pulse for every address. Generally, it is desirable to have a verify cycle after a program cycle for the same address as shown in the attached timing diagram. In the verify cycle (i.e., normal memory read cycle) 'V_{DD}' should be at +5V.

Preliminary timing diagrams and parameter values pertaining to the 8755A programming operation are contained in Figure 7.

SYSTEM APPLICATIONS

System Interface with 8085A and 8088

A system using the 8755A can use either one of the two I/O Interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both CE_3 and $\overline{CE}_1$. By using a combination of unused address lines A_{11-15} and the Chip Enable inputs, the 8085A system can use up to 5 each 8755A's without requiring a CE decoder. See Figure 2a and 2b.

If a memory mapped I/O approach is used the 8755A will be selected by the combination of both the Chip Enables and IO/$\overline{M}$ using the AD_{8-15} address lines. See Figure 1.

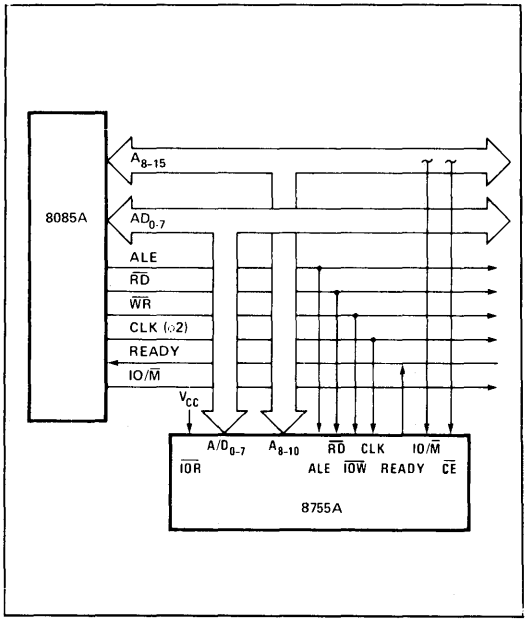


Figure 3. 8755A in 8085A System (Memory-Mapped I/O)

ABSOLUTE MAXIMUM RATINGS*

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 1.5W

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

(T_A = 0°C to 70°, V_{CC} = V_{DD} = 5V ± 5%; V_{CC} = V_{DD} = 5V ±10% for 8755A-2)

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	V_{CC} = 5.0V
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	V_{CC} = 5.0V
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{IL}	Input Leakage		10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$V_{SS} \leq 0.45V \leq V_{OUT} \leq V_{CC}$
I_{CC}	V_{CC} Supply Current		180	mA	
I_{DD}	V_{DD} Supply Current		30	mA	V_{DD} = V_{CC}
C_{IN}	Capacitance of Input Buffer		10	pF	f_C = 1μHz
$C_{I/O}$	Capacitance of I/O Buffer		15	pF	f_C = 1μHz

D.C. CHARACTERISTICS — PROGRAMMING

(T_A = 0°C to 70°, V_{CC} = 5V ± 5%, V_{SS} = 0V, V_{DD} = 25V ±1V; V_{CC} = V_{DD} = 5V ±10% for 8755A-2)

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{DD}	Programming Voltage (during Write to EPROM)	24	25	26	V
I_{DD}	Prog Supply Current		15	30	mA

A.C. CHARACTERISTICS

(T_A = 0°C to 70°, V_{CC} = 5V ± 5%; V_{CC} = V_{DD} = 5V ±10% for 8755A-2)

Symbol	Parameter	8755A		8755A-2 (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{CYC}	Clock Cycle Time	320		200		ns
T_1	CLK Pulse Width	80		40		ns
T_2	CLK Pulse Width	120		70		ns
t_f,t_r	CLK Rise and Fall Time		30		30	ns
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time after Latch	80		45		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170*		140*	ns
t_{AD}	Address Stable to Data Out Valid		450		330	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float after READ	0	100	0	85	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to Write Set Up Time	150		150		ns
t_{WD}	Data In Hold Time After WRITE	30		10		ns
t_{WP}	WRITE to Port Output		400		300	ns
t_{PR}	Port Input Set Up Time	50		50		ns
t_{RP}	Port Input Hold Time to Control	50		50		ns
t_{RYH}	READY HOLD Time to Control	0	160	0	160	ns
t_{ARY}	ADDRESS (CE) to READY		160		160	ns
t_{RV}	Recovery Time Between Controls	300		200		ns
t_{RDE}	READ Control to Data Bus Enable	10		10		ns
t_{LD}	ALE to Data Out Valid		350		270	ns

NOTE:

C_{LOAD} = 150pF.

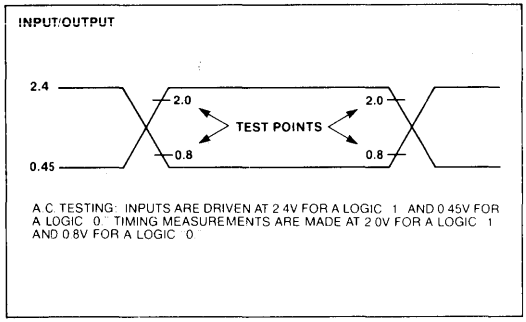
*Or $T_{AD} - (T_{AL} + T_{LC})$, whichever is greater.

A.C. CHARACTERISTICS — PROGRAMMING

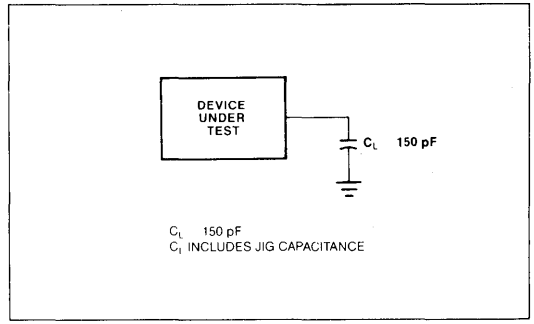
(T_A = 0°C to 70°, V_{CC} = 5V ± 5%, V_{SS} = 0V, V_{DD} = 25V ±1V; V_{CC} = V_{DD} = 5V ±10% for 8755A-2)

Symbol	Parameter	Min.	Typ.	Max.	Unit
t_{PS}	Data Setup Time	10			ns
t_{PD}	Data Hold Time	0			ns
t_S	Prog Pulse Setup Time	2			μs
t_H	Prog Pulse Hold Time	2			μs
t_{PR}	Prog Pulse Rise Time	0.01	2		μs
t_{PF}	Prog Pulse Fall Time	0.01	2		μs
t_{PRG}	Prog Pulse Width	45	50		msec

A.C. TESTING INPUT, OUTPUT WAVEFORM

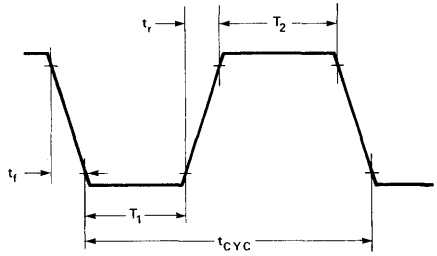


A.C. TESTING LOAD CIRCUIT

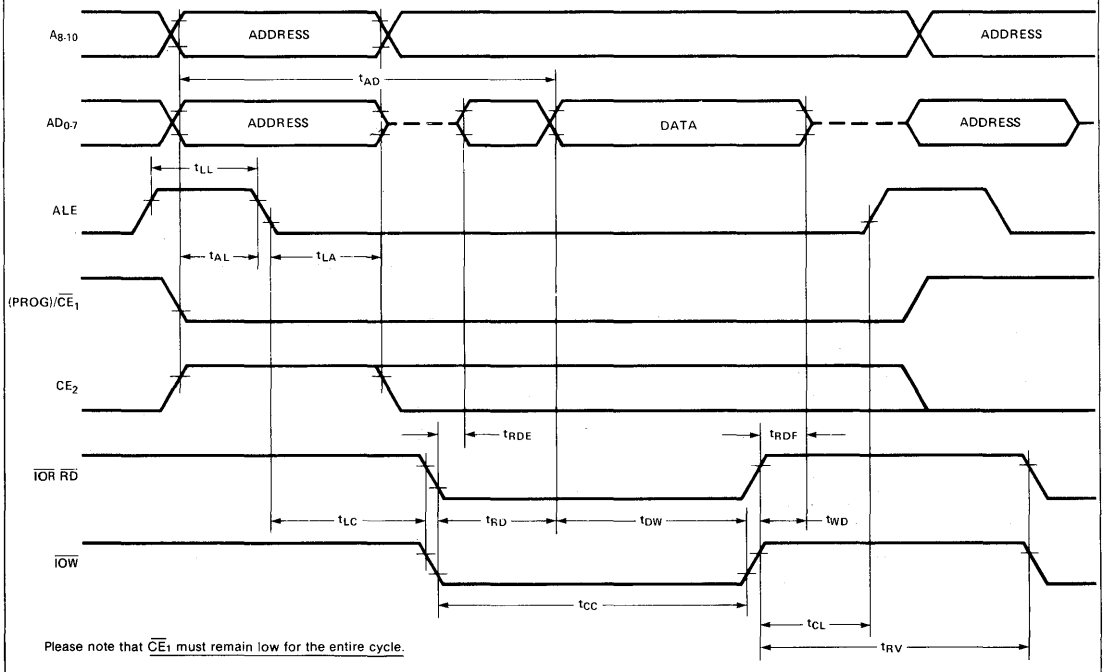


WAVEFORMS

CLOCK SPECIFICATION FOR 8755A



PROM READ, I/O READ AND WRITE

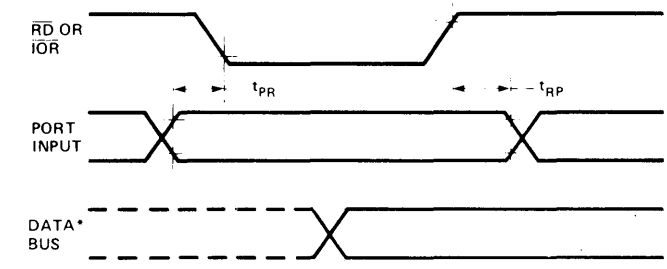


Please note that $\overline{CE_1}$ must remain low for the entire cycle.

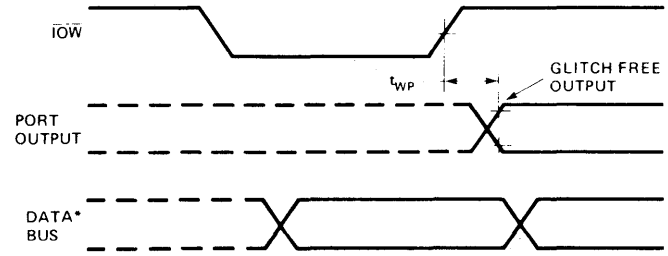
WAVEFORMS (Continued)

I/O PORT

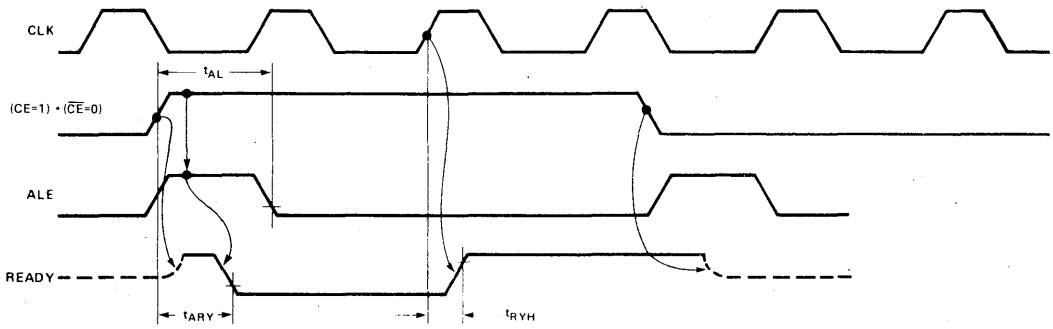
A. INPUT MODE



B. OUTPUT MODE

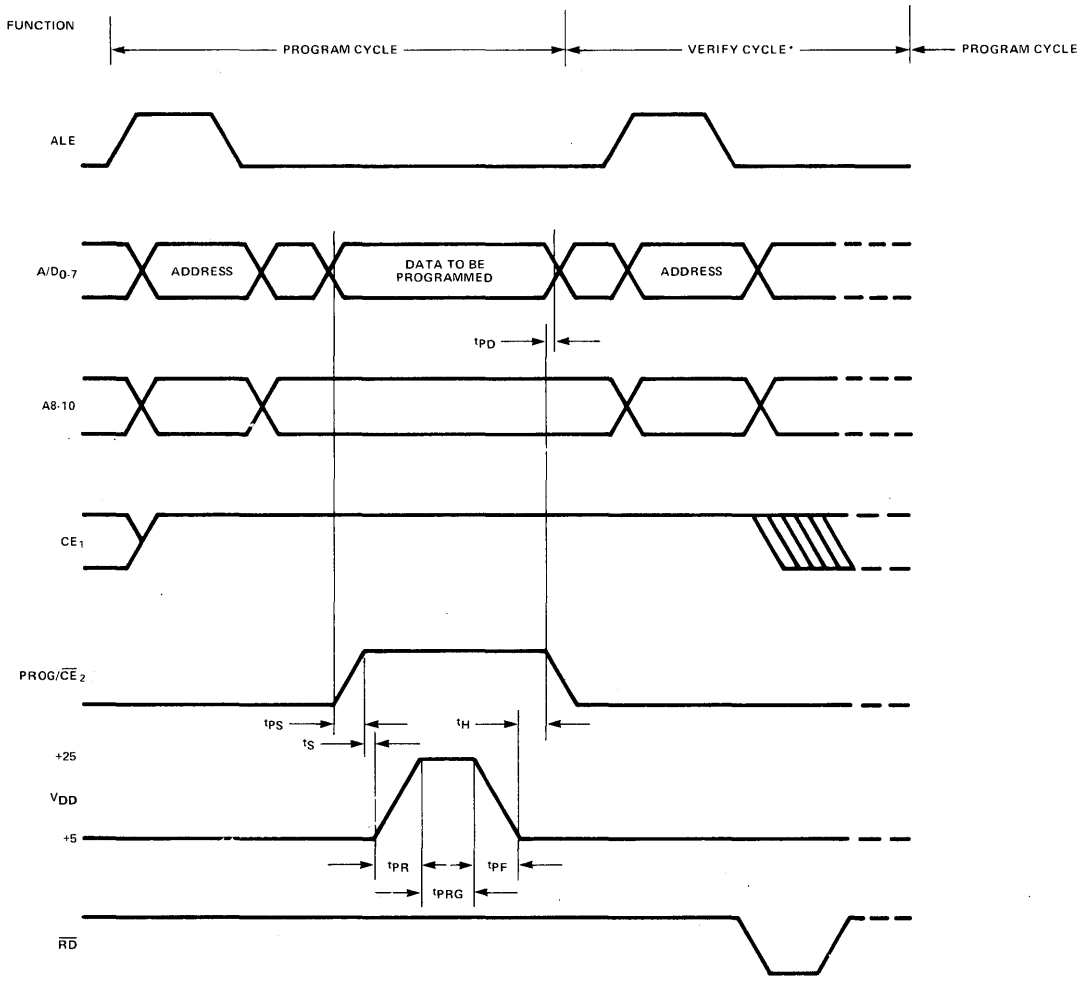


WAIT STATE (READY = 0)



WAVEFORMS (Continued)

8755A PROGRAM MODE



*VERIFY CYCLE IS A REGULAR MEMORY READ CYCLE (WITH V_{DD} = +5V FOR 8755A)

8041A/8641A/8741A
UNIVERSAL PERIPHERAL INTERFACE
8-BIT MICROCOMPUTER

- **8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package**
- **One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface**
- **DMA, Interrupt, or Polled Operation Supported**
- **1024 x 8 ROM/EPROM, 64 x 8 RAM, 8-Bit Timer/Counter, 18 Programmable I/O Pins**
- **Fully Compatible with MCS-48™, MCS-80™, MCS-85™, and MCS-86™ Microprocessor Families**
- **Interchangeable ROM and EPROM Versions**
- **3.6 MHz 8741A-8 Available**
- **Expandable I/O**
- **RAM Power-Down Capability**
- **Over 90 Instructions: 70% Single Byte**
- **Single 5V Supply**

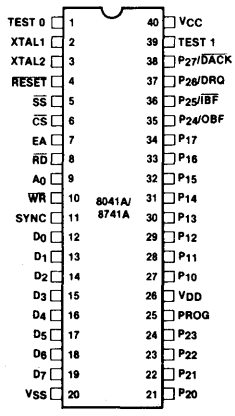
The Intel® 8041A/8741A is a general purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-48™, MCS-80™, MCS-85™, MCS-86™, and other 8-bit systems.

The UPI-41A™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041A version or as UV-erasable EPROM in the 8741A version. The 8741A and the 8041A are fully pin compatible for easy transition from prototype to production level designs. The 8641A is a one-time programmable (at the factory) 8741A which can be ordered as the first 25 pieces of a new 8041A order. The substitution of 8641A's for 8041A's allows for very fast turnaround for initial code verification and evaluation results.

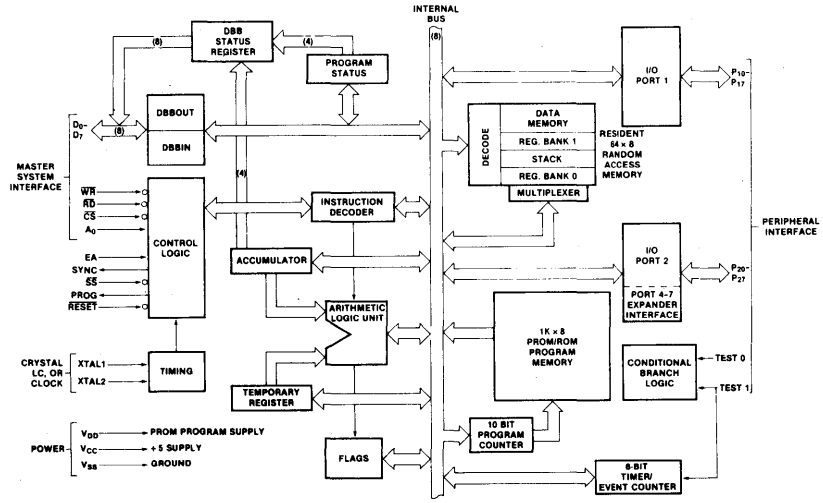
The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041A), single-step mode for debug (in the 8741A), and dual working register banks.

Because it's a complete microcomputer, the UPI provides more flexibility for the designer than conventional LSI interface devices. It is designed to be an efficient controller as well as an arithmetic processor. Applications include keyboard scanning, printer control, display multiplexing and similar functions which involve interfacing peripheral devices to microprocessor systems.

PIN CONFIGURATION



BLOCK DIAGRAM



00188A

PIN DESCRIPTION

Signal	Description
$D_0 - D_7$ (BUS)	Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-41A to an 8-bit master system data bus.
P_{10}-P_{17}	8-bit, PORT 1 quasi-bidirectional I/O lines.
P_{20}-P_{27}	8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P_{20}-P_{23}) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P_{24}-P_{27}) can be programmed to provide Interrupt Request and DMA Handshake capability. Software control can configure P_{24} as OBF (Output Buffer Full), P_{25} as $\overline{IBF}$ (Input Buffer Full), P_{26} as DRQ (DMA Request), and P_{27} as $\overline{DACK}$ (DMA ACKnowledge).
$\overline{WR}$	I/O write input which enables the master CPU to write data and command words to the UPI-41A INPUT DATA BUS BUFFER.
$\overline{RD}$	I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
$\overline{CS}$	Chip select input used to select one UPI-41A out of several connected to a common data bus.
A_0	Address input used by the master processor to indicate whether byte transfer is data or command.
TEST 0, TEST 1	Input pins which can be directly tested using conditional branch instructions.
	T_1 also functions as the event timer input (under software control). T_0 is used during PROM programming and verification in the 8741A.
XTAL1, XTAL2	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	Output signal which occurs once per UPI-41A instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
EA	External access input which allows emulation, testing and PROM/ROM verification.
PROG	Multifunction pin used as the program pulse input during PROM programming.
	During I/O expander access the PROG pin acts as an address/data strobe to the 8243.
$\overline{RESET}$	Input used to reset status flip-flops and to set the program counter to zero.
	$\overline{RESET}$ is also used during PROM programming and verification.
$\overline{SS}$	Single step input used in the 8741A in conjunction with the SYNC output to step the program through each instruction.
V_{CC}	+5V main power supply pin.
V_{DD}	+5V during normal operation. +25V during programming operation. Low power standby pin in ROM version.
V_{SS}	Circuit ground potential.

UPI™ INSTRUCTION SET

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A,Rr	Add register to A	1	1
ADD A,@Rr	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,Rr	Add register to A with carry	1	1
ADDC A,@Rr	Add data memory to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL A,Rr	AND register to A	1	1
ANL A,@Rr	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,Rr	OR register to A	1	1
ORL A,@Rr	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,Rr	Exclusive OR register to A	1	1
XRL A,@Rr	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A,Pp	Input port to A	1	2
OUTL Pp,A	Output A to port	1	2
ANL Pp,#data	AND immediate to port	2	2
ORL Pp,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOV STS,A	A_4-A_7 to Bits 4-7 of Status	1	1
MOVD A,Pp	Input Expander port to A	1	2
MOVD Pp,A	Output A to Expander port	1	2
ANLD Pp,A	AND A to Expander port	1	2
ORLD Pp,A	OR A to Expander port	1	2
DATA MOVES			
MOV A,Rr	Move register to A	1	1
MOV A,@Rr	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV Rr,A	Move A to register	1	1
MOV @Rr,A	Move A to data memory	1	1
MOV Rr,#data	Move immediate to register	2	2
MOV @Rr,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,Rr	Exchange A and register	1	1
XCH A,@Rr	Exchange A and data memory	1	1
XCHD A,@Rr	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3, A,@A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1

Mnemonic	Description	Bytes	Cycles
CONTROL			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
REGISTERS			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ Rr, addr	Decrement register and jump	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
JBb addr	Jump on Accumulator Bit	2	2

APPLICATIONS

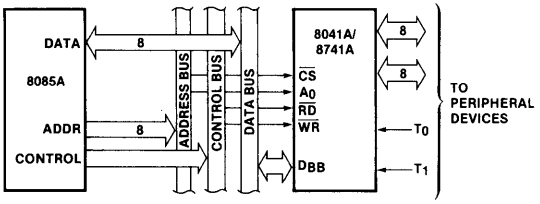


Figure 1. 8085A-8041A Interface

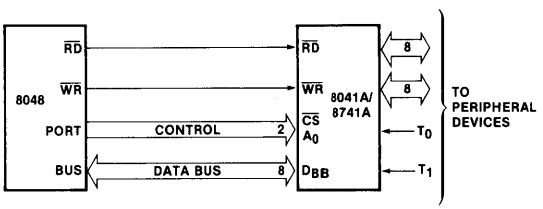


Figure 2. 8048-8041A Interface

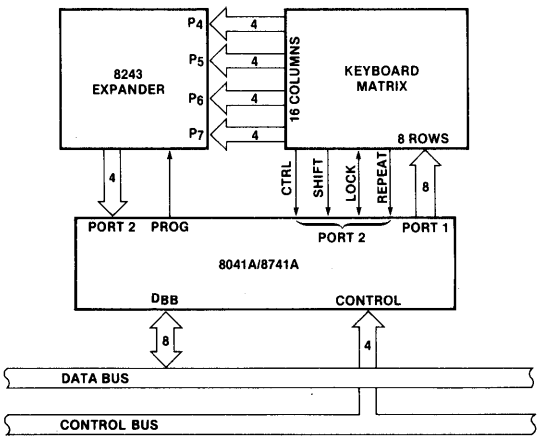


Figure 3. 8041A-8243 Keyboad Scanner

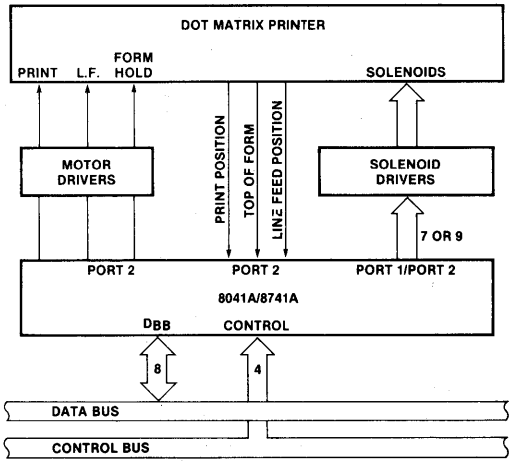


Figure 4. 8041A Matrix Printer Interface

ABSOLUTE MAXIMUM RATINGS*

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 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to 70°C, $V_{SS} = 0V$, 8041A: $V_{CC} = V_{DD} = +5V \pm 10\%$, 8741A: $V_{CC} = V_{DD} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except XTAL1, XTAL2, $\overline{RESET}$)	– 0.5	0.8	V	
V_{IL1}	Input Low Voltage (XTAL1, XTAL2, $\overline{RESET}$)	– 0.5	0.6	V	
V_{IH}	Input High Voltage (Except XTAL1, XTAL2, $\overline{RESET}$)	2.2	V_{CC}		
V_{IH1}	Input High Voltage (XTAL1, XTAL2, $\overline{RESET}$)	3.8	V_{CC}	V	
V_{OL}	Output Low Voltage (D_0-D_7)		0.45	V	$I_{OL} = 2.0$ mA
V_{OL1}	Output Low Voltage ($P_{10}P_{17}$, $P_{20}P_{27}$, Sync)		0.45	V	$I_{OL} = 1.6$ mA
V_{OL2}	Output Low Voltage (Prog)		0.45	V	$I_{OL} = 1.0$ mA
V_{OH}	Output High Voltage (D_0-D_7)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current (T_0, T_1, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0, EA)		± 10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)		± 10	μA	$V_{SS} + 0.45 \le V_{IN} \le V_{CC}$
I_{LI}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$)		0.5	mA	$V_{IL} = 0.8V$
I_{LI1}	Low Input Load Current ($\overline{RESET}$, $\overline{SS}$)		0.2	mA	$V_{IL} = 0.8V$
I_{DD}	V_{DD} Supply Current		15	mA	Typical = 5 mA
$I_{CC} + I_{DD}$	Total Supply Current		125	mA	Typical = 60 mA

A.C. CHARACTERISTICS

$T_A = 0°C$ to 70°C, $V_{SS} = 0V$, 8041A: $V_{CC} = V_{DD} = +5V \pm 10\%$, 8741A: $V_{CC} = V_{DD} = +5V \pm 5\%$

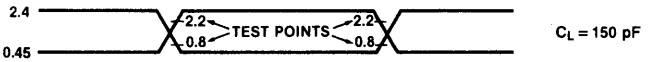
DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}$↓ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{DF}	$\overline{RD}$↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time (Except 8741A-8)	2.5	15	μs	6.0 MHz XTAL
t_{CY}	Cycle Time (8741A-8)	4.17	15	μs	3.6 MHz XTAL

DBB WRITE

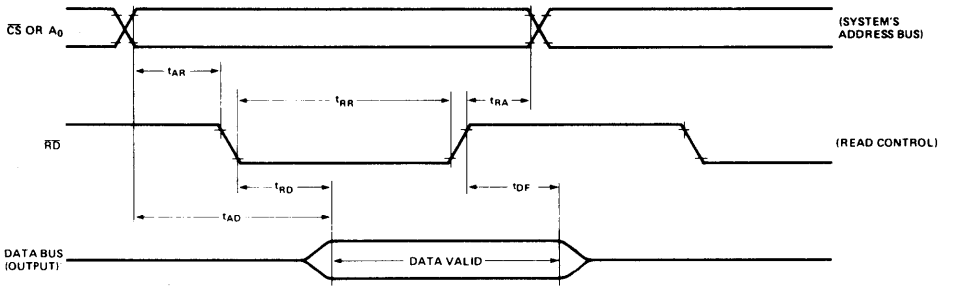
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$↑	150		ns	
t_{WD}	Data Hold After $\overline{WR}$↑	0		ns	

INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS

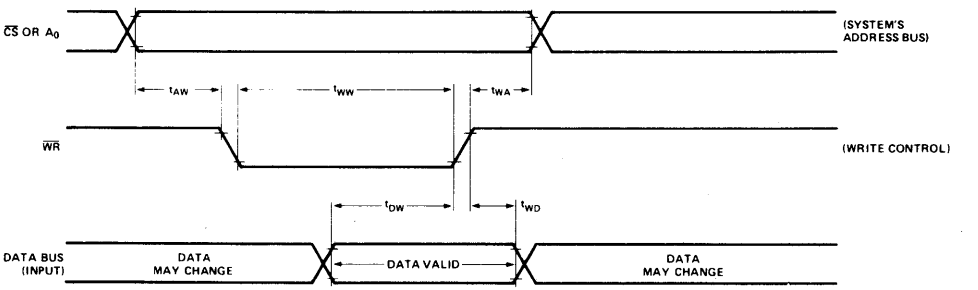


WAVEFORMS

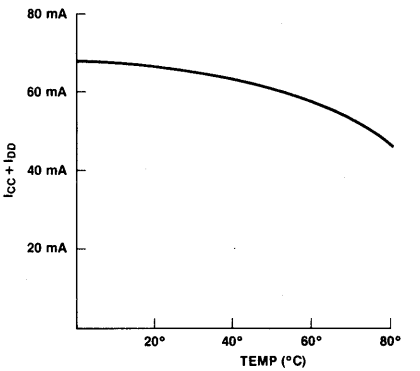
1. READ OPERATION—DATA BUS BUFFER REGISTER.



2. WRITE OPERATION—DATA BUS BUFFER REGISTER.



TYPICAL 8041/8741A CURRENT



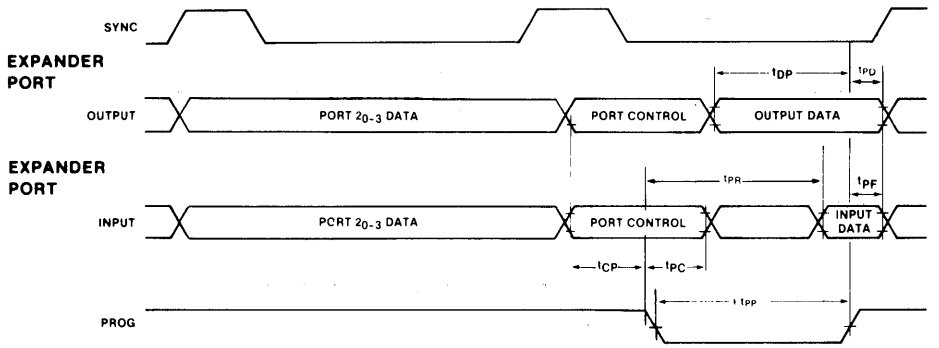
00188A

A.C. CHARACTERISTICS—PORT 2

$T_A = 0°C$ to $70°C$, 8041A: $V_{CC} = +5V \pm 10\%$, 8741A: $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CP}	Port Control Setup Before Falling Edge of PROG	110		ns	
t_{PC}	Port Control Hold After Falling Edge of PROG	100		ns	
t_{PR}	PROG to Time P2 Input Must Be Valid		810	ns	
t_{PF}	Input Data Hold Time	0	150	ns	
t_{DP}	Output Data Setup Time	250		ns	
t_{PD}	Output Data Hold Time	65		ns	
t_{PP}	PROG Pulse Width	1200		ns	

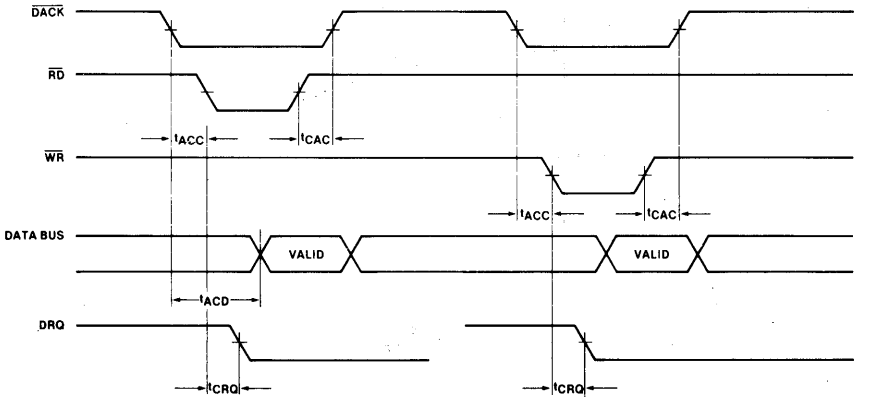
PORT 2 TIMING



A.C. CHARACTERISTICS—DMA

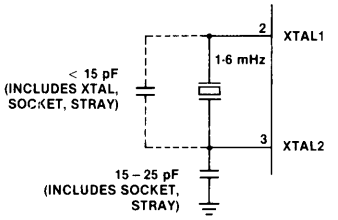
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{ACC}	$\overline{DACK}$ to $\overline{WR}$ or $\overline{RD}$	0		ns	
t_{CAC}	$\overline{RD}$ or $\overline{WR}$ to $\overline{DACK}$	0		ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		225	ns	$C_L = 150$ pF
t_{CRQ}	$\overline{RD}$ or $\overline{WR}$ to DRQ Cleared		200	ns	

WAVEFORMS—DMA



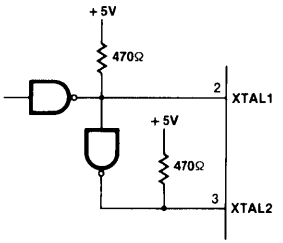
00188A

CRYSTAL OSCILLATOR MODE



CRYSTAL SERIES RESISTANCE SHOULD BE <75Ω AT 6 MHz; <180Ω AT 3.6 MHz.

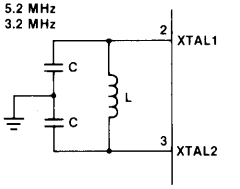
DRIVING FROM EXTERNAL SOURCE



BOTH XTAL1 AND XTAL2 SHOULD BE DRIVEN. RESISTORS TO V_{CC} ARE NEEDED TO ENSURE V_{IH} = 3.8V IF TTL CIRCUITRY IS USED.

LC OSCILLATOR MODE

L	C	NOMINAL f
45 μH	20 pF	5.2 MHz
120 μH	20 pF	3.2 MHz



$$f \approx \frac{1}{2\pi\sqrt{LC'}}$$

$$C' = \frac{C + 3C_{PP}}{2}$$

$C_{PP} \approx 5 - 10$ pF PIN-TO-PIN CAPACITANCE

EACH C SHOULD BE APPROXIMATELY 20 pF, INCLUDING STRAY CAPACITANCE.

PROGRAMMING, VERIFYING, AND ERASING THE 8741A EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock Input (1 to 6MHz)
$\overline{\text{Reset}}$	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING:

An attempt to program a missocketed 8741A will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. A_0 = 0V, $\overline{CS}$ = 5V, EA = 5V, $\overline{RESET}$ = 0V, TEST0 = 5V, V_{DD} = 5V, clock applied or internal oscillator operating, BUS and PROG floating.
2. Insert 8741A in programming socket
3. TEST 0 = 0v (select program mode)
4. EA = 23V (activate program mode)
5. Address applied to BUS and P20-1
6. $\overline{RESET}$ = 5v (latch address)
7. Data applied to BUS
8. V_{DD} = 25v (programming power)
9. PROG = 0v followed by one 50ms pulse to 23V
10. V_{DD} = 5v
11. TEST 0 = 5v (verify mode)
12. Read and verify data on BUS
13. TEST 0 = 0v
14. $\overline{RESET}$ = 0v and repeat from step 5
15. Programmer should be at conditions of step 1 when 8741A is removed from socket.

00188A

8741A Erasure Characteristics

The erasure characteristics of the 8741A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8741A in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8741A window to prevent unintentional erasure.

The recommended erasure procedure for the 8741A is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15 w-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm^2 power rating. The 8741A should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

A.C. TIMING SPECIFICATION FOR PROGRAMMING

T_A = 25°C ±5°C, V_{CC} = 5V ±5%, V_{DD} = 25V ±1V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	Address Setup Time to $\overline{\text{RESET}}$ ↑	4tcy			
t_{WA}	Address Hold Time After $\overline{\text{RESET}}$ ↑	4tcy			
t_{DW}	Data in Setup Time to PROG ↑	4tcy			
t_{WD}	Data in Hold Time After PROG ↓	4tcy			
t_{PH}	$\overline{\text{RESET}}$ Hold Time to Verify	4tcy			
t_{VDDW}	V_{DD} Setup Time to PROG ↑	4tcy			
t_{VDDH}	V_{DD} Hold Time After PROG ↓	0			
t_{PW}	Program Pulse Width	50	60	mS	
t_{TW}	Test 0 Setup Time for Program Mode	4tcy			
t_{WT}	Test 0 Hold Time After Program Mode	4tcy			
t_{DO}	Test 0 to Data Out Delay		4tcy		
t_{WW}	$\overline{\text{RESET}}$ Pulse Width to Latch Address	4tcy			
t_r, t_f	V_{DD} and PROG Rise and Fall Times	0.5	2.0	μs	
t_{CY}	CPU Operation Cycle Time	5.0		μs	
t_{RE}	$\overline{\text{RESET}}$ Setup Time Before EA ↑.	4tcy			

Note: If TEST 0 is high, t_{DO} can be triggered by $\overline{\text{RESET}}$ ↑.

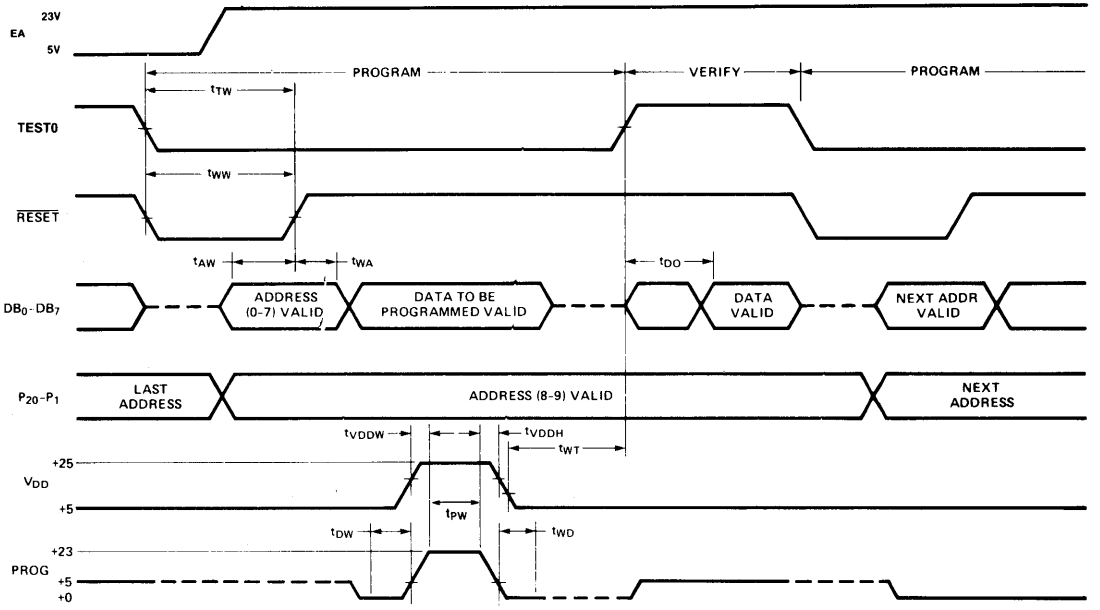
D.C. SPECIFICATION FOR PROGRAMMING

T_A = 25°C ±5°C, V_{CC} = 5V ±5%, V_{DD} = 25V ±1V

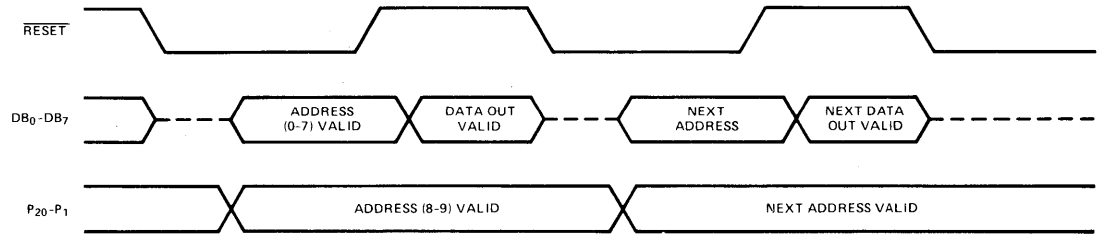
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{DOH}	V_{DD} Program Voltage High Level	24.0	26.0	V	
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V	
V_{PH}	PROG Program Voltage High Level	21.5	24.5	V	
V_{PL}	PROG Voltage Low Level		0.2	V	
V_{EAH}	EA Program or Verify Voltage High Level	21.5	24.5	V	
V_{EAL}	EA Voltage Low Level		5.25	V	
I_{DD}	V_{DD} High Voltage Supply Current		30.0	mA	
I_{PROG}	PROG High Voltage Supply Current		16.0	mA	
I_{EA}	EA High Voltage Supply Current		1.0	mA	

WAVEFORMS FOR PROGRAMMING

COMBINATION PROGRAM/VERIFY MODE (EPROM'S ONLY)



VERIFY MODE (ROM/EPROM)



NOTES:
1. **PROG MUST FLOAT IF EA IS LOW (I.e., ≠23V), OR IF T0 = 5V FOR THE 8741A. FOR THE 8041A PROG MUST ALWAYS FLOAT.**
2. **XTAL1 AND XTAL 2 DRIVEN BY 3.6 MHz CLOCK WILL GIVE 4.17 μsec t_{CY}. THIS IS ACCEPTABLE FOR 8741A-8 PARTS AS WELL AS STANDARD PARTS.**
3. **AO MUST BE HELD LOW (I.e., = 0V) DURING PROGRAM/VERIFY MODES.**

The 8741A EPROM can be programmed by either of two Intel products:

1. PROMPT-48 Microcomputer Design Aid, or
2. Universal PROM Programmer (UPP series) peripheral of the Intellec® Development System with a UPP-848 Personality Card.

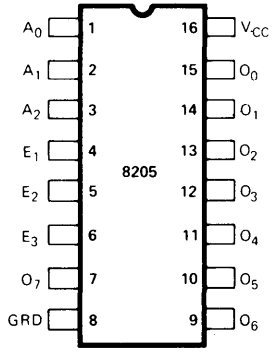
8205
HIGH SPEED 1 OUT OF 8 BINARY DECODER

- **I/O Port or Memory Selector**
- **Simple Expansion — Enable Inputs**
- **High Speed Schottky Bipolar Technology — 18 ns Max Delay**
- **Directly Compatible with TTL Logic Circuits**
- **Low Input Load Current — 0.25 mA Max, 1/6 Standard TTL Input Load**
- **Minimum Line Reflection — Low Voltage Diode Input Clamp**
- **Outputs Sink 10 mA Min**
- **16-Pin Dual In-Line Ceramic or Plastic Package**

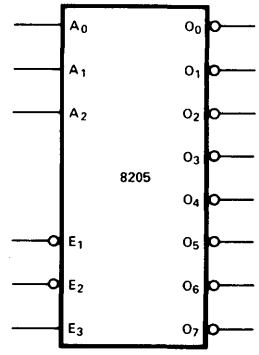
The Intel® 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its 8 outputs goes "low", thus a single row of a memory system is selected. The 3-chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive 8 other decoders for arbitrary memory expansions.

The 8205 is packaged in a standard 16-pin dual in-line package, and its performance is specified over the temperature range of 0°C to +75°C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffusion process.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

A_0-A_2	ADDRESS INPUTS
$\overline{E_1}$, $\overline{E_2}$, E_3	ENABLE INPUTS
$\overline{O_0}$-$\overline{O_7}$	DECODED OUTPUTS

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

FUNCTIONAL DESCRIPTION

Decoder

The 8205 contains a one out of eight binary decoder. It accepts a three bit binary code and by gating this input, creates an exclusive output that represents the value of the input code.

For example, if a binary code of 101 was present on the A0, A1 and A2 address input lines, and the device was enabled, an active low signal would appear on the $\overline{O5}$ output line. Note that all of the other output pins are sitting at a logic high, thus the decoded output is said to be exclusive. The decoders outputs will follow the truth table shown below in the same manner for all other input variations.

Enable Gate

When using a decoder it is often necessary to gate the outputs with timing or enabling signals so that the exclusive output of the decoded value is synchronous with the overall system.

The 8205 has a built-in function for such gating. The three enable inputs ($\overline{E1}$, $\overline{E2}$, E3) are ANDed together and create a single enable signal for the decoder. The combination of both active "high" and active "low" device enable inputs provides the designer with a powerfully flexible gating function to help reduce package count in his system.

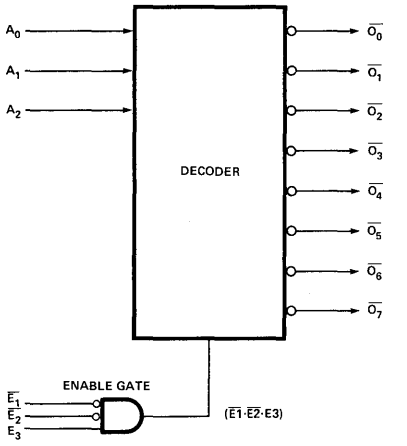


Figure 1. Enable Gate

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias:	Ceramic	−65°C to +125°C
	Plastic	−65°C to +75°C
Storage Temperature		−65°C to +160°C
All Output or Supply Voltages		−0.5 to +7 Volts
All Input Voltages		−1.0 to +5.5 Volts
Output Currents		125 mA

*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

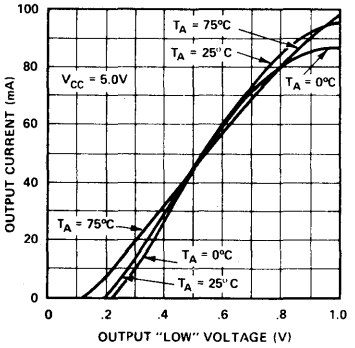
T_A = 0°C to +75°C, V_{CC} = 5V ±5%

8205

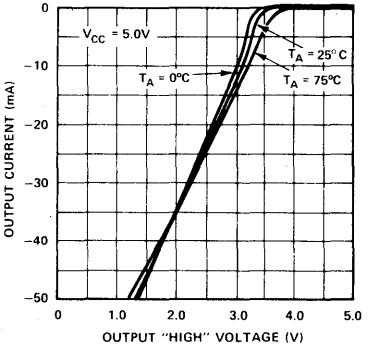
SYMBOL	PARAMETER	LIMIT MIN.	LIMIT MAX.	UNIT	TEST CONDITIONS
I_F	INPUT LOAD CURRENT		−0.25	mA	V_{CC} = 5.25V, V_F = 0.45V
I_R	INPUT LEAKAGE CURRENT		10	µA	V_{CC} = 5.25V, V_R = 5.25V
V_C	INPUT FORWARD CLAMP VOLTAGE		−1.0	V	V_{CC} = 4.75V, I_C = −5.0 mA
V_{OL}	OUTPUT "LOW" VOLTAGE		0.45	V	V_{CC} = 4.75V, I_{OL} = 10.0 mA
V_{OH}	OUTPUT HIGH VOLTAGE	2.4		V	V_{CC} = 4.75V, I_{OH} = −1.5 mA
V_{IL}	INPUT "LOW" VOLTAGE		0.85	V	V_{CC} = 5.0V
V_{IH}	INPUT "HIGH" VOLTAGE	2.0		V	V_{CC} = 5.0V
I_{SC}	OUTPUT HIGH SHORT CIRCUIT CURRENT	−40	−120	mA	V_{CC} = 5.0V, V_{OUT} = 0V
V_{OX}	OUTPUT "LOW" VOLTAGE @ HIGH CURRENT		0.8	V	V_{CC} = 5.0V, I_{OX} = 40 mA
I_{CC}	POWER SUPPLY CURRENT		70	mA	V_{CC} = 5.25V

TYPICAL CHARACTERISTICS

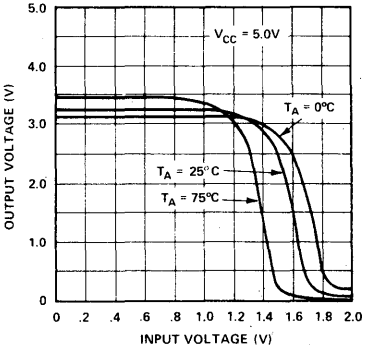
OUTPUT CURRENT VS. OUTPUT "LOW" VOLTAGE



OUTPUT CURRENT VS. OUTPUT "HIGH" VOLTAGE



DATA TRANSFER FUNCTION



SWITCHING CHARACTERISTICS

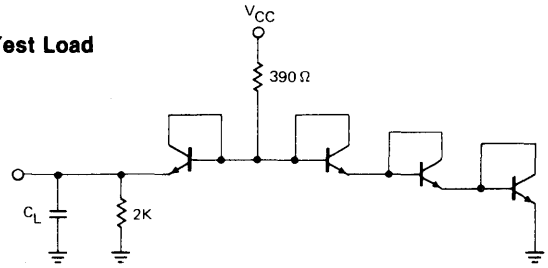
Conditions of Test:

Input pulse amplitudes: 2.5V

Input rise and fall times: 5 nsec between 1V and 2V

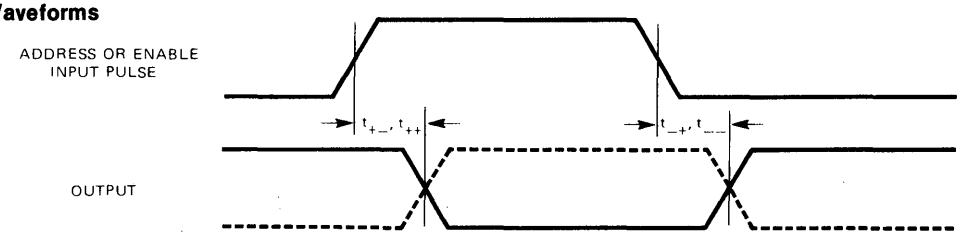
Measurements are made at 1.5V

Test Load



All Transistors 2N2369 or Equivalent. C_L = 30 pF

Test Waveforms



A.C. CHARACTRISTICS

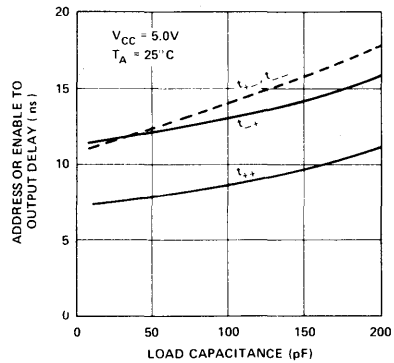
T_A = 0°C to +75°C, V_{CC} = 5V ±5% unless otherwise specified.

SYMBOL	PARAMETER		MAX. LIMIT	UNIT	TEST CONDITIONS
t_{++}	ADDRESS OR ENABLE TO OUTPUT DELAY		18	ns	
t_{-+}			18	ns	
t_{+-}			18	ns	
t_{--}			18	ns	
C_{IN} (1)	INPUT CAPACITANCE	P8205	4(typ.)	pF	f = 1 MHz, V_{CC} = 0V
		C8205	5(typ.)	pF	V_{BIAS} = 2.0V, T_A = 25°C

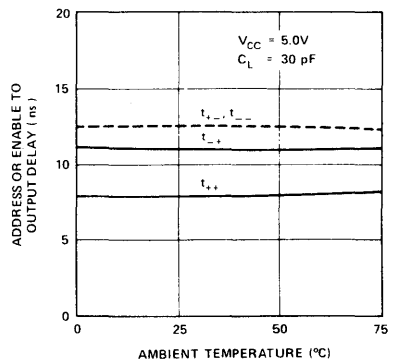
1. This parameter is periodically sampled and is not 100% tested.

TYPICAL CHARACTERISTICS

ADDRESS OR ENABLE TO OUTPUT DELAY VS. LOAD CAPACITANCE



ADDRESS OR ENABLE TO OUTPUT DELAY VS. AMBIENT TEMPERATURE

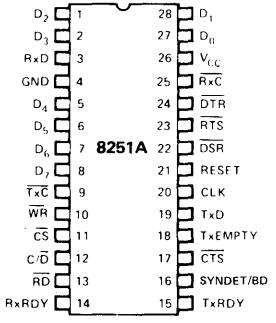


8251A/S2657
PROGRAMMABLE COMMUNICATION INTERFACE

- **Synchronous and Asynchronous Operation**
- **Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion**
- **Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling.**
- **Synchronous Baud Rate — DC to 64K Baud**
- **Asynchronous Baud Rate — DC to 19.2K Baud**
- **Full Duplex, Double Buffered, Transmitter and Receiver**
- **Error Detection — Parity, Overrun and Framing**
- **Fully Compatible with 8080/8085 CPU**
- **28-Pin DIP Package**
- **All Inputs and Outputs are TTL Compatible**
- **Single +5V Supply**
- **Single TTL Clock**

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family of microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silicon gate technology.

PIN CONFIGURATION

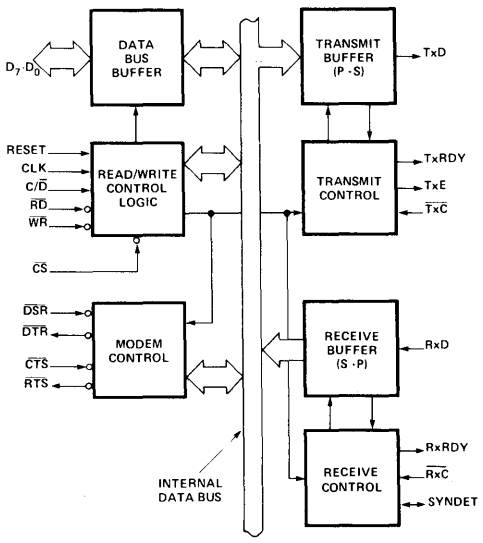


PIN NAMES

D_7-D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char from 8080)

$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground

BLOCK DIAGRAM



FEATURES AND ENHANCEMENTS

8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the $\overline{RD}$ and $\overline{WR}$ do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Synchronous Baud rate from DC to 64K.
- Fully compatible with Intel's new industry standard, the MCS-85.

ABSOLUTE MAXIMUM RATINGS*

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 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 μA
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} TO 0.45V
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} TO 0.45V
I_{CC}	Power Supply Current		100	mA	All Outputs = High

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

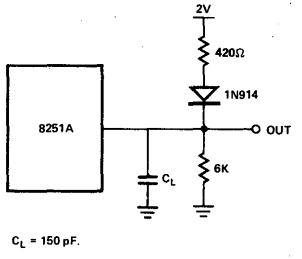


Figure 16. Test Load Circuit

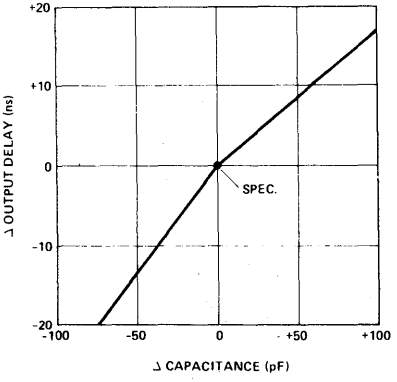


Figure 17. Typical Δ Output Delay vs. Δ Capacitance (pF)

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

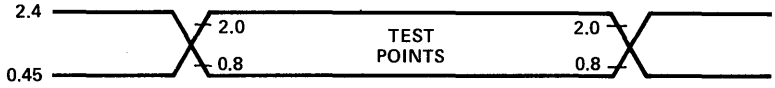
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AR}	Address Stable Before $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	50		ns	Note 2
t_{RA}	Address Hold Time for $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	50		ns	Note 2
t_{RR}	$\overline{READ}$ Pulse Width	250		ns	
t_{RD}	Data Delay from $\overline{READ}$		250	ns	3, C_L = 150 pF
t_{DF}	$\overline{READ}$ to Data Floating	10	100	ns	

Write Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AW}	Address Stable Before $\overline{WRITE}$	50		ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	50		ns	
t_{WW}	$\overline{WRITE}$ Pulse Width	250		ns	
t_{DW}	Data Set Up Time for $\overline{WRITE}$	150		ns	
t_{WD}	Data Hold Time for $\overline{WRITE}$	50		ns	
t_{RV}	Recovery Time Between WRITES	6		t_{CY}	Note 4

NOTES: 1. AC timings measured V_{OH} = 2.0, V_{OL} = 0.8, and with load circuit of Figure 1.
2. Chip Select ($\overline{CS}$) and Command/Data (C/$\overline{D}$) are considered as Addresses.
3. Assumes that Address is valid before $\overline{RD}\downarrow$.
4. This recovery time is for Mode Initialization only. Write Data is allowed only when TxRDY = 1.
Recovery Time between Writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 16 t_{CY}.

Input Waveforms for AC Tests



Other Timings:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{CY}	Clock Period	320	1350	ns	Notes 5, 6
t_{ϕ}	Clock High Pulse Width	140	t_{CY-90}	ns	
$t_{\overline{\phi}}$	Clock Low Pulse Width	90		ns	
t_R, t_F	Clock Rise and Fall Time		20	ns	
t_{DTx}	TxD Delay from Falling Edge of $\overline{TxC}$		1	μs	
f_{Tx}	Transmitter Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{TPW}	Transmitter Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{TPD}	Transmitter Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
f_{Rx}	Receiver Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{RPW}	Receiver Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{RPD}	Receiver Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
t_{TxRDY}	TxRDY Pin Delay from Center of last Bit		8	t_{CY}	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY ↓ from Leading Edge of $\overline{WR}$		6	t_{CY}	Note 7
t_{RxRDY}	RxRDY Pin Delay from Center of last Bit		24	t_{CY}	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY ↓ from Leading Edge of $\overline{RD}$		6	t_{CY}	Note 7
t_{IS}	Internal SYNDET Delay from Rising Edge of $\overline{RxC}$		24	t_{CY}	Note 7
t_{ES}	External SYNDET Set-Up Time Before Falling Edge of $\overline{RxC}$	16		t_{CY}	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Last Bit	20		t_{CY}	Note 7
t_{WC}	Control Delay from Rising Edge of WRITE (TxEn, $\overline{DTR}$, $\overline{RTS}$)	8		t_{CY}	Note 7
t_{CR}	Control to READ Set-Up Time ($\overline{DSR}$, $\overline{CTS}$)	20		t_{CY}	Note 7

5. The TxC and RxC frequencies have the following limitations with respect to CLK.
   For 1x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(30\ t_{CY})$
   For 16x and 64x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(4.5\ t_{CY})$

6. Reset Pulse Width = 6 t_{CY} minimum; System Clock must be running during Reset.
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

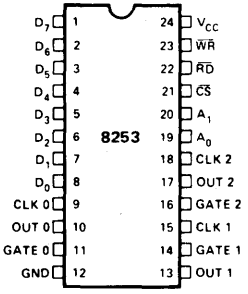
8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- **MCS—85™ Compatible 8253-5**
- **3 Independent 16-Bit Counters**
- **DC to 2 MHz**
- **Programmable Counter Modes**
- **Count Binary or BCD**
- **Single +5V Supply**
- **24-Pin Dual In-Line Package**

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

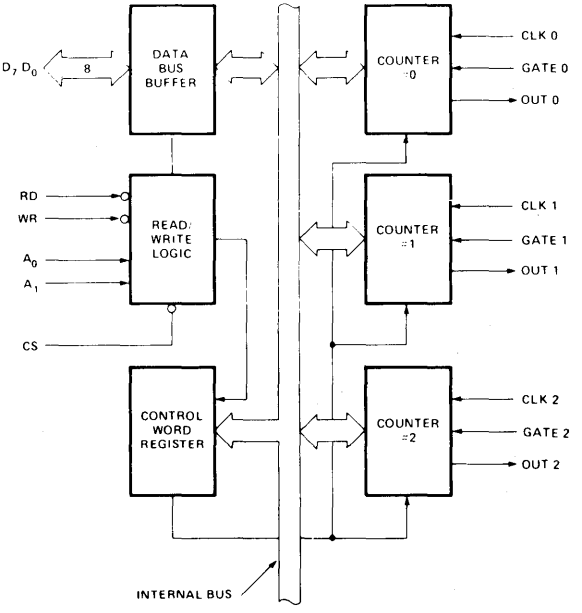
PIN CONFIGURATION



PIN NAMES

$D_7\ D_0$	DATA BUS (8 BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
$\overline{RD}$	READ COUNTER
$\overline{WR}$	WRITE COMMAND OR DATA
$\overline{CS}$	CHIP SELECT
$A_0\ A_1$	COUNTER SELECT
V_{CC}	+5 VOLTS
GND	GROUND

BLOCK DIAGRAM



AFN-00745A-01

FUNCTIONAL DESCRIPTION

General

The 8253 is a programmable interval timer/counter specifically designed for use with the Intel™ Microcomputer systems. Its function is that of a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

- Programmable Rate Generator
- Event Counter
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput CPU instructions. The Data Bus Buffer has three basic functions.

1. Programming the MODES of the 8253.
2. Loading the count registers.
3. Reading the count values.

Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and in turn generates control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic.

$\overline{RD}$ (Read)

A "low" on this input informs the 8253 that the CPU is inputting data in the form of a counters value.

$\overline{WR}$ (Write)

A "low" on this input informs the 8253 that the CPU is outputting data in the form of mode information or loading counters.

A0, A1

These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

$\overline{CS}$ (Chip Select)

A "low" on this input enables the 8253. No reading or writing will occur unless the device is selected. The $\overline{CS}$ input has no effect upon the actual operation of the counters.

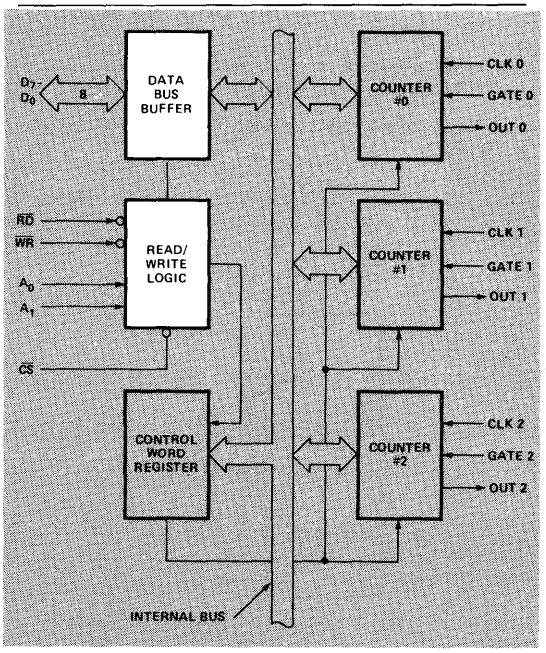


Figure 1. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

Control Word Register

The Control Word Register is selected when A0, A1 are 11. It then accepts information from the data bus buffer and stores it in a register. The information stored in this register controls the operational MODE of each counter, selection of binary or BCD counting and the loading of each count register.

The Control Word Register can only be written into; no read operation of its contents is available.

Counter #0, Counter #1, Counter #2

These three functional blocks are identical in operation so only a single Counter will be described. Each Counter consists of a single, 16-bit, pre-settable, DOWN counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of MODES stored in the Control Word Register.

The counters are fully independent and each can have separate Mode configuration and counting operation, binary or BCD. Also, there are special features in the control word that handle the loading of the count value so that software overhead can be minimized for these functions.

The reading of the contents of each counter is available to the programmer with simple READ operations for event counting applications and special commands and logic are included in the 8253 so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

8253 SYSTEM INTERFACE

The 8253 is a component of the Intel™ Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A0, A1 connect to the A0, A1 address bus signals of the CPU. The $\overline{CS}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel® 8205 for larger systems.

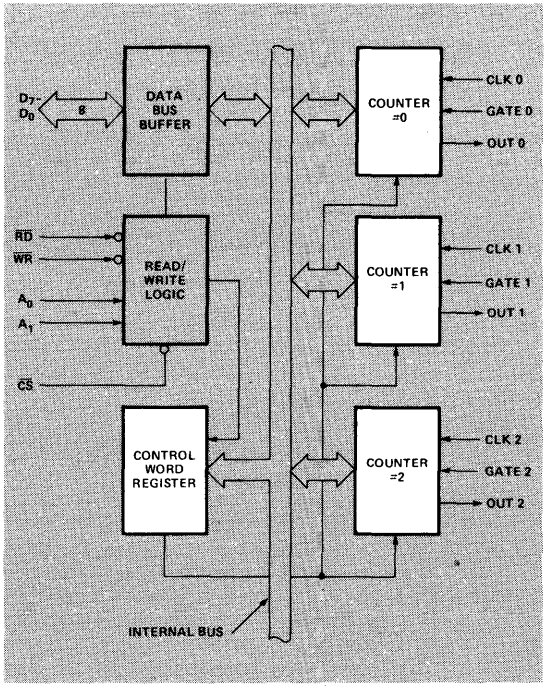


Figure 2. Block Diagram Showing Control Word Register and Counter Functions

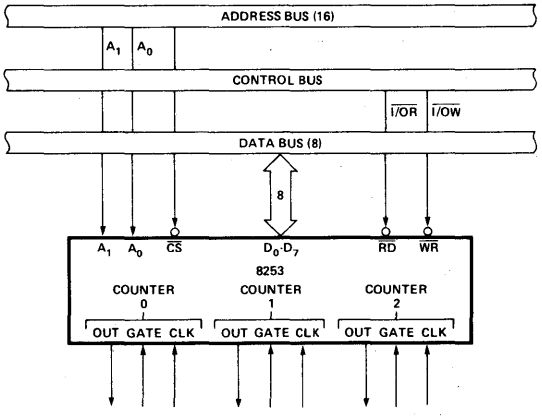
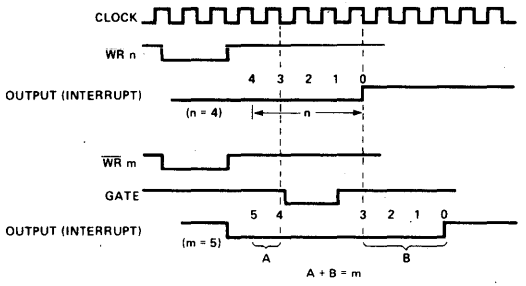
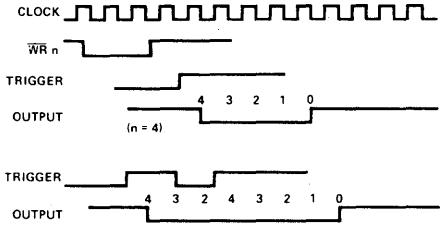


Figure 3. 8253 System Interface

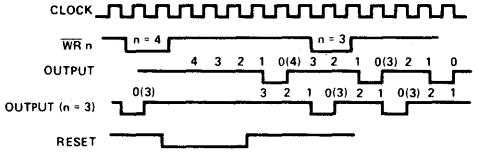
MODE 0: Interrupt on Terminal Count



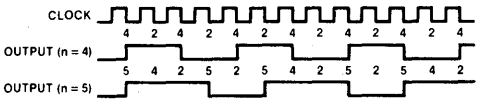
MODE 1: Programmable One-Shot



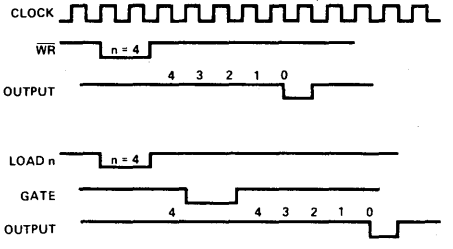
MODE 2: Rate Generator



MODE 3: Square Wave Generator



MODE 4: Software Triggered Strobe



MODE 5: Hardware Triggered Strobe

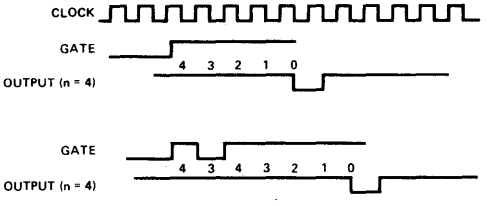


Figure 5. 8253 Timing Diagrams

8253 READ/WRITE PROCEDURE

Write Operations

The systems software must program each counter of the 8253 with the mode and quantity desired. The programmer must write out to the 8253 a MODE control word and the programmed number of count register bytes (1 or 2) prior to actually using the selected counter.

The actual order of the programming is quite flexible. Writing out of the MODE control word can be in any sequence of counter selection, e.g., counter #0 does not have to be first or counter #2 last. Each counter's MODE control word register has a separate address so that its loading is completely sequence independent. (SC0, SC1)

The loading of the Count Register with the actual count value, however, must be done in exactly the sequence programmed in the MODE control word (RL0, RL1). This loading of the counter's count register is still sequence independent like the MODE control word loading, but when a selected count register is to be loaded it must be loaded with the number of bytes programmed in the MODE control word (RL0, RL1). The one or two bytes to be loaded in the count register do not have to follow the associated MODE control word. They can be programmed at any time following the MODE control word loading as long as the correct number of bytes is loaded in order.

All counters are down counters. Thus, the value loaded into the count register will actually be decremented. Loading all zeroes into a count register will result in the maximum count (2^{16} for Binary or 10^4 for BCD). In MODE 0 the new count will not restart until the load has been completed. It will accept one of two bytes depending on how the MODE control words (RL0, RL1) are programmed. Then proceed with the restart operation.

	MODE Control Word Counter n
LSB	Count Register byte Counter n
MSB	Count Register byte Counter n

Note: Format shown is a simple example of loading the 8253 and does not imply that it is the only format that can be used.

Figure 6. Programming Format

			A1	A0
No. 1		MODE Control Word Counter 0	1	1
No. 2		MODE Control Word Counter 1	1	1
No. 3		MODE Control Word Counter 2	1	1
No. 4	LSB	Count Register Byte Counter 1	0	1
No. 5	MSB	Count Register Byte Counter 1	0	1
No. 6	LSB	Count Register Byte Counter 2	1	0
No. 7	MSB	Count Register Byte Counter 2	1	0
No. 8	LSB	Count Register Byte Counter 0	0	0
No. 9	MSB	Count Register Byte Counter 0	0	0

Note: The exclusive addresses of each counter's count register make the task of programming the 8253 a very simple matter, and maximum effective use of the device will result if this feature is fully utilized.

Figure 7. Alternate Programming Formats

Read Operations

In most counter applications it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common application that uses this function. The 8253 contains logic that will allow the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

There are two methods that the programmer can use to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. By controlling the A0, A1 inputs to the 8253 the programmer can select the counter to be read (remember that no read operation of the mode register is allowed A0, A1-11). The only requirement with this method is that in order to assure a stable count reading the actual operation of the selected counter must be inhibited either by controlling the Gate input or by external logic that inhibits the clock input. The contents of the counter selected will be available as follows:

first I/O Read contains the least significant byte (LSB).

second I/O Read contains the most significant byte (MSB).

Due to the internal logic of the 8253 it is absolutely necessary to complete the entire reading procedure. If two bytes are programmed to be read then two bytes must be read before any loading WR command can be sent to the same counter.

Read Operation Chart

A1	A0	RD	
0	0	0	Read Counter No. 0
0	1	0	Read Counter No. 1
1	0	0	Read Counter No. 2
1	1	0	Illegal

Reading While Counting

In order for the programmer to read the contents of any counter without effecting or disturbing the counting operation the 8253 has special internal logic that can be accessed using simple WR commands to the MODE register. Basically, when the programmer wishes to read the contents of a selected counter "on the fly" he loads the MODE register with a special code which latches the present count value into a storage register so that its contents contain an accurate, stable quantity. The programmer then issues a normal read command to the selected counter and the contents of the latched register is available.

MODE Register for Latching Count

A0, A1 = 11

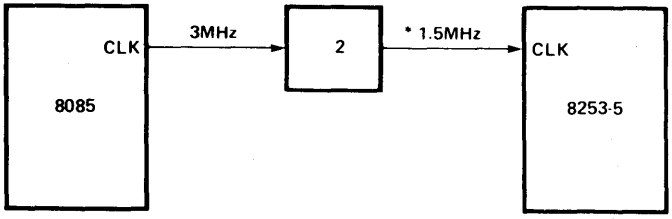
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1,SC0 — specify counter to be latched.

D5,D4 — 00 designates counter latching operation.

X — don't care.

The same limitation applies to this mode of reading the counter as the previous method. That is, it is mandatory to complete the entire read operation as programmed. This command has no effect on the counter's mode.



*If an 8085 clock output is to drive an 8253-5 clock input, it must be reduced to 2 MHz or less.

Figure 8. MCS-85™ Clock Interface*

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground ..... −0.5 V to +7 V
Power Dissipation ..... 1 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ±5%)

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}+.5V	V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH}	Output High Voltage	2.4		V	Note 2
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		140	mA	

Note 1: 8253, I_{OL} = 1.6 mA; 8253-5, I_{OL} = 2.2 mA.
Note 2: 8253, I_{OH} = −150 μA; 8253-5, I_{OH} = −400 μA.

CAPACITANCE T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$; $V_{CC} = 5.0V \pm 5\%$; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

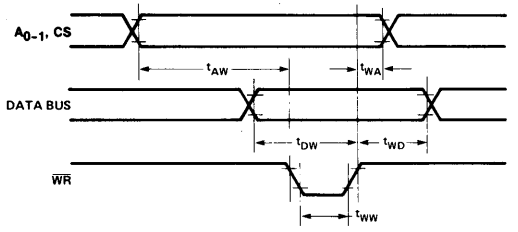
SYMBOL	PARAMETER	8253 MIN.	8253 MAX.	8253-5 MIN.	8253-5 MAX.	UNIT
t_{AR}	Address Stable Before $\overline{READ}$	50		30		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		300		ns
t_{RD}	Data Delay From $\overline{READ}$[2]		300		200	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	25	100	ns
t_{RV}	Recovery Time Between READ and Any Other Control Signal	1		1		μs

Write Cycle:

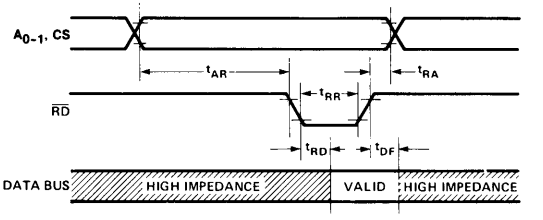
SYMBOL	PARAMETER	8253 MIN.	8253 MAX.	8253-5 MIN.	8253-5 MAX.	UNIT
t_{AW}	Address Stable Before $\overline{WRITE}$	50		30		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	30		30		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		250		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		30		ns
t_{RV}	Recovery Time Between $\overline{WRITE}$ and Any Other Control Signal	1		1		μs

Notes: 1. AC timings measured at $V_{OH} = 2.2$, $V_{OL} = 0.8$
2. Test Conditions: 8253, $C_L = 100pF$; 8253-5: $C_L = 150pF$.

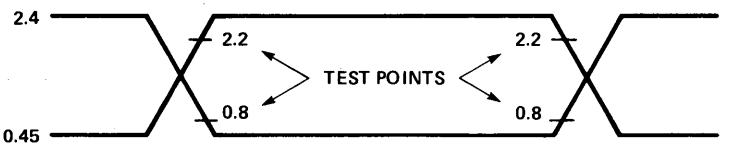
Write Timing:



Read Timing:



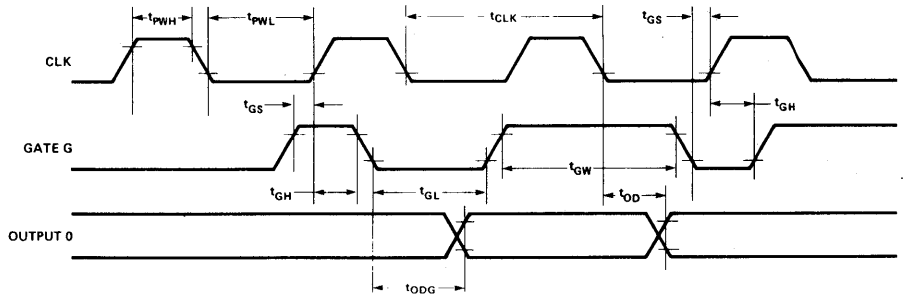
Input Waveforms for A.C. Tests:



Clock and Gate Timing:

SYMBOL	PARAMETER	8253		8253-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{CLK}	Clock Period	380	dc	380	dc	ns
t_{PWH}	High Pulse Width	230		230		ns
t_{PWL}	Low Pulse Width	150		150		ns
t_{GW}	Gate Width High	150		150		ns
t_{GL}	Gate Width Low	100		100		ns
t_{GS}	Gate Set Up Time to CLK↑	100		100		ns
t_{GH}	Gate Hold Time After CLK↑	50		50		ns
t_{OD}	Output Delay From CLK↓[1]		400		400	ns
t_{ODG}	Output Delay From Gate↓[1]		300		300	ns

Note 1: Test Conditions: 8253: C_L = 100pF; 8253-5: C_L = 150pF.

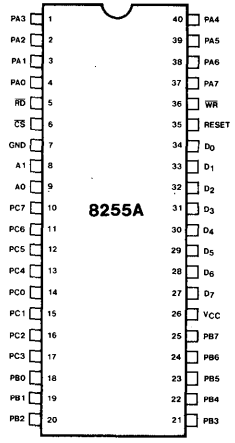


8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

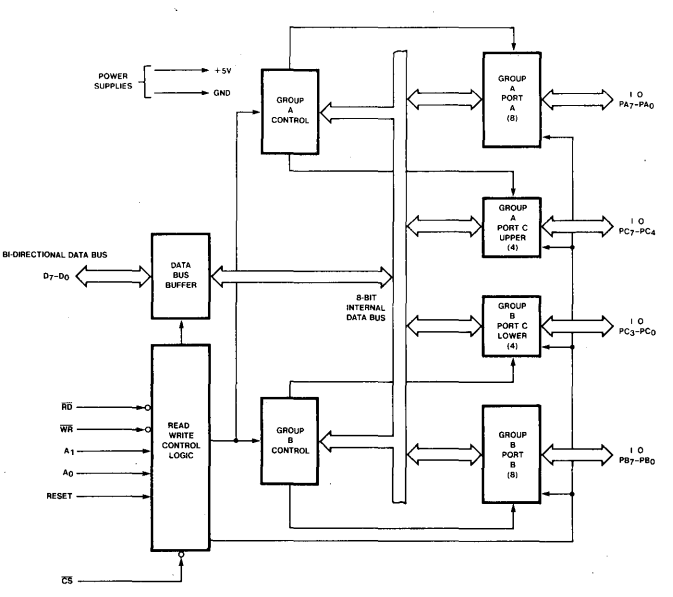
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A BLOCK DIAGRAM



8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel® microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select. A "low" on this input pin enables the communiction between the 8255A and the CPU.

($\overline{RD}$)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

($\overline{WR}$)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A ⇒ DATA BUS
0	1	0	1	0	PORT B ⇒ DATA BUS
1	0	0	1	0	PORT C ⇒ DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS ⇒ PORT A
0	1	1	0	0	DATA BUS ⇒ PORT B
1	0	1	0	0	DATA BUS ⇒ PORT C
1	1	1	0	0	DATA BUS ⇒ CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS ⇒ 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS ⇒ 3-STATE

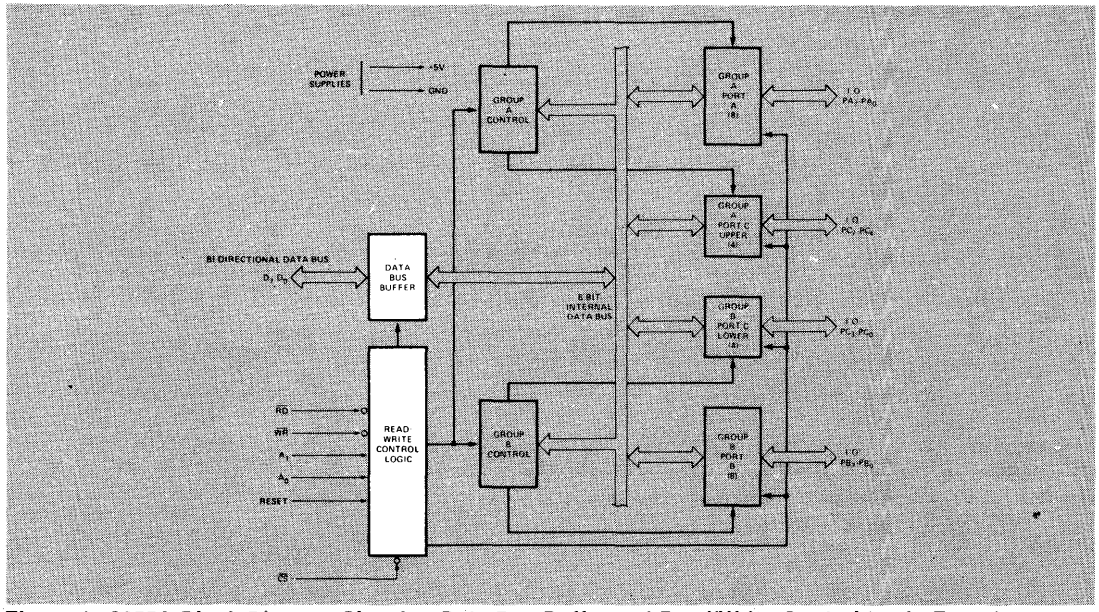


Figure 1. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

(RESET)

Reset. A "high on this input clears the control register and all ports (A, C, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)
Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can **Only** be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

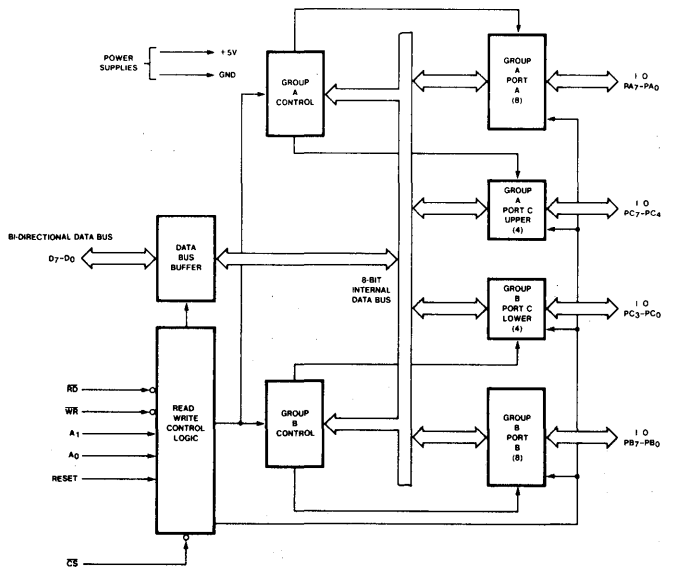
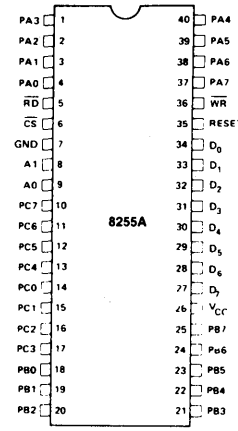


Figure 2. 8225A Block Diagram Showing Group A and Group B Control Functions

PIN CONFIGURATION



PIN NAMES

D_7 D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

ABSOLUTE MAXIMUM RATINGS*

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 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

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

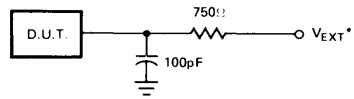
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL} (DB)	Output Low Voltage (Data Bus)		0.45	V	I_{OL} = 2.5mA
V_{OL}(PER)	Output Low Voltage (Peripheral Port)		0.45	V	I_{OL} = 1.7mA
V_{OH}(DB)	Output High Voltage (Data Bus)	2.4		V	I_{OH} = -400μA
V_{OH}(PER)	Output High Voltage (Peripheral Port)	2.4		V	I_{OH} = -200μA
I_{DAR}[1]	Darlington Drive Current	-1.0	-4.0	mA	R_{EXT} = 750Ω; V_{EXT} = 1.5V
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V

Note 1: Available on any 8 pins from Port B and C.

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND



*V_{EXT} is set at various voltages during testing to guarantee the specification.

Figure 24. Test Load Circuit (for dB)

A.C. CHARACTERISTICS

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

NOTE: The 8255A-5 specifications are not final. Some parametric limits are subject to change.

Bus Parameters

Read:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{AR}	Address Stable Before READ	0		0		ns
t_{RA}	Address Stable After READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid From READ[1]		250		200	ns
t_{DF}	Data Float After READ	10	150	10	100	ns
t_{RV}	Time Between READs and/or WRITEs	850		850		ns

Write:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{AW}	Address Stable Before WRITE	0		0		ns
t_{WA}	Address Stable After WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid After WRITE	30		30		ns

Other Timings:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{WB}	WR = 1 to Output[1]		350		350	ns
t_{IR}	Peripheral Data Before RD	0		0		ns
t_{HR}	Peripheral Data After RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data Before T.E. of STB	0		0		ns
t_{PH}	Per. Data After T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output[1]		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns
t_{WOB}	WR = 1 to OBF = 0[1]		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1[1]		350		350	ns
t_{SIB}	STB = 0 to IBF = 1[1]		300		300	ns
t_{RIB}	RD = 1 to IBF = 0[1]		300		300	ns
t_{RIT}	RD = 0 to INTR = 0[1]		400		400	ns
t_{SIT}	STB = 1 to INTR = 1[1]		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1[1]		350		350	ns
t_{WIT}	WR = 0 to INTR = 0[1]		850		850	ns

Notes: 1. Test Conditions: 8255A: C_L = 100pF; 8255A-5: C_L = 150pF.
2. Period of Reset pulse must be at least 50μs during or after power on. Subsequent Reset pulse can be 500 ns min.

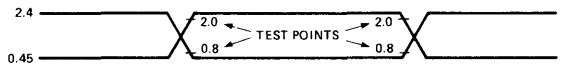


Figure 25. Input Waveforms for A.C. Tests

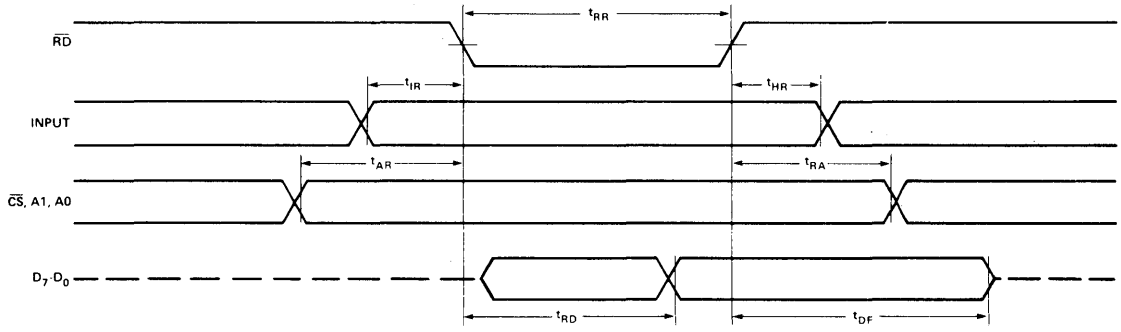


Figure 26. MODE 0 (Basic Input)

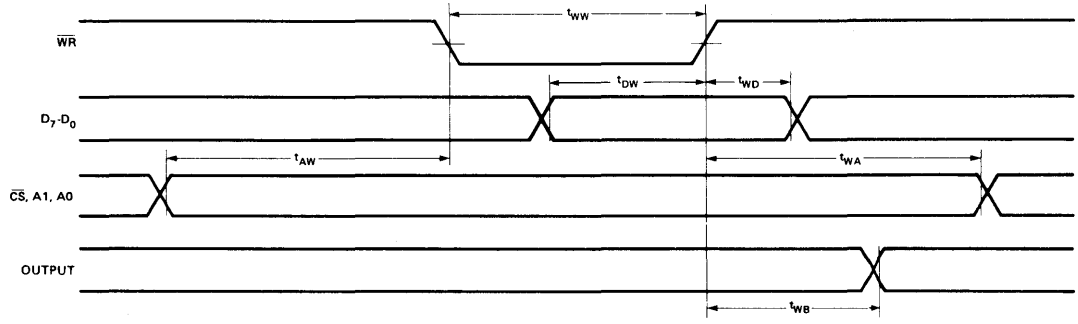


Figure 27. MODE 0 (Basic Output)

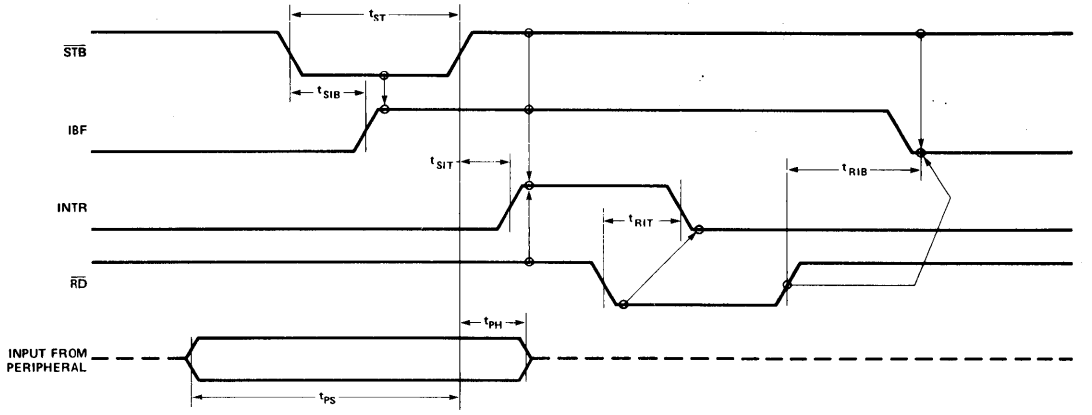


Figure 28. MODE 1 (Strobed Inut)

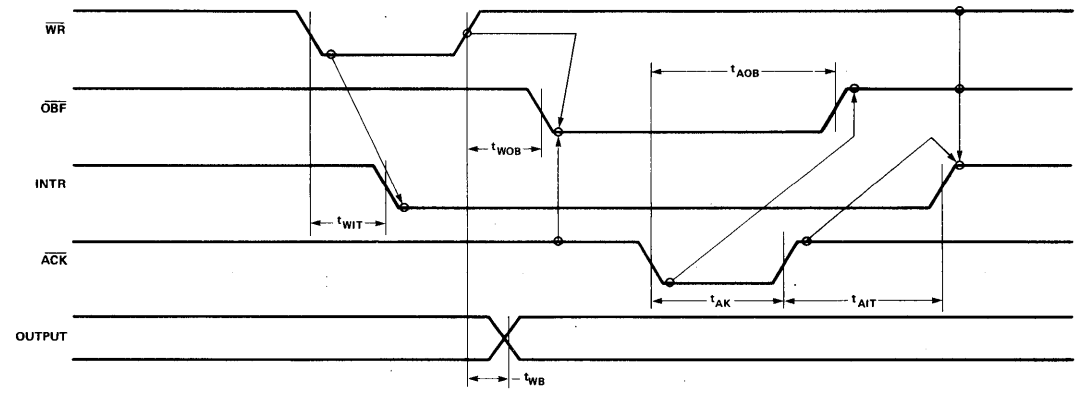


Figure 29. MODE 1 (Strobed Output)

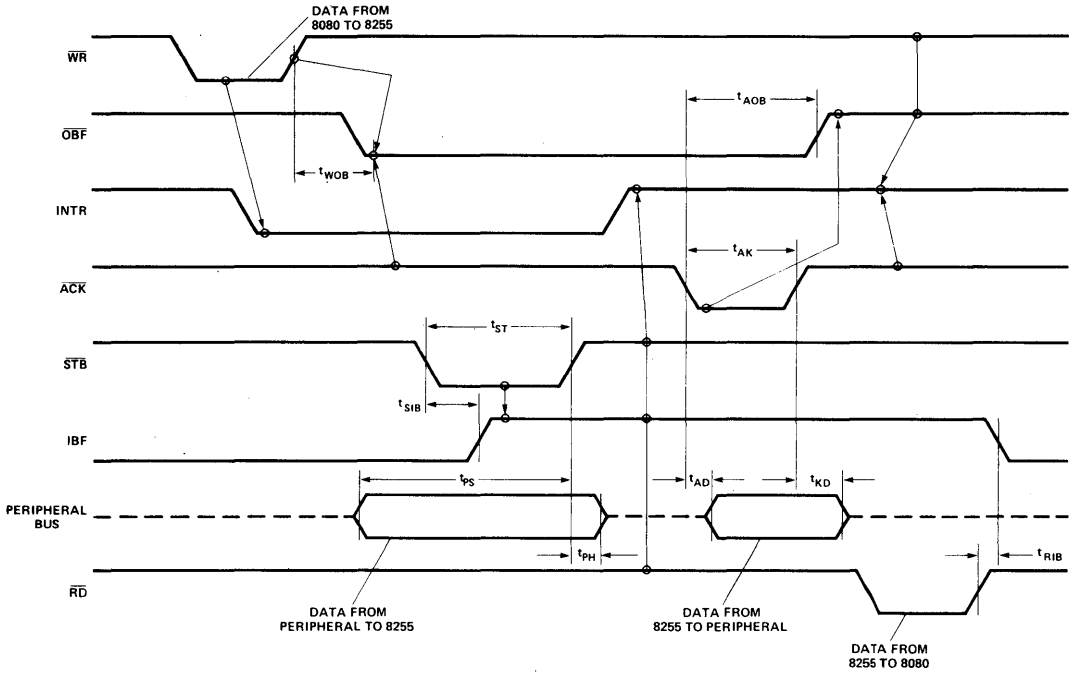


Figure 30. MODE 2 (Bidirectional)

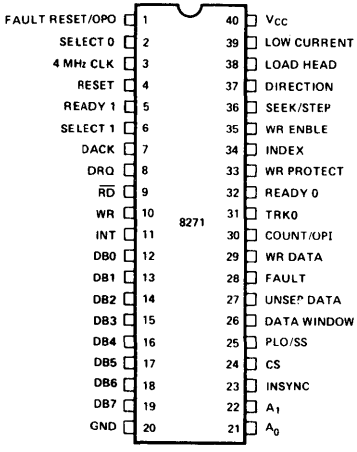
NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

8271/8271-6/8271-8
PROGRAMMABLE FLOPPY DISK CONTROLLER

- **IBM 3740 Soft Sectored Format Compatible**
- **Programmable Record Lengths**
- **Multi-Sector Capability**
- **Maintain Dual Drives with Minimum Software Overhead Expandable to 4 Drives**
- **Automatic Read/Write Head Positioning and Verification**
- **Internal CRC Generation and Checking**
- **Programmable Step Rate, Settle-Time, Head Load Time, Head Unload Index Count**
- **Fully MCS-80™ and MCS-85™ Compatible**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8271 Programmable Floppy Disk Controller (FDC) is an LSI component designed to interface one to 4 floppy disk drives to an 8-bit microcomputer system. Its powerful control functions minimize both hardware and software overhead normally associated with floppy disk controllers.

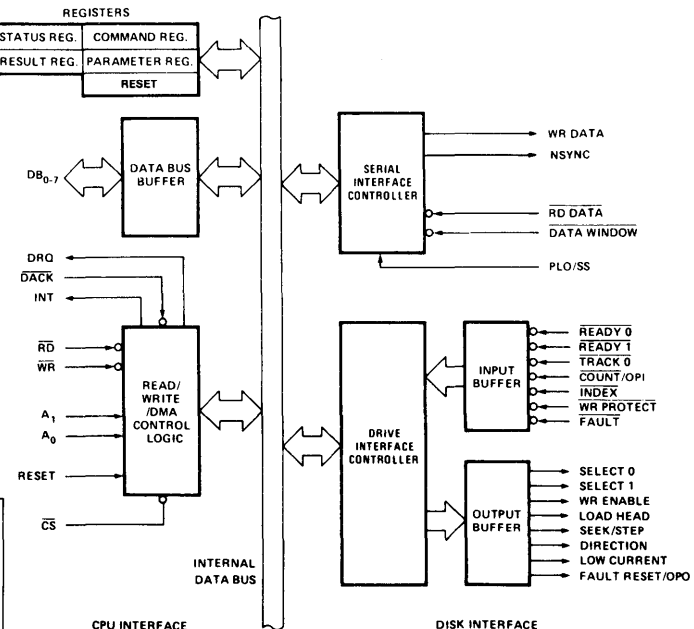
PIN CONFIGURATION



PIN NAMES

$DB_7 - DB_0$	DATA BUS (BI-DIRECTIONAL)	PLO/SS	PLO/SINGLE SHOT
CLK	CLOCK INPUT (TTL)	DATA WINDOW	DATA WINDOW
SELECT 1, 0	SELECT 1, 0	UNSEP DATA	UNSEPARATED DATA
FAULT RESET/OPO	FAULT RESET/OPTIONAL OUTPUT	FAULT	FAULT
RESET	CHIP RESET	WR DATA	WRITE DATA
READY 1, 0	READY 1, 0	COUNT/OPI	COUNT/OPTIONAL INPUT
DACK	DMA ACKNOWLEDGE	TRK 0	TRACK 0
DRQ	DMA REQUEST	WR PROTECT	WRITE PROTECT
RD	CPU READ INPUT	INDEX	INDEX
WR	CPU WRITE INPUT	WR ENABLE	WRITE ENABLE
INT	INTERRUPT	SEEK/STEP	SEEK/STEP
$A_1, 0$	REGISTER SELECT	DIRECTION	DIRECTION
INSYNC	READ DATA INSYNC	LOAD HEAD	LOAD HEAD
CS	CHIP SELECT	LOW CURRENT	LOW CURRENT

BLOCK DIAGRAM



8271 BASIC FUNCTIONAL DESCRIPTION

General

The 8271 Floppy Disk Controller (FDC) interfaces either two single or one dual floppy drive to an eight bit microprocessor and is fully compatible with Intel's new high performance MCS-85 microcomputer system. With minimum external circuitry, this innovative controller supports most standard, commonly-available flexible disk drives including the mini-floppy.

The 8271 FDC supports a comprehensive soft sectored format which is IBM 3740 compatible and includes provision for the designating and handling of bad tracks. It is a high level controller that relieves the CPU (and user) of many of the control tasks associated with implementing a floppy disk interface. The FDC supports a variety of high level instructions which allow the user to store and retrieve data on a floppy disk without dealing with the low level details of disk operation.

In addition to the standard read/write commands, a scan command is supported. The scan command allows the user program to specify a data pattern and instructs the FDC to search for that pattern on a track. Any application that is required to search the disk for information (such as point of sale price lookup, disk directory search, etc.), may use the scan command to reduce the CPU overhead. Once the scan operation is initiated, no CPU intervention is required.

Hardware Description

The 8271 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Pin Name	Pin No.	I/O	Description
V_{CC}	(40)		+5V supply
GND	(20)		Ground
Clock	(3)	I	A square wave clock
Reset	(4)	I	A high signal on the reset input forces the 8271 to an idle state. The 8271 remains idle until a command is issued by the CPU. The output signals of the drive interface are forced inactive (LOW). Reset must be active for 10 or more clock cycles.
$\overline{CS}$	(24)	I	The I/O Read and I/O Write inputs are enabled by the chip select signal.
DB_7-DB_0	(19-12)	I/O	The Data Bus lines are bidirectional, three-state lines (8080 data bus compatible).
$\overline{WR}$	(10)	I	The Write signal is used to signal the control logic that a transfer of data from the data bus to the 8271 is required.
$\overline{RD}$	(9)	I	The Read signal is used to signal the control logic that a transfer of data from the 8271 to the data bus is required.
INT	(11)	O	The interrupt signal indicates that the 8271 requires service.
A_1-A_0	(22-21)	I	These two lines are CPU Interface Register select lines.
DRQ	(8)	O	The DMA request signal is used to request a transfer of data between the 8271 and memory.
$\overline{DACK}$	(7)	I	The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted. For non-DMA transfers, this signal should be driven in the manner of a "Chip Select".
Select 1- Select 0	(6) (2)	O	These lines are used to specify the selected drive. These lines are set by the command byte.
Fault Reset/ OPO	(1)	O	The optional fault reset output line is used to reset an error condition which is latched by the drive. If this line is not used for a fault reset it can be used as an optional output line. This line is set with the write special register command.
Write Enable	(35)	O	This signal enables the drive write logic.
Seek/Step	(36)	O	This multi-function line is used during drive seeks.
Direction	(37)	O	The direction line specifies the seek direction. A high level on this pin steps the R/W head toward the spindle (step-in), a low level steps the head away from the spindle (step-out).
Load Head	(38)	O	The load head line causes the drive to load the Read/Write head against the diskette.
Low Current	(39)	O	This line notifies the drive that track 43 or greater is selected.
$\overline{Ready\ 1}$, $\overline{Ready\ 0}$	(5) (32)	I	These two lines indicate that the specified drive is ready.
$\overline{Fault}$	(28)	I	This line is used by the drive to specify a file unsafe condition.
$\overline{Count/OPI}$	(30)	I	If the optional seek/direction/count seek mode is selected, the count pin receives pulses to step the R/W head to the desired track. Otherwise, this line can be used as an optional input.
$\overline{Write\ Protect}$	(33)	I	This signal specifies that the diskette inserted is write protected.
$\overline{TRK0}$	(31)	I	This signal indicates when the R/W head is positioned over track zero.
$\overline{Index}$	(34)	I	The index signal gives an indication of the relative position of the diskette.
PLO/SS	(25)	I	This pin is used to specify the type of data separator used. Phase-Locked Oscillator/Single Shot.
Write Data	(29)	O	Composite write data.

Pin Name	Pin No.	I/O	Description
$\overline{\text{Unseparated}}$ $\overline{\text{Data}}$	(27)	I	This input is the unseparated data and clocks.
$\overline{\text{Data Window}}$	(26)	I	This is a data window established by a single-shot or phase-locked oscillator data separator.
INSYNC	(23)	O	This line is high when 8271 has attained input data synchronization, by detecting 2 bytes of zeros followed by an expected Address Mark. It will stay high until the end of the ID or data field.

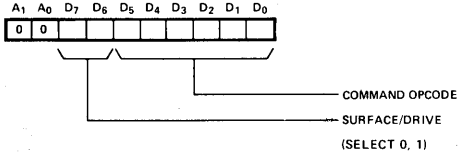
CPU Interface Description

This interface minimizes CPU involvement by supporting a set of high level commands and both DMA and non-DMA type data transfers and by providing hierarchical status information regarding the result of command execution.

The CPU utilizes the control interface (see the Block diagram) to specify the FDC commands and to determine the result of an executed command. This interface is supported by five Registers which are addressed by the CPU via the A_1, A_0, $\overline{RD}$ and $\overline{WR}$ signals. If an 8080 based system is used, the $\overline{RD}$ and $\overline{WR}$ signals can be driven by the 8228's $\overline{I/OR}$ and $\overline{I/OW}$ signals. The registers are defined as follows:

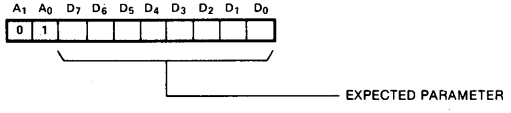
Command Register

The CPU loads an appropriate command into the Command Register which has the following format:



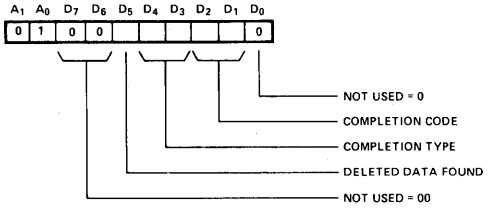
Parameter Register

Accepts parameters of commands that require further description; up to five parameters may be required, example:



Result Register

The Result Register is used to supply the outcome of FDC command execution (such as a good/bad completion) to the CPU. The standard Result byte format is:



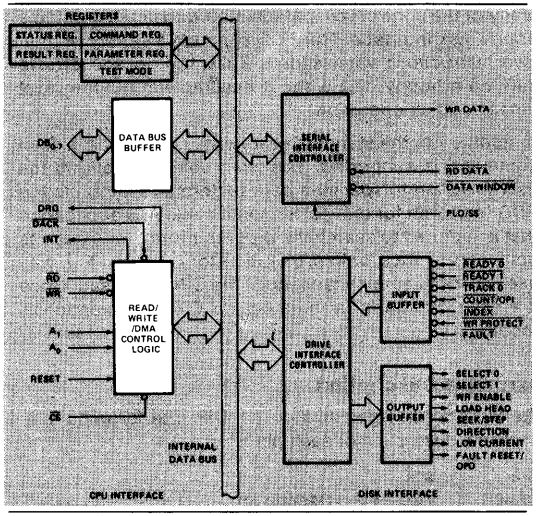
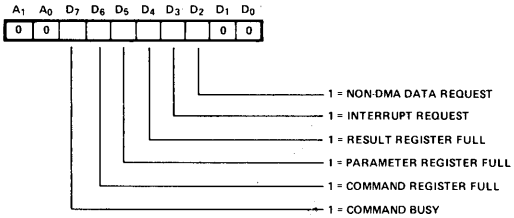


Figure 1. 8271 Block Diagram Showing CPU Interface Functions

Status Register

Reflects the state of the FDC.



Reset Register

Allows the 8271 to be reset by the program. Reset must be active for 11 or more chip clocks.

INT (Interrupt Line)

Another element of the control interface is the Interrupt line (INT). This line is used to signal the CPU that an FDC operation has been completed. It remains active until the result register is read.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias.....0°C to 70°C[1]
Storage Temperature.....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground.....−0.5V to +7V
Power Dissipation.....1 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

V_{CC} = +5.0V ±5%
8721 and 8271-8: T_A = 0°C to 70°C; 8271-6: T_A = 0°C to 50°C

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	(V_{CC} + 0.5)	V	
V_{OLD}	Output Low Voltage (Data Bus)		0.45	V	I_{OL} = 2.0 mA
V_{OLI}	Output Low Voltage (Interface Pins)		0.5	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −220 μA
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OZ}	Off-State Output Current		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		180	mA	

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	t_c = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

NOTE: 1. Ambient temperature under bias for 8271-6 is 0°C to 50°C.

A.C. CHARACTERISTICS

$V_{CC} = +5.0V \pm 5\%$
8271 and 8271-8: $T_A = 0°C$ to $70°C$; 8271-6: $T_A = 0°C$ to $50°C$

Read Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{RD}$	0		ns	Note 2
t_{CA}	Select Hold from $\overline{RD}$	0		ns	Note 2
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	Data Delay from Address		250	ns	Note 2
t_{RD}	Data Delay from $\overline{RD}$		150	ns	$C_L = 150$ pF, Note 2
t_{DF}	Output Float Delay	20	100	ns	$C_L = 20$ pF for Minimum; 150 pF for Maximum
t_{DC}	DACK Setup to $\overline{RD}$	25		ns	
t_{CD}	DACK Hold from $\overline{RD}$	25		ns	
t_{KD}	Data Delay from DACK		250	ns	

Write Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{WR}$	0		ns	
t_{CA}	Select Hold from $\overline{WR}$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$	150		ns	
t_{WD}	Data Hold from $\overline{WR}$	0		ns	
t_{DC}	DACK Setup to $\overline{WR}$	25		ns	
t_{CD}	DACK Hold from $\overline{WR}$	25		ns	

DMA

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CQ}	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		150	ns	

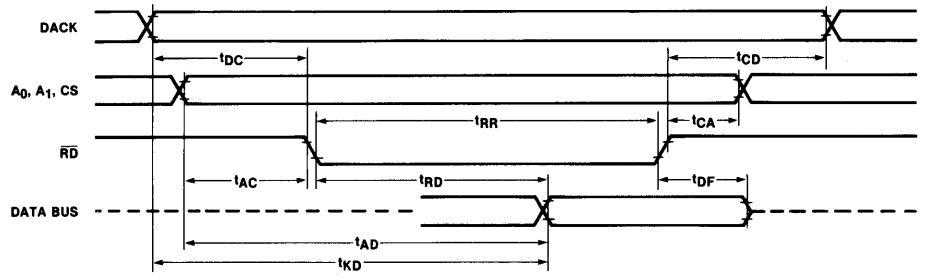
Other Timing

Symbol	Parameter	8271/8271-6		8271-8		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
t_{RSTW}	Reset Pulse Width	10		10		t_{CY}	
t_r	Input Signal Rise Time		20		20	ns	
t_f	Input Signal Fall Time		20		20	ns	
t_{RSTS}	Reset to First IOWR	2		2		t_{CY}	
t_{CY}	Clock Period	250		500			Note 3
t_{CL}	Clock Low Period	110		215		ns	
t_{CH}	Clock High Period	125		250		ns	
t_{DS}	Data Window Setup to Unseparated Clock and Data	50		50		ns	
t_{DH}	Data Window Hold from Unseparated Clock and Data	0		0		ns	

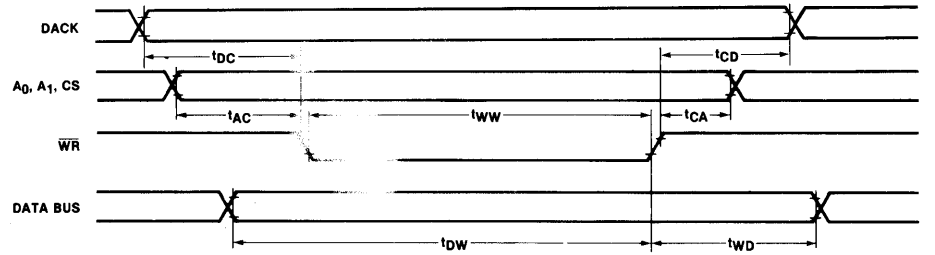
NOTES:
1. All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V
   Output "1" at 2.0V, "0" at 0.8V
2. t_{AD}, t_{RD}, t_{AC}, and t_{CA} are not concurrent specs.
3. Standard Floppy: $t_{CY} = 250$ ns $\pm 0.4\%$ Mini-Floppy: $t_{CY} = 500$ ns $\pm 0.4\%$

WAVEFORMS

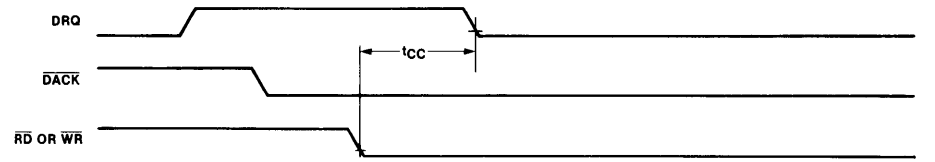
Read Waveforms

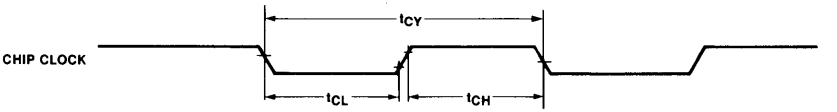


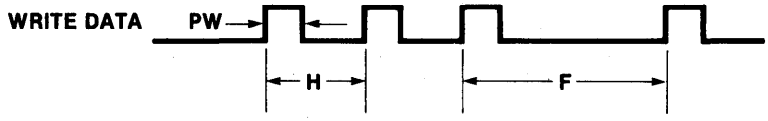
Write Waveforms



DMA Waveforms



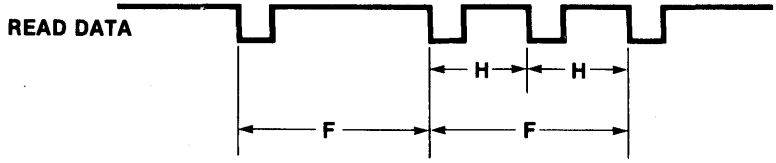




PULSE WIDTH PW = t_{CY} ±30 ns
H (HALF BIT CELL) = 8 t_{CY}
F (FULL BIT CELL) = 16 t_{CY}

*t_{CY} =	** t_{CY} =
250 ns ±0.4%	500 ns ±0.4%
250 ns ±30 ns	500 ns ±30 ns
2.0 μs ± 8 ns	4.0 μs ±16 ns
4.0 μs ±16 ns	8.0 μs ±32 ns

Figure 24. Write Data



*t_{CY} = 250 ns **t_{CY} = 500 ns

F = 16 t_{CY} ±8 t_{CY}
H = 8 t_{CY} ±4 t_{CY}

Figure 25. Read Data

*STANDARD FLEXIBLE DISK DRIVE TIMING
**MINI-FLOPPY TIMING

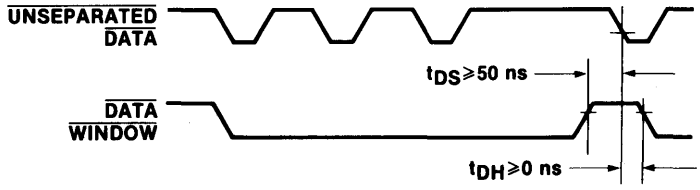
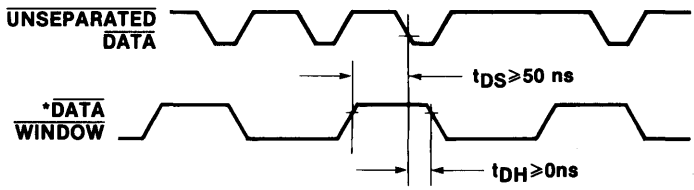


Figure 26. Single-Shot Data Separator



*DATA WINDOW MAY BE 180° OUT OF PHASE IN PLO DATA SEPARATION MODE.

Figure 27. PLO Data Separator

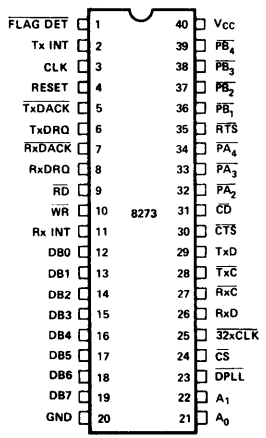
PRELIMINARY

8273, 8273-4, 8273-8
PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

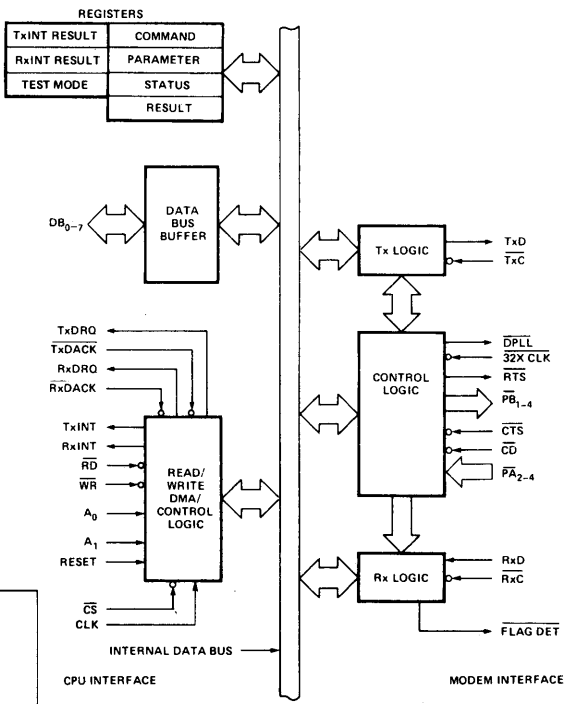
- **CCITT X.25 Compatible**
- **HDLC/SDLC Compatible**
- **Full Duplex, Half Duplex, or Loop SDLC Operation**
- **Up to 64K Baud Synchronous Transfers**
- **Automatic FCS (CRC) Generation and Checking**
- **Up to 9.6K Baud with On-Board Phase Locked Loop**
- **Programmable NRZI Encode/Decode**
- **Two User Programmable Modem Control Ports**
- **Digital Phase Locked Loop Clock Recovery**
- **Minimum CPU Overhead**
- **Fully Compatible with 8048/8080/8085/ 8088/8086 CPUs**
- **Single +5V Supply**

The Intel® 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/ CCITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS-88/86™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

DB0–DB7	DATA BUS (8 BITS)	$\overline{CS}$	CHIP SELECT
$\overline{FLAG\ DET}$	FLAG DETECT	$\overline{32xCLK}$	32 TIMES CLOCK
TxINT	TRANSMITTER INTERRUPT	Rx D	RECEIVER DATA
CLK	CLOCK INPUT	$\overline{Rx\ C}$	RECEIVER CLOCK
$\overline{RESET}$	RESET	$\overline{Tx\ C}$	TRANSMITTER CLOCK
$\overline{Tx\ DACK}$	TRANSMITTER DMA ACKNOWLEDGE	Tx D	TRANSMITTER DATA
TxDRQ	TRANSMITTER DMA REQUEST	$\overline{CTS}$	CLEAR TO SEND
$\overline{RD}$	READ INPUT	$\overline{CD}$	CARRIER DETECT
$\overline{WR}$	WRITE INPUT	$\overline{PA_2}$–$\overline{PA_4}$	GP INPUT PORTS
$\overline{Rx\ DACK}$	RECEIVER DMA ACKNOWLEDGE	$\overline{PB_1}$–$\overline{PB_4}$	GP OUTPUT PORTS
Rx DRQ	RECEIVER DMA REQUEST	$\overline{RTS}$	REQUEST TO SEND
Rx INT	RECEIVER INTERRUPT	V_{CC}	+5 VOLT SUPPLY
A0–A1	COMMAND REGISTER SELECT ADDRESS	GND	GROUND
$\overline{DPLL}$	DIGITAL PHASE LOCKED LOOP		

A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by International Standards Organization (ISO). HDLC is the discipline used to implement ISO X.25 packet switching systems.

The Synchronous Data Link Control (SDLC) is an IBM communication link protocol used to implement the System Network Architecture (SNA). Both the protocols are bit oriented, code independent, and ideal for full duplex communication. Some common applications include terminal to terminal, terminal to CPU, CPU to CPU, satellite communication, packet switching and other high speed data links. In systems which require expensive cabling and interconnect hardware, any of the two protocols could be used to simplify interfacing (by going serial), thereby reducing interconnect hardware costs. Since both the protocols are speed independent, reducing interconnect hardware could become an important application.

Network

In both the HDLC and SDLC line protocols, according to a pre-assigned hierarchy, a PRIMARY (Control) STATION controls the overall network (data link) and issues commands to the SECONDARY (Slave) STATIONS. The latter comply with instructions and respond by sending appropriate RESPONSES. Whenever a transmitting station must end transmission prematurely it sends an ABORT character. Upon detecting an abort character, a receiving station ignores the transmission block called a FRAME. Time fill between frames can be accomplished by transmitting either continuous frame preambles called FLAGS or an abort character. A time fill within a frame is not permitted. Whenever a station receives a string of more that fifteen consecutive ones, the station goes into an IDLE state.

Frames

A single communication element is called a FRAME which can be used for both Link Control and data transfer purposes. The elements of a frame are the beginning eight bit FLAG (F) consisting of one zero, six ones, and a zero, an eight bit ADDRESS FIELD (A), an eight bit CONTROL FIELD (C), a variable (N-bit) INFORMATION FIELD (I), a sixteen bit FRAME CHECK SEQUENCE (FCS), and an eight bit end FLAG (F), having the same bit pattern as the beginning flag. In HDLC the Address (A) and Control (C) bytes are extendable. The HDLC and the SDLC use three types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Non-sequenced Frame is used for initialization and control of the secondary stations.

Frame Characteristics

An important characteristic of a frame is that its contents are made code transparent by use of a zero bit insertion and deletion technique. Thus, the user can adopt any format or code suitable for his system — it may even be a computer word length or a "memory dump". The frame is bit oriented that is, bits, not characters in each field, have specific meanings. The Frame Check Sequence (FCS) is an error detection scheme similar to the Cyclic Redundancy Checkword (CRC) widely used in magnetic disk storage devices. The Command and Response information frames contain sequence numbers in the control fields identifying the sent and received frames. The sequence numbers are used in Error Recovery Procedures (ERP) and as implicit acknowledgement of frame communication, enhancing the true full-duplex nature of the HDLC/SDLC protocols.

In contrast, BISYNC is basically half-duplex (two way alternate) because of necessity to transmit immediate acknowledgement frames. HDLC/SDLC therefore saves propagation delay times and have a potential of twice the throughput rate of BISYNC.

It is possible to use HDLC or SDLC over half duplex lines but there is a corresponding loss in throughput because both are primarily designed for full-duplex communication. As in any synchronous system, the bit rate is determined by the clock bits supplied by the modem, protocols themselves are speed independent.

A byproduct of the use of zero-bit insertion-deletion technique is the non-return-to-zero invert (NRZI) data transmission/reception compatibility. The latter allows HDLC/SDLC protocols to be used with asynchronous data communication hardware in which the clocks are derived from the NRZI encoded data.

References

IBM Synchronous Data Link Control General Information, IBM, GA27-3093-1.

Standard Network Access Protocol Specification, DATAPAC, Trans-Canada Telephone System CCG111

Recommendation X.25, ISO/CCITT March 2, 1976.

IBM 3650 Retail Store System Loop Interface OEM Information, IBM, GA 27-3098-0

Guidebook to Data Communications, Training Manual, Hewlett-Packard 5955-1715

IBM Introduction to Teleprocessing, IBM, GC 20-8095-02

System Network Architecture, Technical Overview, IBM, GA 27-3102

System Network Architecture Format and Protocol, IBM GA 27-3112

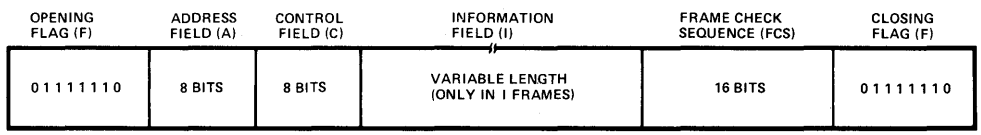


Figure 1. Frame Format

FUNCTIONAL DESCRIPTION

General

The Intel® 8273 HDLC/SDLC controller is a microcomputer peripheral device which supports the International Standards Organization (ISO) High Level Data Link Control (HDLC), and IBM Synchronous Data Link Control (SDLC) communications protocols. This controller minimizes CPU software by supporting a comprehensive frame-level instruction set and by hardware implementation of the low level tasks associated with frame assembly/disassembly and data integrity. The 8273 can be used in either synchronous or asynchronous applications. In asynchronous applications the data can be programmed to be encoded/decoded in NRZI code. The clock is derived from the NRZI data using a digital phase locked loop. The data transparency is achieved by using a zero-bit insertion/deletion technique. The frames are automatically checked for errors during reception by verifying the Frame Check Sequence (FCS); the FCS is automatically generated and appended before the final flag in transmit.

The 8273 recognizes and can generate flags (01111110), Abort, Idle, and GA (EOP) characters.

The 8273 can assume either a primary (control) or a secondary (slave) role. It can therefore be readily implemented in an SDLC loop configuration as typified by the IBM 3650 Retail Store System by programming the 8273 into a one-bit delay mode. In such a configuration, a two wire pair can be effectively used for data transfer between controllers and loop stations. The digital phase locked loop output pin can be used by the loop station without the presence of an accurate Tx clock.

Hardware Description

The 8273 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Pin Name (No.)	I/O	Description
V_{CC} (40)		+5V Supply
GND (20)		Ground
RESET (4)	I	A high signal on this pin will force the 8273 to an idle state. The 8273 will remain idle until a command is issued by the CPU. The modem interface output signals are forced high. Reset must be true for a minimum of 10 TCY.
$\overline{CS}$ (24)	I	The $\overline{RD}$ and $\overline{WR}$ inputs are enabled by the chip select input.
DB_7-DB_0 (19-12)	I/O	The Data Bus lines are bidirectional three-state lines which interface with the system Data Bus.
$\overline{WR}$ (10)	I	The Write signal is used to control the transfer of either a command or data from CPU to the 8273.
$\overline{RD}$ (9)	I	The Read signal is used to control the transfer of either a data byte or a status word from the 8273 to the CPU.
TxINT (2)	O	The Transmitter interrupt signal indicates that the transmitter logic requires service.
RxINT (11)	O	The Receiver interrupt signal indicates that the Receiver logic requires service.
TxDRQ (6)	O	Requests a transfer of data between memory and the 8273 for a transmit operation.
RxRDQ (8)	O	Requests a transfer of data between the 8273 and memory for a receive operation.
$\overline{TxDACK}$ (5)	I	The Transmitter DMA acknowledge signal notifies the 8273 that the TxDMA cycle has been granted.
$\overline{RxDACK}$ (7)	I	The Receiver DMA acknowledge signal notifies the 8273 that the RxDMA cycle has been granted.
A_1-A_0 (22-21)	I	These two lines are CPU Interface Register Select lines.
TxD (29)	O	This line transmits the serial data to the communication channel.
$\overline{TxC}$ (28)	I	The transmitter clock is used to synchronize the transmit data.
RxD (26)	I	This line receives serial data from the communication channel.
$\overline{RxC}$ (27)	I	The Receiver Clock is used to synchronize the receive data.
$\overline{32X\ CLK}$ (25)	I	The 32X clock is used to provide clock recovery when an asynchronous modem is used. In loop configuration the loop station can run without an accurate 1X clock by using the 32X CLK in conjunction with the DPLL output. (This pin must be grounded when not used).
$\overline{DPLL}$ (23)	O	Digital Phase Locked Loop output can be tied to RxC and/or TxC when 1X clock is not available. DPLL is used with 32X CLK.
$\overline{FLAG\ DET}$ (1)	O	Flag Detect signals that a flag (01111110) has been received by an active receiver.
$\overline{RTS}$ (35)	O	Request to Send signals that the 8273 is ready to transmit data.
$\overline{CTS}$ (30)	I	Clear to Send signals that the modem is ready to accept data from the 8273.
$\overline{CD}$ (31)	I	Carrier Detect signals that the line transmission has started and the 8273 may begin to sample data on RxD line.
$\overline{PA_{2-4}}$ (32-34)	I	General purpose input ports. The logic levels on these lines can be Read by the CPU through the Data Bus Buffer.
$\overline{PB_{1-4}}$ (36-39)	O	General purpose output ports. The CPU can write these output lines through Data Bus Buffer.
CLK (3)	I	A square wave TTL clock.

ABSOLUTE MAXIMUM RATINGS*

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 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (8273, 8273-4, 8273-8)

$T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
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		0.45	V	$I_{OL} = 2.0mA$ for Data Bus Pins $I_{OL} = 1.0mA$ for Output Port Pins $I_{OL} = 1.6mA$ for All Other Pins
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -200\mu A$ for Data Bus Pins $I_{OH} = -100\mu A$ for All Other Pins
I_{IL}	Input Load Current		±10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OZ}	Off-State Output Current		±10	μA	$V_{OUT} = V_{CC}$ to 0V
I_{CC}	V_{CC} Supply Current		180	mA	

CAPACITANCE (8273, 8273-4, 8273-8)

$T_A = 25°C$, $V_{CC} = GND = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	$t_c = 1MHz$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 5\%$

Clock Timing (8273)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{CY}	Clock	250			ns	64K Baud Max Operating Rate
t_{CL}	Clock Low	120			ns	
t_{CH}	Clock High	120			ns	

Clock Timing (8273-4)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{CY}	Clock	286			ns	56K Baud Max Operating Rate
t_{CL}	Clock Low	135			ns	
t_{CH}	Clock High	135			ns	

Clock Timing (8273-8)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{CY}	Clock	330			ns	48K Baud Max Operating Rate
t_{CL}	Clock Low	150			ns	
t_{CH}	Clock High	150			ns	

A.C. CHARACTERISTICS (8273, 8273-4, 8273-8) $T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 5\%$

Read Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{RD}$	0		ns	Note 3
t_{CA}	Select Hold from $\overline{RD}$	0		ns	Note 3
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	Data Delay from Address		300	ns	Note 3
t_{RD}	Data Delay from $\overline{RD}$		200	ns	$C_L = 150pF$, Note 3
t_{DF}	Output Float Delay	20	100	ns	$C_L = 20pF$ for Minimum; 150pF for Maximum
t_{DC}	DACK Setup to $\overline{RD}$	25		ns	
t_{CD}	DACK Hold from $\overline{RD}$	25		ns	
t_{KD}	Data Delay from DACK		300	ns	

Write Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{WR}$	0		ns	
t_{CA}	Select Hold from $\overline{WR}$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$	150		ns	
t_{WD}	Data Hold from $\overline{WR}$	0		ns	
t_{DC}	DACK Setup to $\overline{WR}$	25		ns	
t_{CD}	DACK Hold from $\overline{WR}$	25		ns	

DMA

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CQ}	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		200	ns	

Other Timing

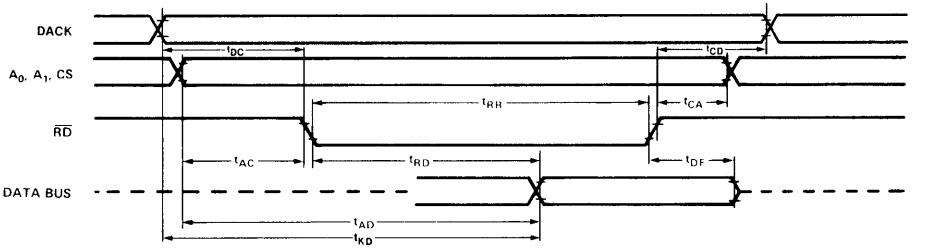
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{RSTW}	Reset Pulse Width	10		t_{CY}	
t_r	Input Signal Rise Time		20	ns	
t_f	Input Signal Fall Time		20	ns	
t_{RSTS}	Reset to First $\overline{IOWR}$	2		t_{CY}	
t_{CY32}	32X Clock Cycle Time	$9.7 \cdot t_{CY}$		ns	
t_{CL32}	32X Clock Low Time	$4 \cdot t_{CY}$		ns	
t_{CH32}	32X Clock High Time	$4 \cdot t_{CY}$		ns	
t_{DPLL}	$\overline{DPLL}$ Output Low	$1 \cdot t_{CY} - 50$		ns	
t_{DCL}	Data Clock Low	$1 \cdot t_{CY} - 50$		ns	
t_{DCH}	Data Clock High	$2 \cdot t_{CY}$		ns	
t_{DCY}	Data Clock	$62.5 \cdot t_{CY}$		ns	
t_{TD}	Transmit Data Delay		200	ns	
t_{DS}	Data Setup Time	200		ns	
t_{DH}	Data Hold Time	100		ns	
t_{FLD}	$\overline{FLAG\ DET}$ Output Low	$8 \cdot t_{CY} \pm 50$		ns	

NOTES:

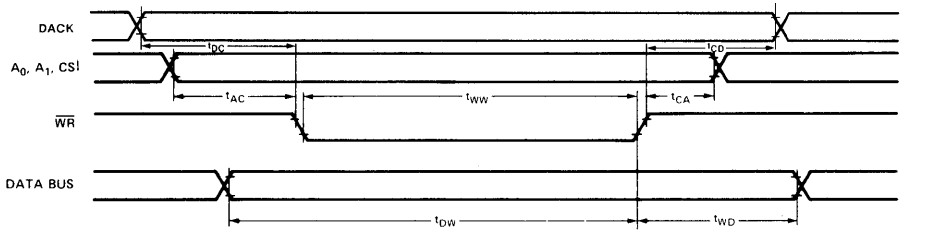
1. All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V; Output "1" at 2.0V, "0" at 0.8V.
2. t_{AD}, t_{RD}, t_{AC}, and t_{CA} are not concurrent specs.

WAVEFORMS

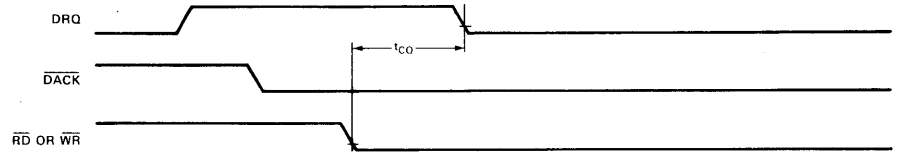
Read Waveforms

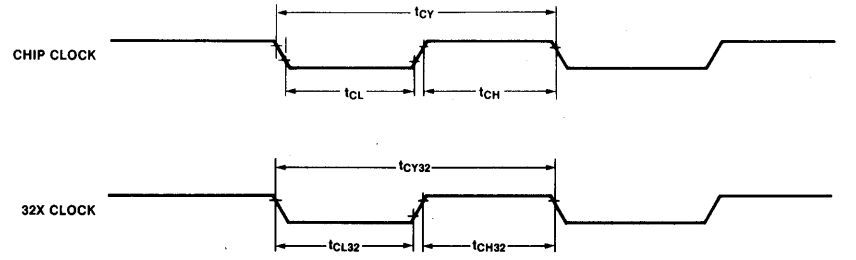


Write Waveforms

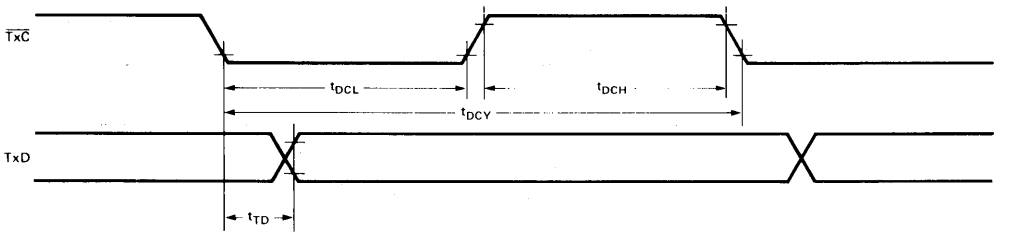


DMA Waveforms

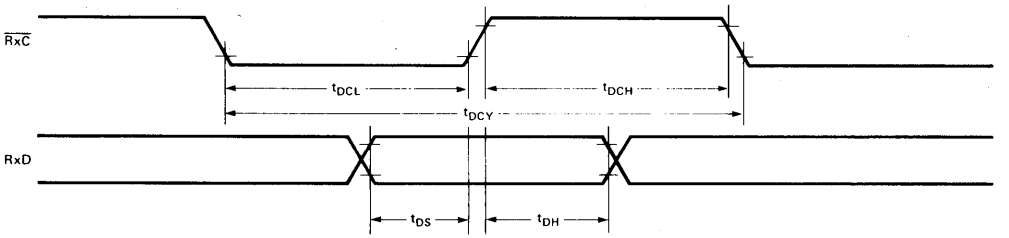




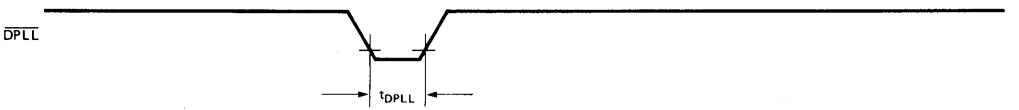
Transmit Data Waveforms



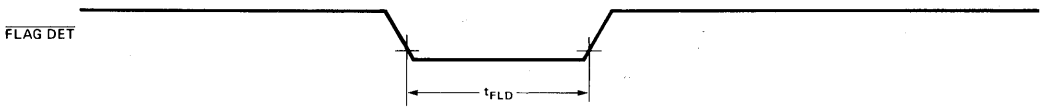
Receive Data Waveforms



DPLL Output Waveform



Flag Detect Output Waveform

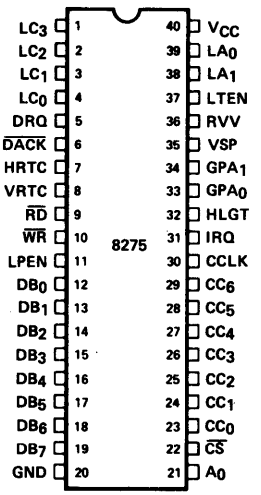


8275
PROGRAMMABLE CRT CONTROLLER

- **Programmable Screen and Character Format**
- **6 Independent Visual Field Attributes**
- **11 Visual Character Attributes (Graphic Capability)**
- **Cursor Control (4 Types)**
- **Light Pen Detection and Registers**
- **Fully MCS-80™ and MCS-85™ Compatible**
- **Dual Row Buffers**
- **Programmable DMA Burst Mode**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8275 Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8275 will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

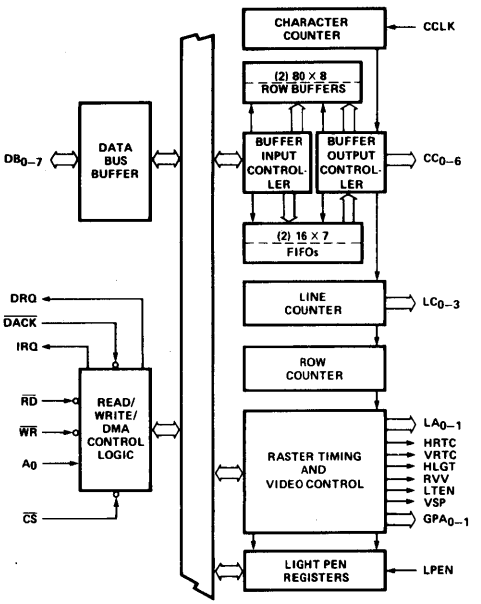
PIN CONFIGURATION



PIN NAMES

DB_{0-1}	BI-DIRECTIONAL DATA BUS	LC_{0-3}	LINE COUNTER OUTPUTS
DRQ	DMA REQUEST OUTPUT	LA_{0-1}	LINE ATTRIBUTE OUTPUTS
$\overline{DACK}$	DMA ACKNOWLEDGE INPUT	HRTC	HORIZONTAL RETRACE OUTPUT
IRQ	INTERRUPT REQUEST OUTPUT	VRTC	VERTICAL RETRACE OUTPUT
$\overline{RD}$	READ STROBE INPUT	HLGT	HIGHLIGHT OUTPUT
$\overline{WR}$	WRITE STROBE INPUT	RVV	REVERSE VIDEO OUTPUT
A_0	REGISTER ADDRESS INPUT	LTEN	LIGHT ENABLE OUTPUT
CS	CHIP SELECT INPUT	VSP	VIDEO SUPPRESS OUTPUT
CCLK	CHARACTER CLOCK INPUT	GPA_{0-1}	GENERAL PURPOSE ATTRIBUTE OUTPUTS
CC_{0-6}	CHARACTER CODE OUTPUTS	LPEN	LIGHT PEN INPUT

BLOCK DIAGRAM



00224A

PIN DESCRIPTIONS

Pin #	Pin Name	I/O	Pin Description
1	LC_3	O	Line count. Output from the line counter which is used to address the character generator for the line positions on the screen.
2	LC_2		
3	LC_1		
4	LC_0		
5	DRQ	O	DMA request. Output signal to the 8257 DMA controller requesting a DMA cycle.
6	$\overline{DACK}$	I	DMA acknowledge. Input signal from the 8257 DMA controller acknowledging that the requested DMA cycle has been granted.
7	HRTC	O	Horizontal retrace. Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.
8	VRTC	O	Vertical retrace. Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.
9	$\overline{RD}$	I	Read input. A control signal to read registers.
10	$\overline{WR}$	I	Write input. A control signal to write commands into the control registers or write data into the row buffers during a DMA cycle.
11	LPEN	I	Light pen. Input signal from the CRT system signifying that a light pen signal has been detected.
12	DB_0	I/O	Bi-directional three-state data bus lines. The outputs are enabled during a read of the C or P ports.
13	DB_1		
14	DB_2		
15	DB_3		
16	DB_4		
17	DB_5		
18	DB_6		
19	DB_7		
20	Ground		Ground
40	V_{CC}		+5V power supply
39	LA_0	O	Line attribute codes. These attribute codes have to be decoded externally by the dot/timing logic to generate the horizontal and vertical line combinations for the graphic displays specified by the character attribute codes.
38	LA_1		
37	LTEN	O	Light enable. Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
36	RVV	O	Reverse video. Output signal used to indicate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
35	VSP	O	Video suppression. Output signal used to blank the video signal to the CRT. This output is active: — during the horizontal and vertical retrace intervals. — at the top and bottom lines of rows if underline is programmed to be number 8 or greater. — when an end of row or end of screen code is detected. — When a DMA underrun occurs. — at regular intervals (1/16 frame frequency for cursor, 1/32 frame frequency for character and field attributes) — to create blinking displays as specified by cursor, character attribute, or field attribute programming.
34	GPA_1	O	General purpose attribute codes. Outputs which are enabled by the general purpose field attribute codes.
33	GPA_0		
32	HLGT	O	Highlight. Output signal used to intensify the display at particular positions on the screen as specified by the character attribute codes or field attribute codes.
31	IRQ	O	Interrupt request.
30	CCLK	I	Character clock (from dot/timing logic).
29	CC_6	O	Character codes. Output from the row buffers used for character selection in the character generator.
28	CC_5		
27	CC_4		
26	CC_3		
25	CC_2		
24	CC_1		
23	CC_0		
22	$\overline{CS}$	I	Chip select. The read and write are enabled by $\overline{CS}$.
21	A_0	I	Port address. A high input on A_0 selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

FUNCTIONAL DESCRIPTION

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8275 to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

A_0	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

$\overline{RD}$ (Read)

A "low" on this input informs the 8275 that the CPU is reading data or status information from the 8275.

$\overline{WR}$ (Write)

A "low" on this input informs the 8275 that the CPU is writing data or control words to the 8275.

$\overline{CS}$ (Chip Select)

A "low" on this input selects the 8275. No reading or writing will occur unless the device is selected. When $\overline{CS}$ is high, the Data Bus in the float state and $\overline{RD}$ and $\overline{WR}$ will have no effect on the chip.

DRQ (DMA Request)

A "high" on this output informs the DMA Controller that the 8275 desires a DMA transfer.

$\overline{DACK}$ (DMA Acknowledge)

A "low" on this input informs the 8275 that a DMA cycle is in progress.

IRQ (Interrupt Request)

A "high" on this output informs the CPU that the 8275 desires interrupt service.

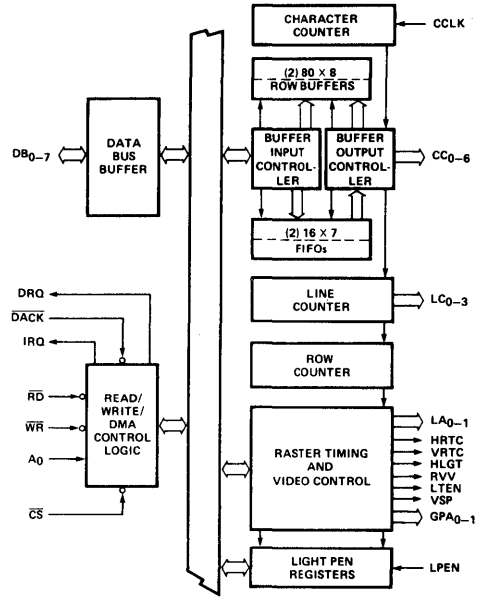


Figure 1. 8275 Block Diagram Showing Data Bus Buffer and Read/Write Functions

A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	Write 8275 Parameter
0	1	0	0	Read 8275 Parameter
1	0	1	0	Write 8275 Command
1	1	0	0	Read 8275 Status
X	1	1	0	Three-State
X	X	X	1	Three-state

Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be a derivative of the external dot clock.

Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Sweeps) per character row. Its outputs are used to address the external character generator ROM.

Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

Light Pen Registers

The Light Pen Registers are two registers that store the contents of the character counter and the row counter whenever there is a rising edge on the LPEN (Light Pen) input.

Note: Software correction is required.

Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video Control circuitry controls the generation of LA_{0-1} (Line Attribute), HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA_{0-1} (General Purpose Attribute) outputs.

Row Buffers

The Row Buffers are two 80 character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.

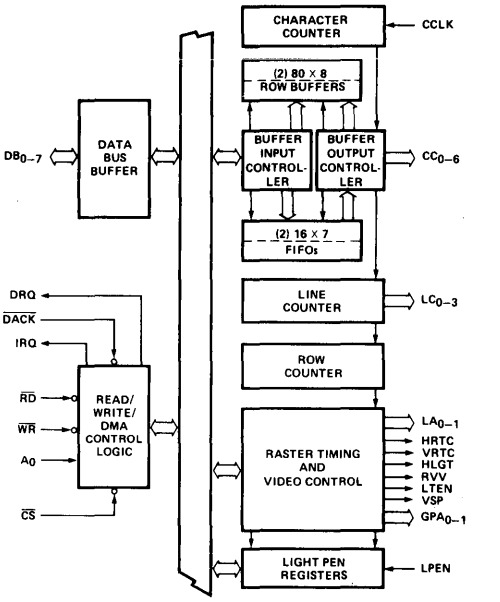


Figure 2. 8275 Block Diagram Showing Counter and Register Functions

FIFOs

There are two 16 character FIFOs in the 8275. They are used to provide extra row buffer length in the Transparent Attribute Mode (see Detailed Operation section).

Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the character is a character attribute, field attribute or special code, these controllers control the appropriate action. (Examples: An "End of Screen–Stop DMA" special code will cause the Buffer Input Controller to stop further DMA requests. A "Highlight" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

SYSTEM OPERATION

The 8275 is programmable to a large number of different display formats. It provides raster timing, display row buffering, visual attribute decoding, cursor timing, and light pen detection. It is designed to interface with the 8257 DMA Controller and standard character generator ROMs for dot matrix decoding. Dot level timing must be provided by external circuitry.

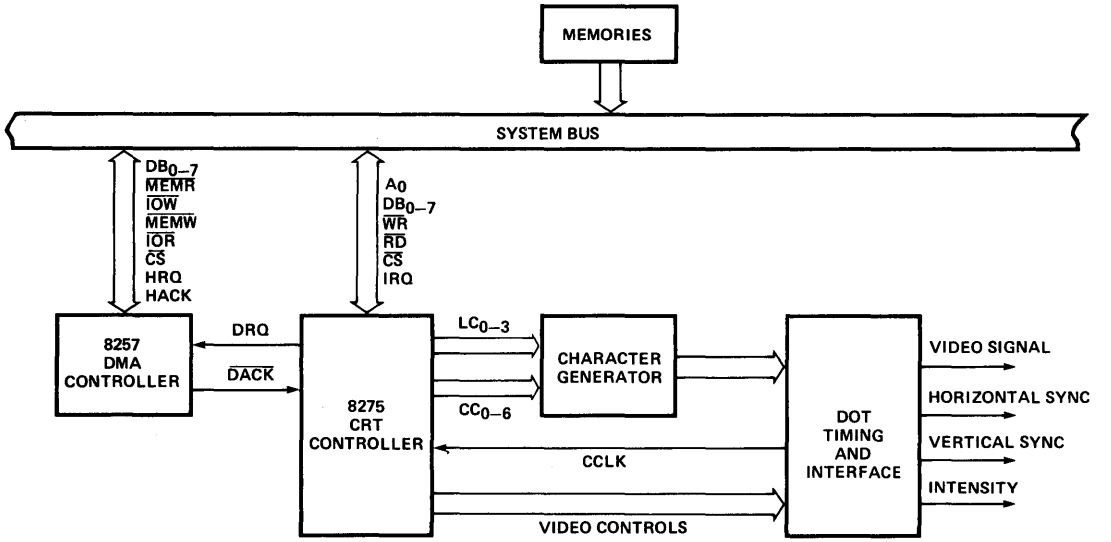


Figure 3. 8275 Systems Block Diagram Showing Systems Operation

General Systems Operational Description

The 8275 provides a "window" into the microcomputer system memory.

Display characters are retrieved from memory and displayed on a row by row basis. The 8275 has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8275 requests DMA to fill the row buffer that is not being used for display. DMA burst length and spacing is programmable. (See Programming Section.)

The 8275 displays character rows one line at a time.

The number of lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8275 provides special Control Codes which can be used to minimize DMA or software overhead. It also provides Visual Attribute Codes to cause special action or symbols on the screen without the use of the character generator (see Visual Attributes Section).

The 8275 also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is programmable.

The 8275 can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

The 8275 has a light pen input and registers. The light pen input is used to load the registers. Light pen registers can be read on command. (See Programming Section.)

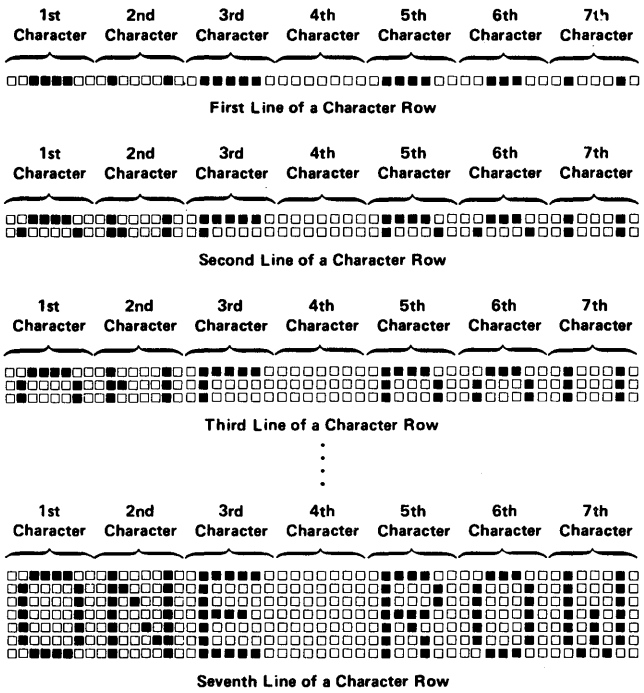


Figure 4. Display of a Character Row

Display Row Buffering

Before the start of a frame, the 8275 requests DMA and one row buffer is filled with characters.

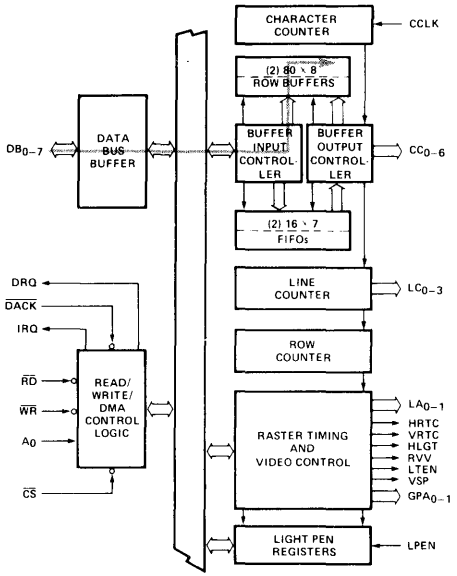


Figure 5. First Row Buffer Filled

When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, DMA begins filling the other row buffer with the next row of characters.

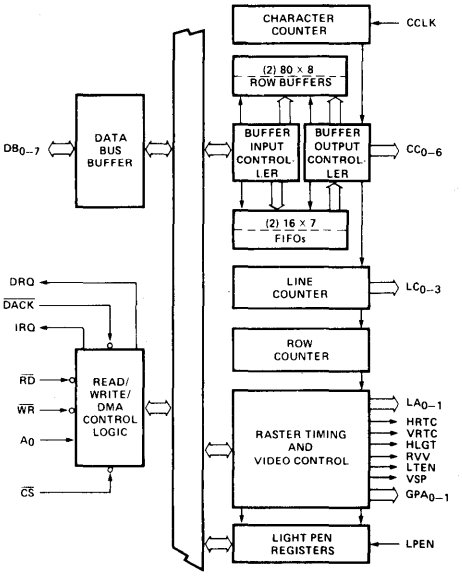


Figure 6. Second Buffer Filled, First Row Displayed

After all the lines of the character row are scanned, the roles of the two row buffers are reversed and the same procedure is followed for the next row.

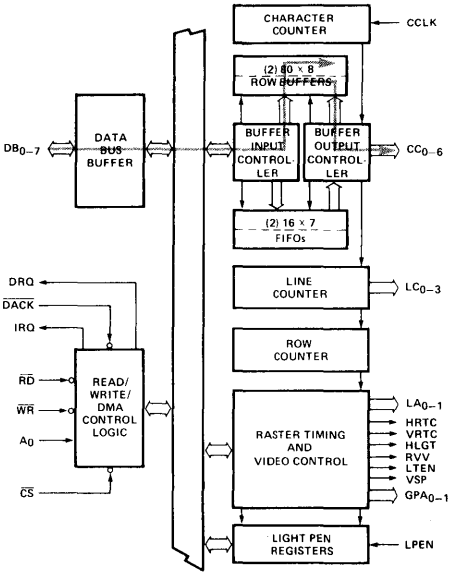


Figure 7. First Buffer Filled with Third Row, Second Row Displayed

This is repeated until all of the character rows are displayed.

Display Format

Screen Format

The 8275 can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

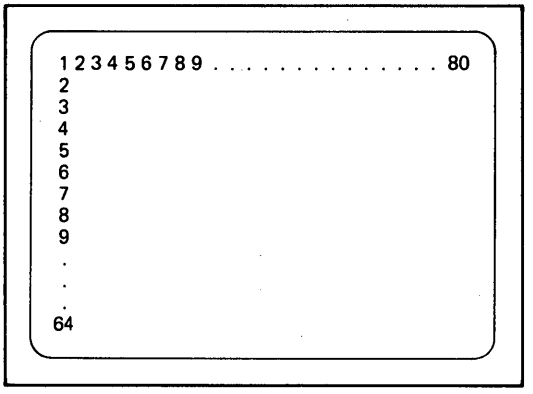


Figure 8. Screen Format

The 8275 can also be programmed to blank alternate rows. In this mode, the first row is displayed, the second blanked, the third displayed, etc. DMA is not requested for the blanked rows.

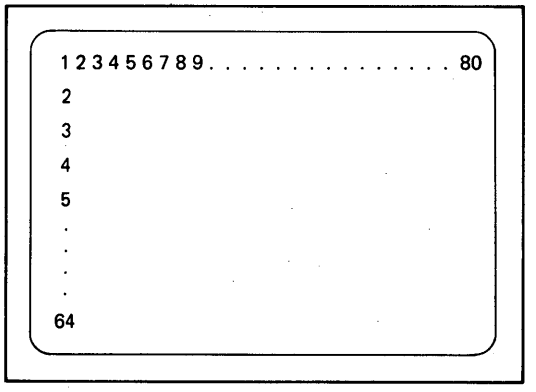


Figure 9. Blank Alternate Rows Mode

Row Format

The 8275 is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the whole character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line *counter* is the same as the line *number*.

In mode 1, the line *counter* is offset by one from the line *number*.

Note: In mode 1, while the *first* line (line number 0) is being displayed, the *last* count is output by the line counter (see examples).

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □ □	0000	1111
1	□ □ □ □ ■ □ □ □ □	0001	0000
2	□ □ □ ■ □ ■ □ □ □	0010	0001
3	□ □ ■ □ □ □ ■ □ □	0011	0010
4	□ ■ □ □ □ □ □ ■ □	0100	0011
5	□ ■ □ □ □ □ □ ■ □	0101	0100
6	□ ■ ■ ■ ■ ■ ■ ■ □	0110	0101
7	□ ■ □ □ □ □ □ ■ □	0111	0110
8	□ ■ □ □ □ □ □ ■ □	1000	0111
9	□ ■ □ □ □ □ □ ■ □	1001	1000
10	□ □ □ □ □ □ □ □ □	1010	1001
11	□ □ □ □ □ □ □ □ □	1011	1010
12	□ □ □ □ □ □ □ □ □	1100	1011
13	□ □ □ □ □ □ □ □ □	1101	1100
14	□ □ □ □ □ □ □ □ □	1110	1101
15	□ □ □ □ □ □ □ □ □	1111	1110

Figure 10. Example of a 16-Line Format

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □	0000	1001
1	□ □ □ ■ □ □ □	0001	0000
2	□ □ ■ □ ■ □ □	0010	0001
3	□ ■ □ □ □ ■ □	0011	0010
4	□ ■ □ □ □ ■ □	0100	0011
5	□ ■ ■ ■ ■ ■ □	0101	0100
6	□ ■ □ □ □ ■ □	0110	0101
7	□ ■ □ □ □ ■ □	0111	0110
8	□ □ □ □ □ □ □	1000	0111
9	□ □ □ □ □ □ □	1001	1000

Figure 11. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

Underline placement is also programmable (from line *number* 0 to 15). This is independent of the line *counter* mode.

If the line *number* of the underline is greater than 7 (line *number* MSB = 1), then the top and bottom lines will be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1011
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ □ ■ □ ■ □ □	0010	0001
3	□ □ ■ □ □ □ ■ □	0011	0010
4	□ ■ □ □ □ □ □ ■	0100	0011
5	□ ■ □ □ □ □ □ ■	0101	0100
6	□ ■ ■ ■ ■ ■ ■ ■	0110	0101
7	□ ■ □ □ □ □ □ ■	0111	0110
8	□ ■ □ □ □ □ □ ■	1000	0111
9	□ ■ □ □ □ □ □ ■	1001	1000
10	■ ■ ■ ■ ■ ■ ■ ■	1010	1001
11	□ □ □ □ □ □ □ □	1011	1010

Top and Bottom Lines are Blanked

Figure 12. Underline in Line Number 10

If the line *number* of the underline is less than or equal to 7 (line *number* MSB = 0), then the top and bottom lines will *not* be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ ■ □ □ □	0000	0111
1	□ □ ■ □ ■ □ □	0001	0000
2	□ ■ □ □ □ ■ □	0010	0001
3	□ ■ □ □ □ ■ □	0011	0010
4	□ ■ ■ ■ ■ ■ □	0100	0011
5	□ ■ □ □ □ ■ □	0101	0100
6	□ ■ □ □ □ ■ □	0110	0101
7	■ ■ ■ ■ ■ ■ ■	0111	0110

Top and Bottom Lines are not Blanked

Figure 13. Underline in Line Number 7

If the line *number* of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

Dot Format

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.

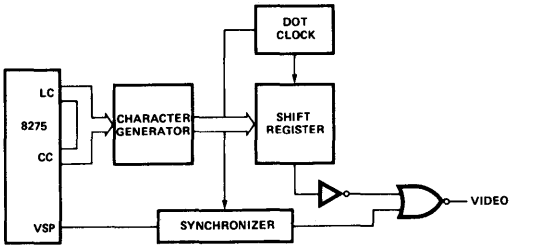


Figure 14. Typical Dot Level Block Diagram

Dot width is a function of dot clock frequency.

Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

Note: Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

Raster Timing

The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This is constantly repeated.

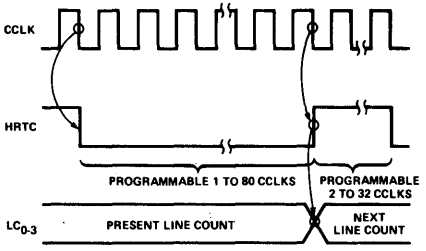


Figure 15. Line Timing

The line counter is driven by the character counter. It is used to generate the line address outputs (LC_{0-3}) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.

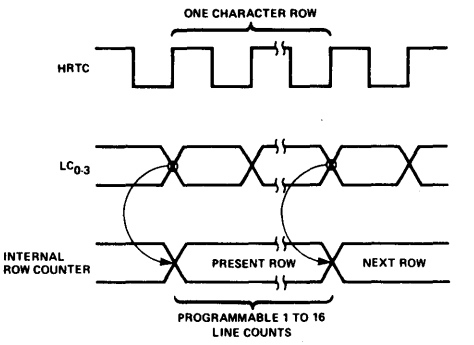


Figure 16. Row Timing

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).

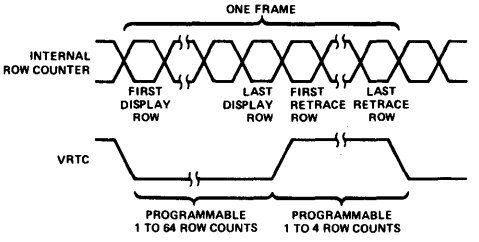


Figure 17. Frame Timing

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

DMA Timing

The 8275 can be programmed to request burst DMA transfers of 1 to 8 characters. The interval between bursts is also programmable (from 0 to 55 character clock periods ±1). This allows the user to tailor his DMA overhead to fit his system needs.

The first DMA request of the frame occurs one *row time* before the end of vertical retrace. DMA requests continue as programmed, until the row buffer is filled. If the row buffer is filled in the middle of a burst, the 8275 terminates the burst and resets the burst counter. No more DMA requests will occur until the *beginning* of the *next* row. At that time, DMA requests are activated as programmed until the other buffer is filled.

The first DMA request for a row will start at the first character clock of the preceding row. If the burst mode is used, the first DMA request may occur a number of character clocks later. This number is equal to the programmed burst space.

If, for any reason, there is a DMA underrun, a flag in the status word will be set.

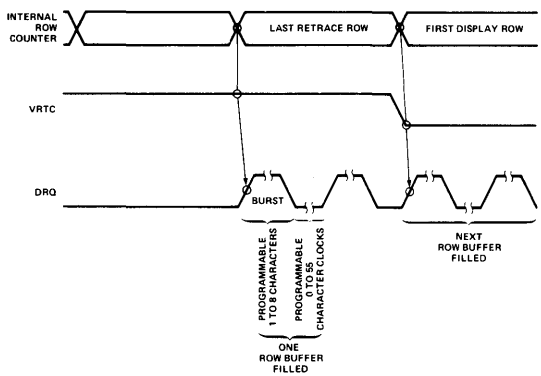


Figure 18. DMA Timing

The DMA controller is typically initialized for the next frame at the end of the current frame.

Interrupt Timing

The 8275 can be programmed to generate an interrupt request at the end of each frame. This can be used to reinitialize the DMA controller. If the 8275 interrupt enable flag is set, an interrupt request will occur at the *beginning* of the *last display row*.

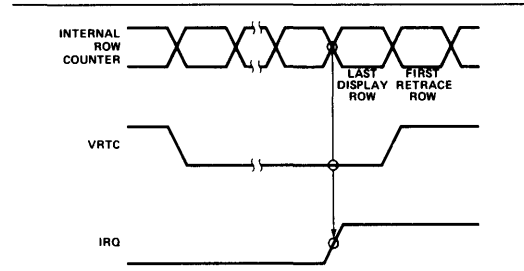


Figure 19. Beginning of Interrupt Request

IRQ will go inactive after the status register is read.

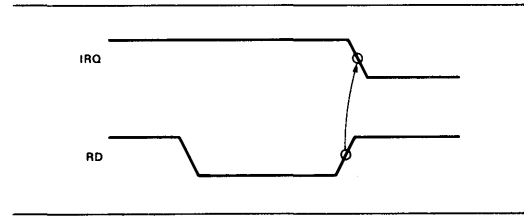


Figure 20. End of Interrupt Request

A reset command will also cause IRQ to go inactive, but this is not recommended during normal service.

Another method of reinitializing the DMA controller is to have the DMA controller itself interrupt on terminal count. With this method, the 8275 interrupt enable flag should not be set.

Note: Upon power-up, the 8275 Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8275 before system interrupts are enabled.

VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8275 are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Visual Attribute or Special Code (MSB = 1).

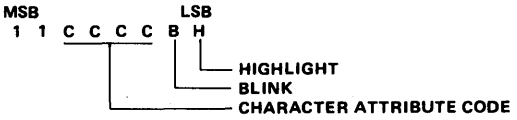
There are two types of Visual Attribute Codes. They are Character Attributes and Field Attributes.

Character Attribute Codes

Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA_{0-1}), the Video Suppression output (VSP), and the Light Enable output. The dot level timing circuitry can use these signals to generate the proper symbols.

Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT).

Character Attributes



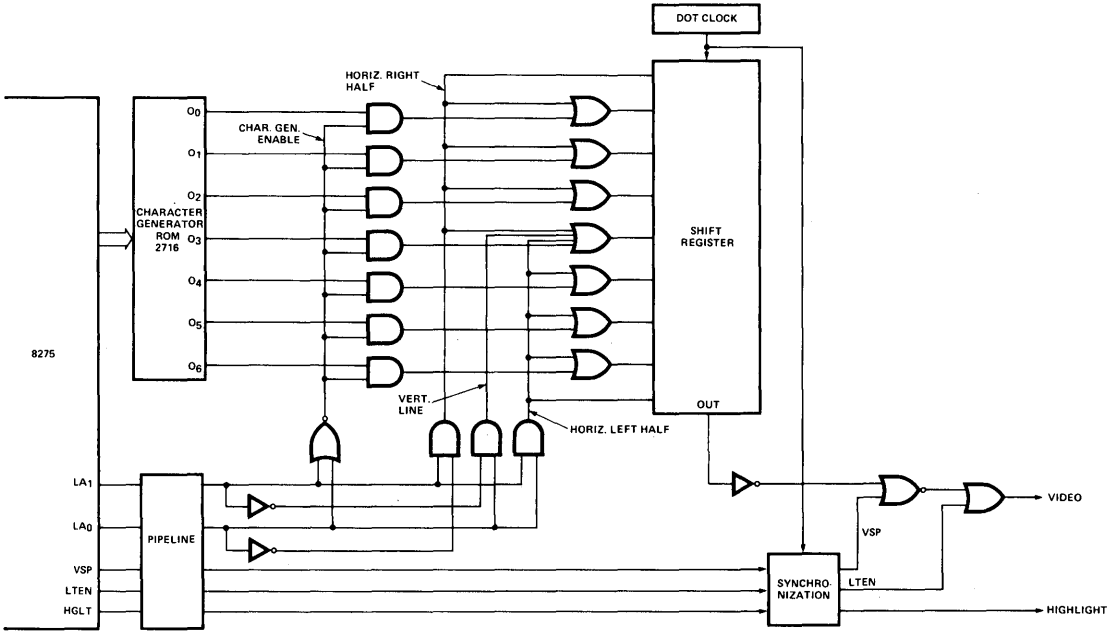


Figure 21. Typical Character Attribute Logic

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"		OUTPUTS LA_1	LA_0	VSP	LTEN	SYMBOL	DESCRIPTION
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

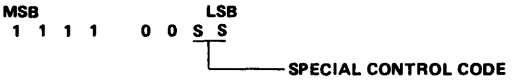
Blinking is active when B = 1.

Highlight is active when H = 1.

Special Codes

Four special codes are available to help reduce memory, software, or DMA overhead.

Special Control Character



S S	FUNCTION
0 0	End of Row
0 1	End of Row-Stop DMA
1 0	End of Screen
1 1	End of Screen-Stop DMA

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop DMA Code (01) causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop DMA Code (11) causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the Row Buffer. It affects the display in the same way as the End of Screen Code (10).

If the Stop DMA feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character are ignored.

Note: If a Stop DMA character is not the last character in a burst or row, DMA is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop DMA character.

Field Attributes

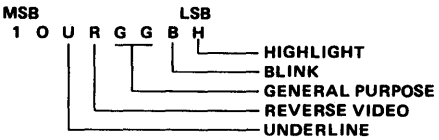
The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

There are six field attributes:

1. *Blink* – Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight* – Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video* – Characters following the code are caused to appear with reverse video by activating the Reverse Video output (RVV).
4. *Underline* – Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).

5,6. *General Purpose* – There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. GPA_{0-1} are active high outputs.

Field Attribute Code



H = 1 FOR HIGHLIGHTING
B = 1 FOR BLINKING
R = 1 FOR REVERSE VIDEO
U = 1 FOR UNDERLINE
GG = GPA_1, GPA_0

*More than one attribute can be enabled at the same time. If the blinking and reverse video attributes are enabled simultaneously, only the reversed characters will blink.

The 8275 can be programmed to provide visible or invisible field attribute characters.

If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.

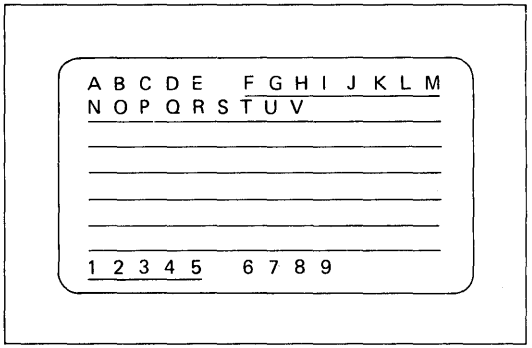


Figure 22. Example of the Visible Field Attribute Mode (Underline Attribute)

If the 8275 is programmed in the invisible field attribute mode, the 8275 FIFO is activated.

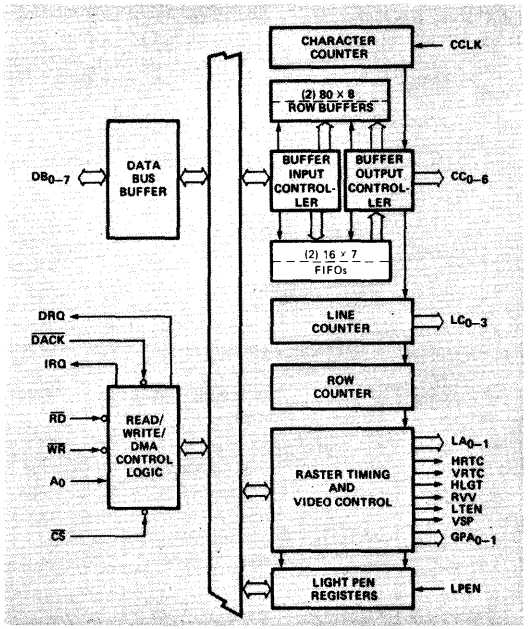


Figure 23. Block Diagram Showing FIFO Activation

Each row buffer has a corresponding FIFO. These FIFOs are 16 characters by 7 bits in size.

When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the *next* character in the proper FIFO.

When a field attribute is placed in the Buffer Output Controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC_{0-6}). The chosen Visual Attributes are also activated.

Since the FIFO is 16 characters long, no more than 16 field attribute characters may be used per line in this mode. If more are used, a bit in the status word is set and the first characters in the FIFO are written over and lost.

Note: Since the FIFO is 7 bits wide, the MSB of any characters put in it are stripped off. Therefore, a Visual Attribute or Special Code must *not* immediately follow a field attribute code. If this situation does occur, the Visual Attribute or Special Code will be treated as a normal display character.

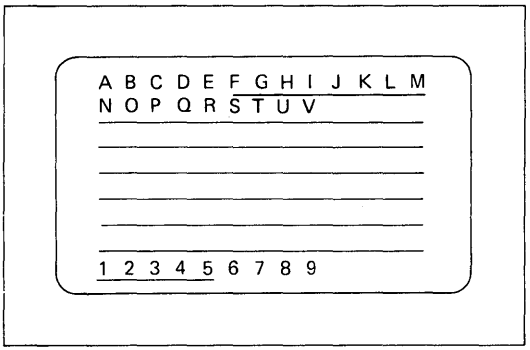


Figure 24. Example of the Invisible Field Attribute Mode (Underline Attribute)

Field and Character Attribute Interaction

Character Attribute Symbols are affected by the Reverse Video (RVV) and General Purpose (GPA_{0-1}) field attributes. They are not affected by Underline, Blink or Highlight field attributes; however, these characteristics can be programmed *individually* for Character Attribute Symbols.

Cursor Timing

The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

1. a blinking underline
2. a blinking reverse video block
3. a non-blinking underline
4. a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.

Light Pen Detection

A light pen consists of a micro switch and a tiny light sensor. When the light pen is pressed against the CRT screen, the micro switch enables the light sensor. When the raster sweep reaches the light sensor, it triggers the light pen output.

If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in a pair of registers. These registers can be read on command. A bit in the status word is set, indicating that the light pen signal was detected. The LPEN input must be a 0 to 1 transition for proper operation.

Note: Due to internal and external delays, the character position coordinate will be off by at least three character positions. This has to be corrected in software.

Device Programming

The 8275 has two programming registers, the Command Register (CREG) and the Parameter Register (PREG). It also has a Status Register (SREG). The Command Register can only be written into and the Status Registers can only be read from. They are addressed as follows:

A_0	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

The 8275 expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

Instruction Set

The 8275 instruction set consists of 8 commands.

COMMAND	NO. OF PARAMETER BYTES
Reset	4
Start Display	0
Stop Display	0
Read Light Pen	2
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8275 (SREG) can be read by the CPU at any time.

1. Reset Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Reset Command	0	0	0	0	0	0	0	0
Parameters	Write	0	Screen Comp Byte 1	S	H	H	H	H	H	H	H
	Write	0	Screen Comp Byte 2	V	V	R	R	R	R	R	R
	Write	0	Screen Comp Byte 3	U	U	U	U	L	L	L	L
	Write	0	Screen Comp Byte 4	M	F	C	C	Z	Z	Z	Z

Action – After the reset command is written, DMA requests stop, 8275 interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

Parameter – S Spaced Rows

S	FUNCTIONS
0	Normal Rows
1	Spaced Rows

Parameter – HHHHHHH Horizontal Characters/Row

H	H	H	H	H	H	H	NO. OF CHARACTERS PER ROW
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	2
0	0	0	0	0	1	0	3
			.				.
			.				.
			.				.
1	0	0	1	1	1	1	80
1	0	1	0	0	0	0	Undefined
			.				.
			.				.
			.				.
1	1	1	1	1	1	1	Undefined

Parameter – VV Vertical Retrace Row Count

V	V	NO. OF ROW COUNTS PER VRTC
0	0	1
0	1	2
1	0	3
1	1	4

Parameter – RRRRRR Vertical Rows/Frame

R	R	R	R	R	R	NO. OF ROWS/FRAME
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
			.			.
			.			.
			.			.
1	1	1	1	1	1	64

Parameter – UUUU Underline Placement

U	U	U	U	LINE NUMBER OF UNDERLINE
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		.		.
		.		.
		.		.
1	1	1	1	16

Parameter – LLLL Number of Lines per Character Row

L	L	L	L	NO. OF LINES/ROW
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		.		.
		.		.
		.		.
1	1	1	1	16

Parameter – M Line Counter Mode

M	LINE COUNTER MODE
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

Parameter – F Field Attribute Mode

F	FIELD ATTRIBUTE MODE
0	Transparent
1	Non-Transparent

Parameter – CC Cursor Format

C	C	CURSOR FORMAT
0	0	Blinking reverse video block
0	1	Blinking underline
1	0	Nonblinking reverse video block
1	1	Nonblinking underling

Parameter – ZZZZ Horizontal Retrace Count

Z	Z	Z	Z	NO. OF CHARACTER COUNTS PER HRTC
0	0	0	0	2
0	0	0	1	4
0	0	1	0	6
		.		.
		.		.
		.		.
1	1	1	1	32

Note: uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

2. Start Display Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Start Display	0	0	1	S	S	S	B	B
No parameters											

S S S BURST SPACE CODE

S S S	NO. OF CHARACTER CLOCKS BETWEEN DMA REQUESTS
0 0 0	0
0 0 1	7
0 1 0	15
0 1 1	23
1 0 0	31
1 0 1	39
1 1 0	47
1 1 1	55

B B BURST COUNT CODE

B B	NO. OF DMA CYCLES PER BURST
0 0	1
0 1	2
1 0	4
1 1	8

Action – 8275 interrupts are enabled, DMA requests begin, video is enabled, Interrupt Enable and Video Enable status flags are set.

3. Stop Display Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Stop Display	0	1	0	0	0	0	0	0
No parameters											

Action – Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to re-enable the display.

4. Read Light Pen Command

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Read Light Pen	0	1	1	0	0	0	0	0
Parameters	Read	0	Char. Number	(Char. Position in Row)							
	Read	0	Row Number	(Row Number)							

Action – The 8275 is conditioned to supply the contents of the light pen position registers in the next two read cycles of the parameter register. Status flags are not affected.

Note: Software correction of light pen position is required.

5. Load Cursor Position:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Load Cursor	1	0	0	0	0	0	0	0
Parameters	Write	0	Char. Number	(Char. Position in Row)							
	Write	0	Row Number	(Row Number)							

Action – The 8275 is conditioned to place the next two parameter bytes into the cursor position registers. Status flags not affected.

6. Enable Interrupt Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Enable Interrupt	1	0	1	0	0	0	0	0
No parameters											

Action – The interrupt enable status flag is set and interrupts are enabled.

7. Disable Interrupt Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Disable Interrupt	1	1	0	0	0	0	0	0
No parameters											

Action – Interrupts are disabled and the interrupt enable status flag is reset.

8. Preset Counters Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Preset Counters	1	1	1	0	0	0	0	0
No parameters											

Action – The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU.

Status Flags

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB	LSB
Command	Read	1	Status Word	0 IE IR LP IC VE OU F0	

IE – (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a "Start Display" command and reset with the "Reset" command.

IR – (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.

LP – This flag is set when the light pen input (LPEN) is activated and the light pen registers have been loaded. This flag is automatically reset after a status read.

IC – (Improper Command) This flag is set when a command parameter string is too long or too short. The flag is automatically reset after a status read.

VE – (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a "Start Display" command, and reset on a "Stop Display" or "Reset" command.

DU – (DMA Underrun) This flag is set whenever a data underrun occurs during DMA transfers. Upon detection of DU, the DMA operation is stopped and the screen is blanked until after the vertical retrace interval. This flag is reset after a status read.

FO – (FIFO Overrun) This flag is set whenever the FIFO is overrun. It is reset on a status read.

ABSOLUTE MAXIMUM RATINGS*

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 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5V ±5%

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5V	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 μA
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		160	mA	

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

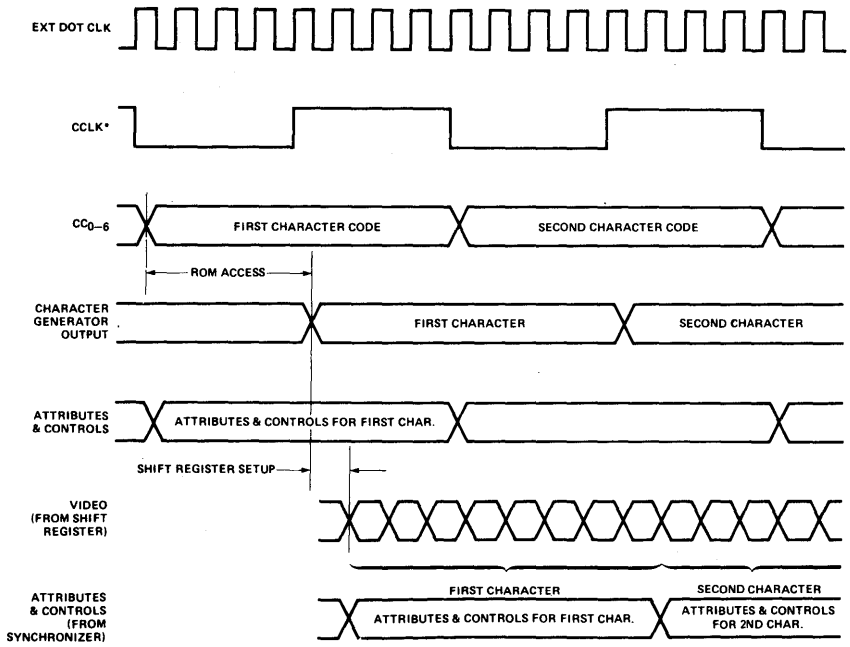
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to V_{SS}.

Other Timing:

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{CC}	Character Code Output Delay		150	ns	C_L = 50 pF
t_{HR}	Horizontal Retrace Output Delay		200	ns	C_L = 50 pF
t_{LC}	Line Count Output Delay		400	ns	C_L = 50 pF
t_{AT}	Control/Attribute Output Delay		275	ns	C_L = 50 pF
t_{VR}	Vertical Retrace Output Delay		275	ns	C_L = 50 pF
t_{RI}	IRQ↓ from RD↑		250	ns	C_L = 50 pF
t_{WQ}	DRQ↑ from WR↑		250	ns	C_L = 50 pF
t_{RQ}	DRQ↓ from WR↓		200	ns	C_L = 50 pF
t_{LR}	DACK↓ to WR↓	0		ns	
t_{RL}	WR↑ to DACK↑	0		ns	
t_{PR}	LPEN Rise		50	ns	
t_{PH}	LPEN Hold	100		ns	

Note: Timing measurements are made at the following reference voltages: Output "1" = 2.0V, "0" = 0.8V.
Input "1"=2.4V, "0"=0.45V

WAVEFORMS



*CCLK IS A MULTIPLE OF THE DOT CLOCK AND AN INPUT TO THE 8275.

Figure 25. Typical Dot Level Timing

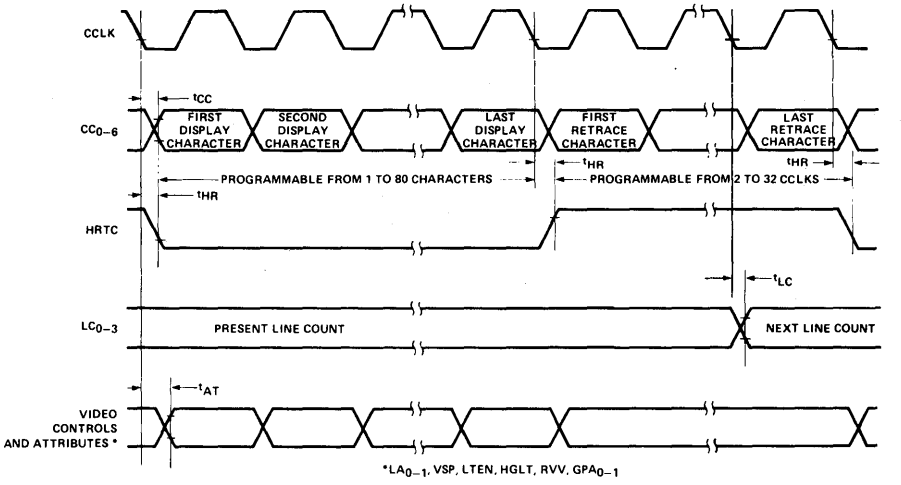


Figure 26. Line Timing

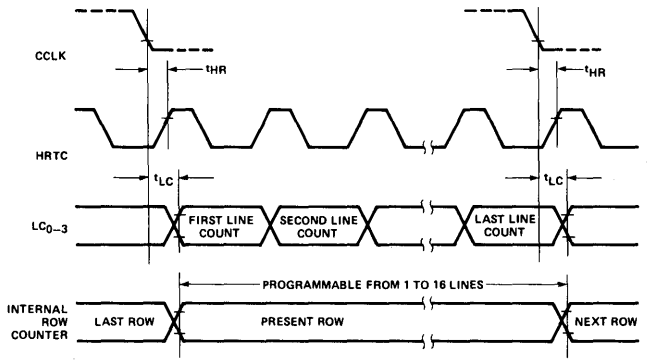


Figure 27. Row Timing

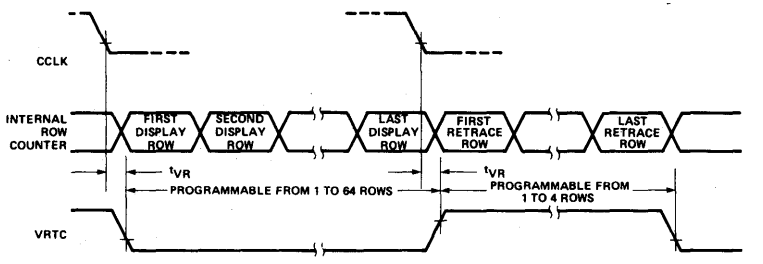


Figure 28. Frame Timing

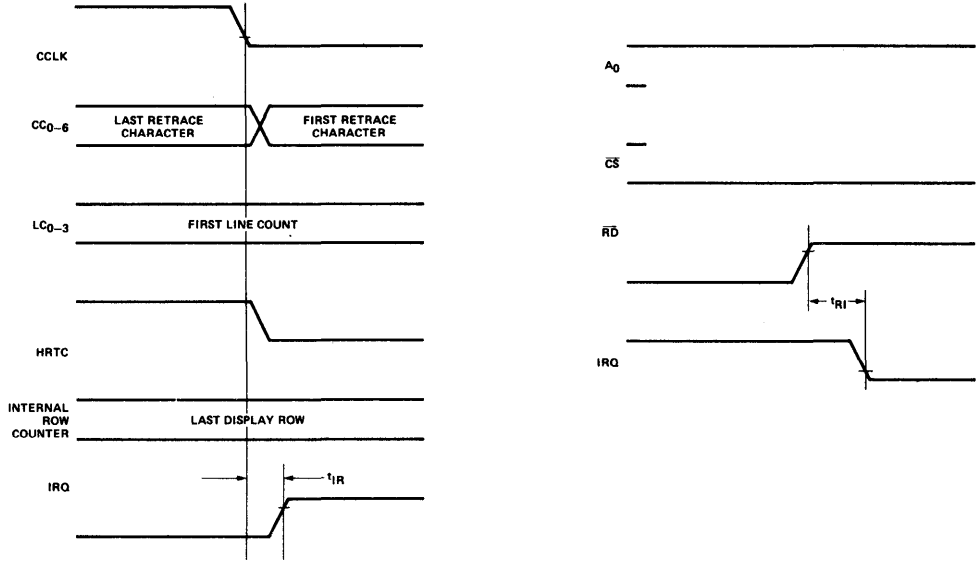


Figure 29. Interrupt Timing

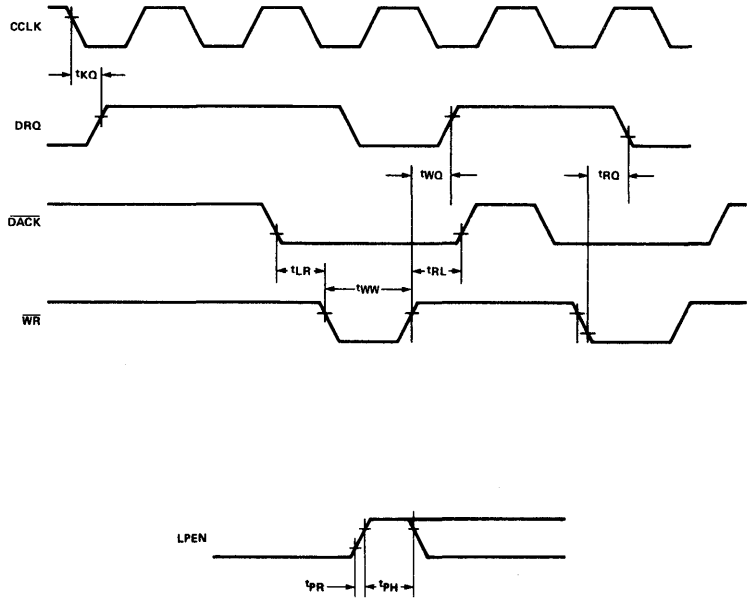


Figure 30. DMA Timing

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{AR}	Address Stable Before READ	0		ns	
t_{RA}	Address Hold Time for READ	0		ns	
t_{RR}	READ Pulse Width	250		ns	
t_{RD}	Data Delay from READ		200	ns	C_L = 150 pF
t_{DF}	READ to Data Floating	20	100	ns	

Write Cycle:

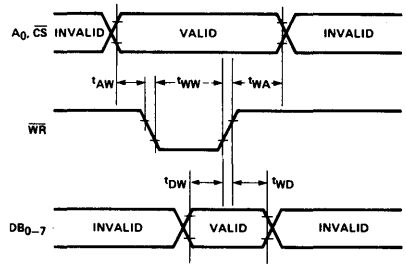
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{AW}	Address Stable Before WRITE	0		ns	
t_{WA}	Address Hold Time for WRITE	0		ns	
t_{WW}	WRITE Pulse Width	250		ns	
t_{DW}	Data Setup Time for WRITE	150		ns	
t_{WD}	Data Hold Time for WRITE	0		ns	

Clock Timing:

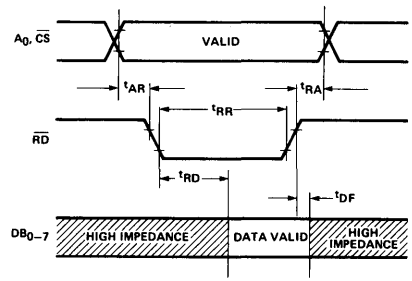
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{CLK}	Clock Period	480		ns	
t_{KH}	Clock High	240		ns	
t_{KL}	Clock Low	160		ns	
t_{KR}	Clock Rise	5	30	ns	
t_{KF}	Clock Fall	5	30	ns	

Note 1: AC timings measured at V_{OH} = 2.0, V_{OL} = 0.8 V_{IH}=2.4, V_{IL}=0.45

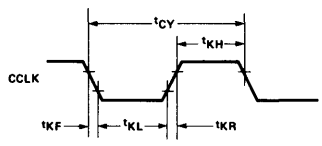
Write Timing



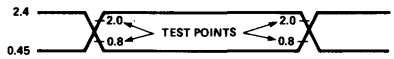
Read Timing



Clock Timing



Input Waveforms (For A.C. Tests)



00224A

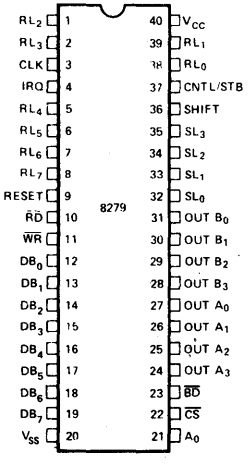
8279/8279-5
PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- MCS-85™ Compatible 8279-5
- Simultaneous Keyboard Display Operations
- Scanned Keyboard Mode
- Scanned Sensor Mode
- Strobed Input Entry Mode
- 8-Character Keyboard FIFO
- 2-Key Lockout or N-Key Rollover with Contact Debounce
- Dual 8- or 16-Numerical Display
- Single 16-Character Display
- Right or Left Entry 16-Byte Display RAM
- Mode Programmable from CPU
- Programmable Scan Timing
- Interrupt Output on Key Entry

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

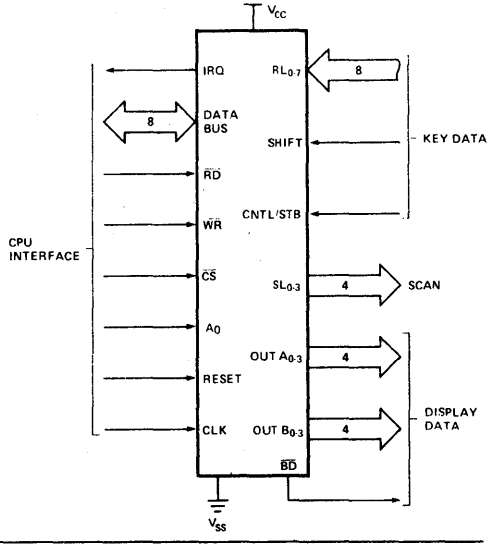
PIN CONFIGURATION



PIN NAMES

DB_{0-7}	I/O	DATA BUS (BI DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
$\overline{CS}$	I	CHIP SELECT
$\overline{RD}$	I	READ INPUT
$\overline{WR}$	I	WRITE INPUT
A_0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL_{0-3}	O	SCAN LINES
RL_{0-7}	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
$OUT\ A_{0-3}$	O	DISPLAY (A) OUTPUTS
$OUT\ B_{0-3}$	O	DISPLAY (B) OUTPUTS
$\overline{BD}$	O	BLANK DISPLAY OUTPUT

LOGIC SYMBOL



AFN-00742A-01

HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

No. Of Pins	Designation	Function
8	DB_0-DB_7	Bi-directional data bus. All data and commands between the CPU and the 8279 are transmitted on these lines.
1	CLK	Clock from system used to generate internal timing.
1	RESET	A high signal on this pin resets the 8279. After being reset the 8279 is placed in the following mode: 1) 16 8-bit character display —left entry. 2) Encoded scan keyboard—2 key lockout. Along with this the program clock prescaler is set to 31.
1	$\overline{CS}$	Chip Select. A low on this pin enables the interface functions to receive or transmit.
1	A_0	Buffer Address. A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
2	$\overline{RD}$, $\overline{WR}$	Input/Output read and write. These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
1	IRQ	Interrupt Request. In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
2	V_{SS}, V_{CC}	Ground and power supply pins.
4	SL_0-SL_3	Scan Lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
8	RL_0-RL_7	Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
1	SHIFT	The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
1	CNTL/STB	For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
4 4	OUT A_0-OUT A_3 OUT B_0-OUT B_3	These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL_0-SL_3) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8 bit port.
1	$\overline{BD}$	Blank Display. This output is used to blank the display during digit switching or by a display blanking command.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by $\overline{CS}$. The character of the information, given or desired by the CPU, is identified by A_0. A logic one means the information is a command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS}$ = 1), the devices are in a high impedance state. The drivers input during $\overline{WR} \bullet \overline{CS}$ and output during $\overline{RD} \bullet \overline{CS}$.

Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with A_0 = 1 and then sending a $\overline{WR}$. The command is latched on the rising edge of $\overline{WR}$.

FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

Input Modes

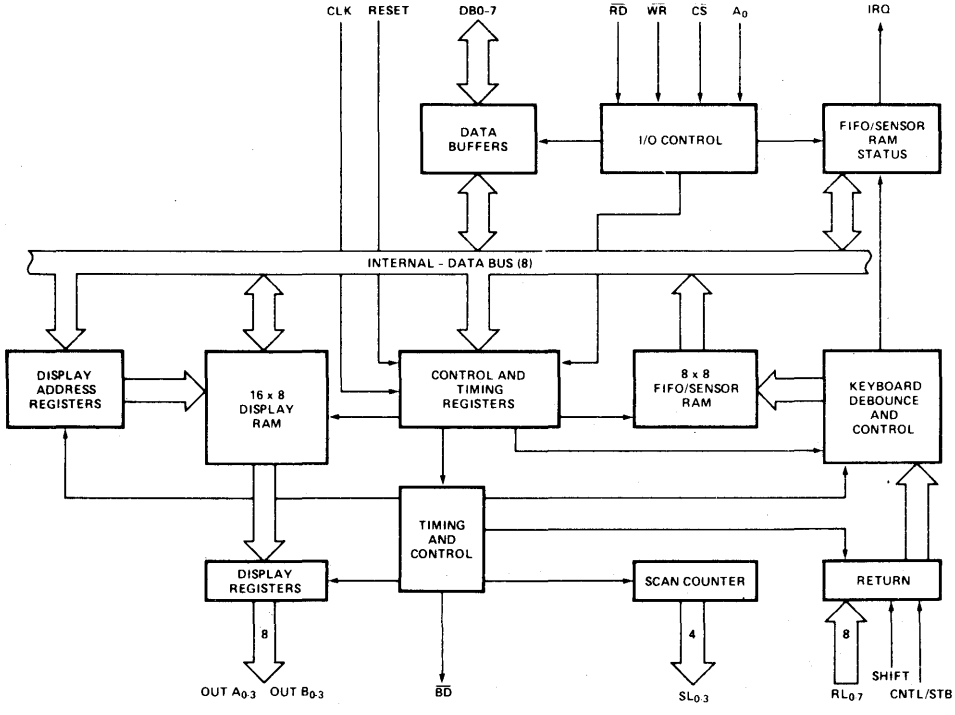
- Scanned Keyboard — with encoded (8 x 8 key keyboard) or decoded (4 x 8 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.
- Scanned Sensor Matrix — with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input -- Data on return lines during control line strobe is transferred to FIFO.

Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit ($B_0 = D_0$, $A_3 = D_7$).
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Clock Prescaler
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.



The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a ÷ N prescaler that can be programmed to yield an internal frequency of 100 kHz which gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note than when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

SOFTWARE OPERATION

8279 commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with $\overline{CS}$ low and A_0 high and are loaded to the 8279 on the rising edge of $\overline{WR}$.

Keyboard/Display Mode Set

	MSB							LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

0	0	8 8-bit character display — Left entry
0	1	16 8-bit character display — Left entry*
1	0	8 8-bit character display — Right entry
1	1	16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

0	0	0	Encoded Scan Keyboard — 2 Key Lockout*
0	0	1	Decoded Scan Keyboard — 2-Key Lockout
0	1	0	Encoded Scan Keyboard — N-Key Rollover
0	1	1	Decoded Scan Keyboard — N-Key Rollover
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input, Encoded Display Scan
1	1	1	Strobed Input, Decoded Display Scan

Program Clock

Code:	0	0	1	P	P	P	P	P

All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits PPPPP determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, PPPPP should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

Read FIFO/Sensor RAM

Code:	0	1	0	AI	X	A	A	A	X = Don't Care

The CPU sets up the 8279 for a read of the FIFO/Sensor RAM by first writing this command. In the Scan Key-

*Default after reset.

board Mode, the Auto-Increment flag (AI) and the RAM address bits (AAA) are irrelevant. The 8279 will automatically drive the data bus for each subsequent read ($A_0 = 0$) in the same sequence in which the data first entered the FIFO. All subsequent reads will be from the FIFO until another command is issued.

In the Sensor Matrix Mode, the RAM address bits AAA select one of the 8 rows of the Sensor RAM. If the AI flag is set (AI = 1), each successive read will be from the subsequent row of the sensor RAM.

Read Display RAM

Code:

0	1	1	AI	A	A	A	A

The CPU sets up the 8279 for a read of the Display RAM by first writing this command. The address bits AAAA select one of the 16 rows of the Display RAM. If the AI flag is set (AI = 1), this row address will be incremented after each following read *or write* to the Display RAM. Since the same counter is used for both reading and writing, this command sets the next read *or write* address and the sense of the Auto-Increment mode for both operations.

Write Display RAM

Code:

1	0	0	AI	A	A	A	A

The CPU sets up the 8279 for a write to the Display RAM by first writing this command. After writing the command with $A_0 = 1$, all subsequent writes with $A_0 = 0$ will be to the Display RAM. The addressing and Auto-Increment functions are identical to those for the Read Display RAM. However, this command does not affect the source of subsequent Data Reads; the CPU will read from whichever RAM (Display or FIFO/Sensor) which was last specified. If, indeed, the Display RAM was last specified, the Write Display RAM will, nevertheless, change the next Read location.

Display Write Inhibit/Blanking

The IW Bits can be used to mask nibble A and nibble B in applications requiring separate 4-bit display ports. By setting the IW flag (IW = 1) for one of the ports, the port becomes marked so that entries to the Display RAM from the CPU do not affect that port. Thus, if each nibble is input to a BCD decoder, the CPU may write a digit to the Display RAM without affecting the other digit being displayed. It is important to note that bit B_0 corresponds to bit D_0 on the CPU bus, and that bit A_3 corresponds to bit D_7.

If the user wishes to blank the display, the BL flags are available for each nibble. The last Clear command issued determines the code to be used as a "blank." This code defaults to all zeros after a reset. Note that both BL flags must be set to blank a display formatted with a single 8-bit port.

Clear

The C_D bits are available in this command to clear all rows of the Display RAM to a selectable blanking code as follows:

C_D C_D	
0 X	All Zeros (X = Don't Care)
1 0	AB = Hex 20 (0010 0000)
1 1	All Ones

Enable clear display when = 1 (or by C_A = 1)

During the time the Display RAM is being cleared (~160 μs), it may not be written to. The most significant bit of the FIFO status word is set during this time. When the Display RAM becomes available again, it automatically resets.

If the C_F bit is asserted ($C_F = 1$), the FIFO status is cleared and the interrupt output line is reset. Also, the Sensor RAM pointer is set to row 0.

C_A, the Clear All bit, has the combined effect of C_D and C_F; it uses the C_D clearing code on the Display RAM and also clears FIFO status. Furthermore, it resynchronizes the internal timing chain.

End Interrupt/Error Mode Set

Code:

1	1	1	E	X	X	X	X

X = Don't care.

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode — if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A_0 is high and $\overline{CS}$ and $\overline{RD}$ are low. See Interface Considerations for more detail on status word.

Data Read

Data is read when A_0, $\overline{CS}$ and $\overline{RD}$ are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of $\overline{RD}$ will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

Data Write

Data that is written with A_0, $\overline{CS}$ and $\overline{WR}$ low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of $\overline{WR}$ occurs if AI set by the latest display command.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature . . . . . 0°C to 70°C
Storage Temperature . . . . . -65°C to 125°C
Voltage on any Pin with
Respect to Ground . . . . . -0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{SS} = 0V, Note 1

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL1}	Input Low Voltage for Shift Control and Return Lines	-0.5	1.4	V	
V_{IL2}	Input Low Voltage for All Others	-0.5	0.8	V	
V_{IH1}	Input High Voltage for Shift, Control and Return Lines	2.2		V	
V_{IH2}	Input High Voltage for All Others	2.0		V	
V_{OL}	Output Low Voltage		0.45	V	Note 2
V_{OH}	Output High Voltage on Interrupt Line	3.5		V	Note 3
I_{IL1}	Input Current on Shift, Control and Return Lines		+10 -100	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0V$
I_{IL2}	Input Leakage Current on All Others		±10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		±10	μA	$V_{OUT} = V_{CC}$ to 0V
I_{CC}	Power Supply Current		120	mA	

Notes:
1. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%.
2. 8279, I_{OL} = 1.6mA; 8279-5, I_{OL} = 2.2mA.
3. 8279, I_{OH} = -100μA; 8279-5, I_{OH} = -400μA.

CAPACITANCE

SYMBOL	TEST	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{in}	Input Capacitance	5	10	pF	$V_{in}=V_{CC}$
C_{out}	Output Capacitance	10	20	pF	$V_{out}=V_{CC}$

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, (Note 1)

BUS PARAMETERS

READ CYCLE:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AR}	Address Stable Before $\overline{READ}$	50		0		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		0		ns
t_{RR}	$\overline{READ}$ Pulse Width	420		250		ns
t_{RD}[2]	Data Delay from $\overline{READ}$		300		150	ns
t_{AD}[2]	Address to Data Valid		450		250	ns
t_{DF}	$\overline{READ}$ to Data Floating	10	100	10	100	ns
t_{RCY}	Read Cycle Time	1		1		μs

WRITE CYCLE:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AW}	Address Stable Before $\overline{WRITE}$	50		0		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	20		0		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		250		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		150		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		0		ns

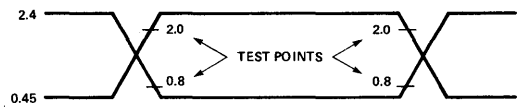
Notes:
1. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%.
2. 8279, C_L = 100pF; 8279-5, C_L = 150pF.

OTHER TIMINGS:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
$t_{\phi W}$	Clock Pulse Width	230		120		nsec
t_{CY}	Clock Period	500		320		nsec

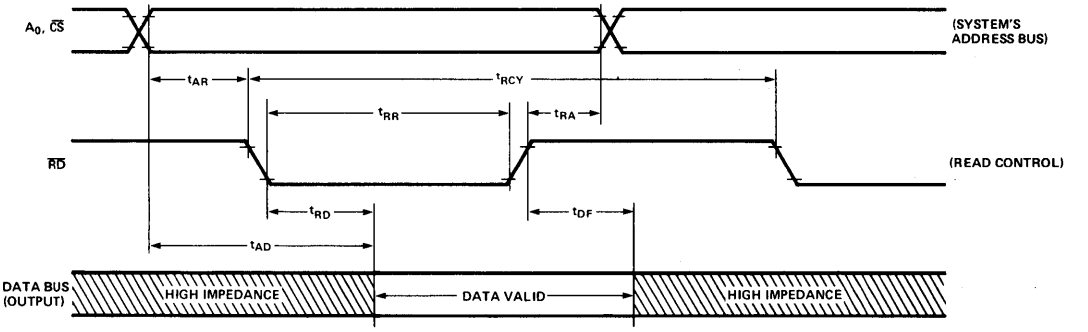
Keyboard Scan Time:	5.1 msec	Digit-on Time:	480 μsec
Keyboard Debounce Time:	10.3 msec	Blanking Time:	160 μsec
Key Scan Time:	80 μsec	Internal Clock Cycle:	10 μsec
Display Scan Time:	10.3 msec		

INPUT WAVEFORMS FOR A.C. TESTS:

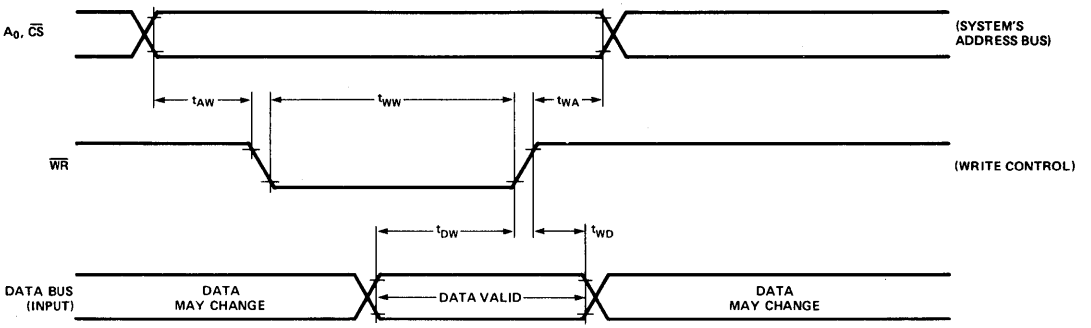


WAVEFORMS

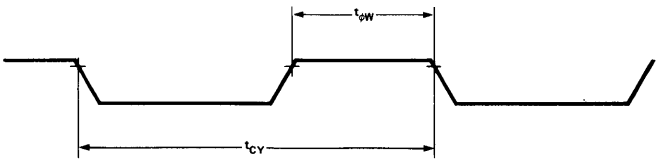
1. Read Operation



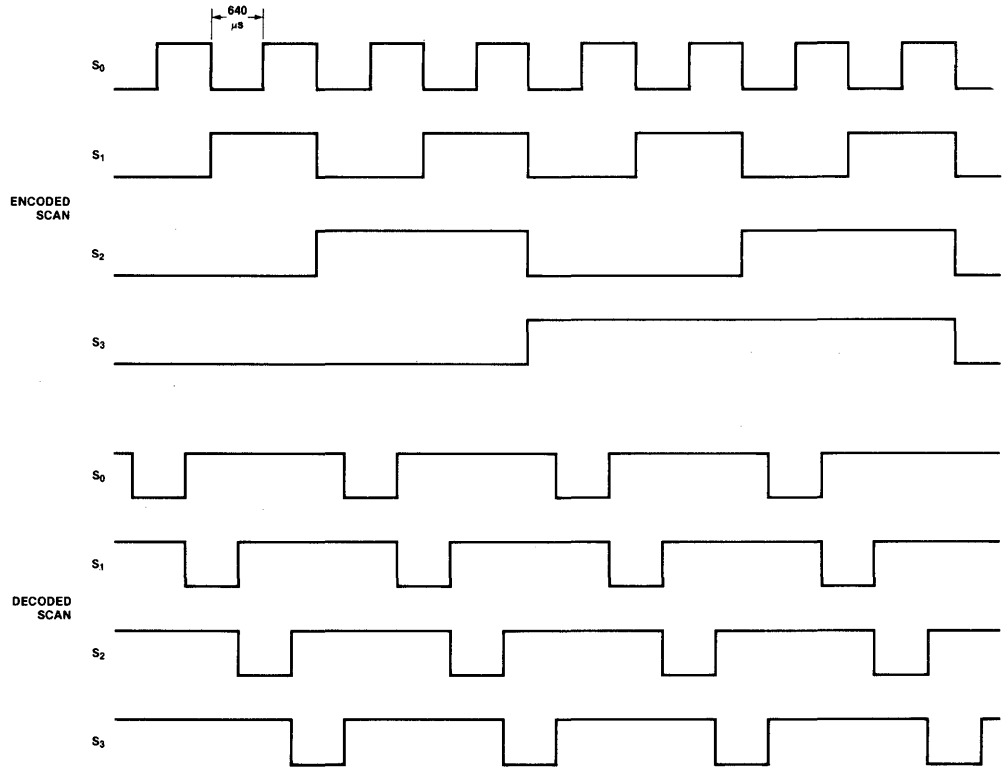
2. Write Operation



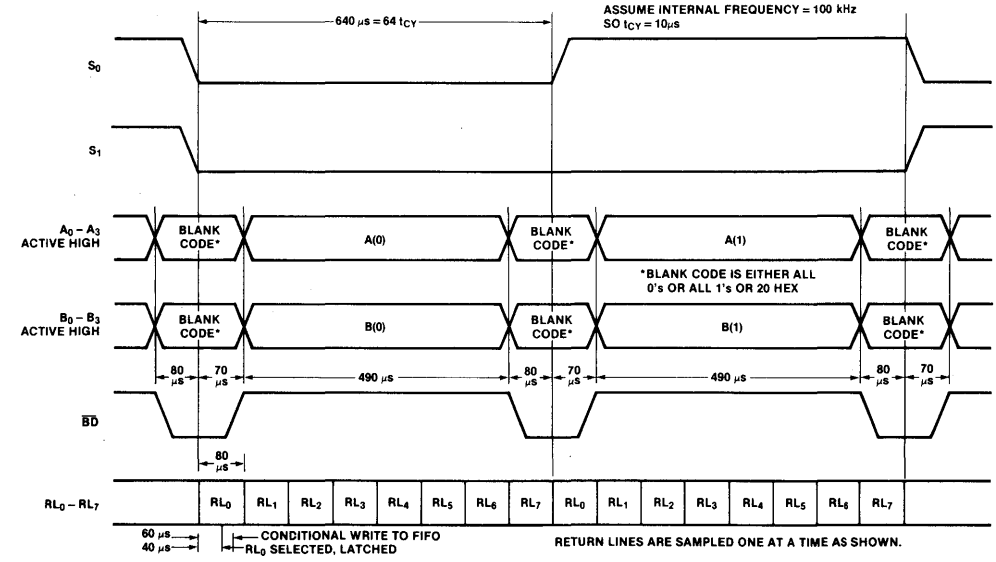
3. Clock Input



SCAN WAVEFORMS



DISPLAY WAVEFORMS



NOTE: SHOWN IS ENCODED SCAN LEFT ENTRY
S_2-S_3 ARE NOT SHOWN BUT THEY ARE SIMPLY S_1 DIVIDED BY 2 AND 4

8282/8283
OCTAL LATCH

- **Address Latch for iAPX 86,88, MCS-80™, MCS-85™, MCS-48™ Families**
- **High Output Drive Capability for Driving System Data Bus**
- **Fully Parallel 8-Bit Data Register and Buffer**
- **Transparent during Active Strobe**
- **3-State Outputs**
- **20-Pin Package with 0.3" Center**
- **No Output Low Noise when Entering or Leaving High Impedance State**

The 8282 and 8283 are 8-bit bipolar latches with 3-state output buffers. They can be used to implement latches, buffers, or multiplexers. The 8283 inverts the input data at its outputs while the 8282 does not. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with these devices.

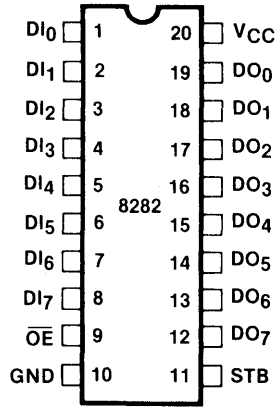


Figure 1. 8282 Pin Configuration

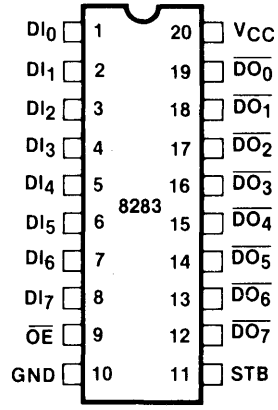


Figure 2. 8283 Pin Configuration

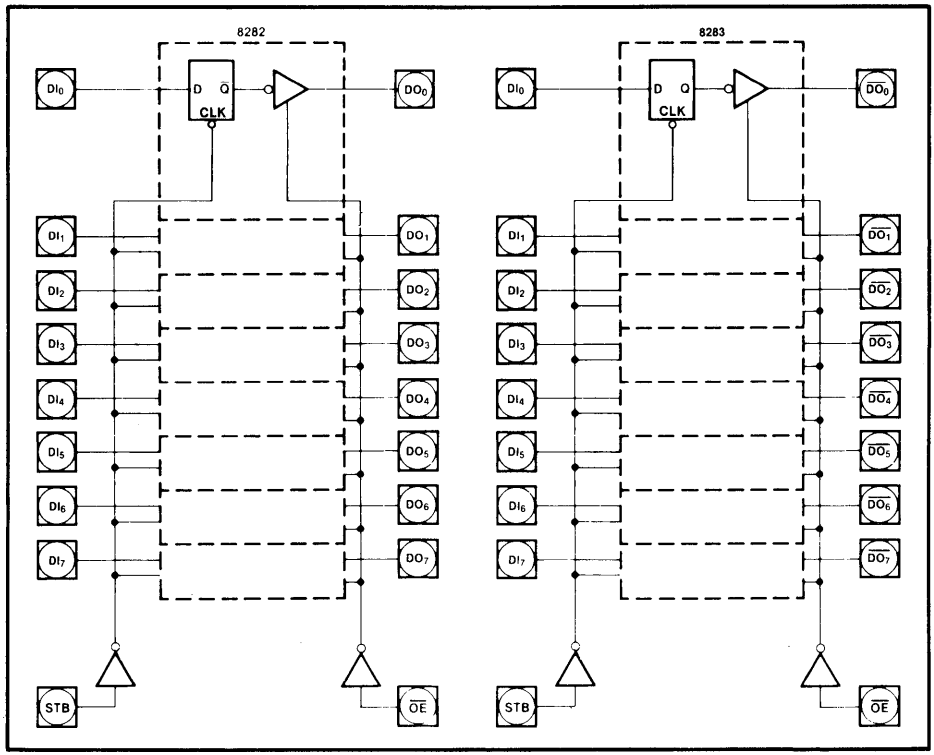


Figure 3. Logic Diagrams

PIN DEFINITIONS

Pin	Description
STB	STROBE (Input). STB is an input control pulse used to strobe data at the data input pins (A_0-A_7) into the data latches. This signal is active HIGH to admit input data. The data is latched at the HIGH to LOW transition of STB.
$\overline{OE}$	OUTPUT ENABLE (Input). $\overline{OE}$ is an input control signal which when active LOW enables the contents of the data latches onto the data output pin (B_0-B_7). OE being inactive HIGH forces the output buffers to their high impedance state.
DI_0-DI_7	DATA INPUT PINS (Input). Data presented at these pins satisfying setup time requirements when STB is strobed and latched into the data input latches.
DO_0-DO_7 (8282) $\overline{DO_0}$-$\overline{DO_7}$ (8283)	DATA OUTPUT PINS (Output). When $\overline{OE}$ is true, the data in the data latches is presented as inverted (8283) or non-inverted (8282) data onto the data output pins.

OPERATIONAL DESCRIPTION

The 8282 and 8283 octal latches are 8-bit latches with 3-state output buffers. Data having satisfied the setup time requirements is latched into the data latches by strobing the STB line HIGH to LOW. Holding the STB line in its active HIGH state makes the latches appear transparent. Data is presented to the data output pins by activating the $\overline{OE}$ input line. When $\overline{OE}$ is inactive HIGH the output buffers are in their high impedance state. Enabling or disabling the output buffers will not cause negative-going transients to appear on the data output bus.

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
All Output and Supply Voltages . . . . . −0.5V to +7V
All Input Voltages . . . . . −1.0V to +5.5V
Power Dissipation . . . . . 1 Watt

*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

Conditions: $V_{CC} = 5V \pm 10\%$, $T_A = 0°C$ to $70°C$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_C	Input Clamp Voltage		−1	V	$I_C = -5$ mA
I_{CC}	Power Supply Current		160	mA	
I_F	Forward Input Current		−0.2	mA	$V_F = 0.45V$
I_R	Reverse Input Current		50	μA	$V_R = 5.25V$
V_{OL}	Output Low Voltage		.45	V	$I_{OL} = 32$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -5$ mA
I_{OFF}	Output Off Current		±50	μA	$V_{OFF} = 0.45$ to $5.25V$
V_{IL}	Input Low Voltage		0.8	V	$V_{CC} = 5.0V$ See Note 1
V_{IH}	Input High Voltage	2.0		V	$V_{CC} = 5.0V$ See Note 1
C_{IN}	Input Capacitance		12	pF	$F = 1$ MHz $V_{BIAS} = 2.5V$, $V_{CC} = 5V$ $T_A = 25°C$

NOTE: 1. Output Loading $I_{OL} = 32$ mA, $I_{OH} = -5$ mA, $C_L = 300$ pF.

A.C. CHARACTERISTICS

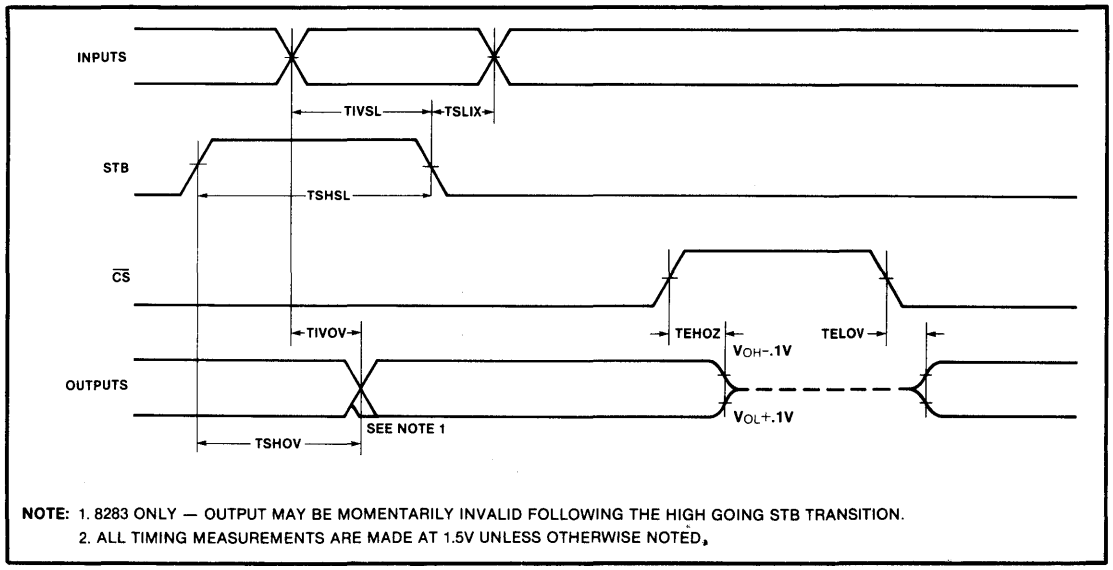
Conditions: $V_{CC} = 5V \pm 10\%$, $T_A = 0°C$ to $70°C$

Loading: Outputs — $I_{OL} = 32$ mA, $I_{OH} = -5$ mA, $C_L = 300$ pF

Symbol	Parameter	Min	Max	Units	Test Conditions
TIVOV	Input to Output Delay				(See Note 1)
	—Inverting	5	22	ns	
	—Non-Inverting	5	30	ns	
TSHOV	STB to Output Delay				
	—Inverting	10	40	ns	
	—Non-Inverting	10	45	ns	
TEHOZ	Output Disable Time	5	18	ns	
TELOV	Output Enable Time	10	30	ns	
TIVSL	Input to STB Setup Time	0		ns	
TSLIX	Input to STB Hold Time	25		ns	
TSHSL	STB High Time	15		ns	

NOTE: 1. See waveforms and test load circuit on following page.

WAVEFORMS



NOTE: 1. 8283 ONLY — OUTPUT MAY BE MOMENTARILY INVALID FOLLOWING THE HIGH GOING STB TRANSITION.
2. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.

Figure 4. Timing Diagram

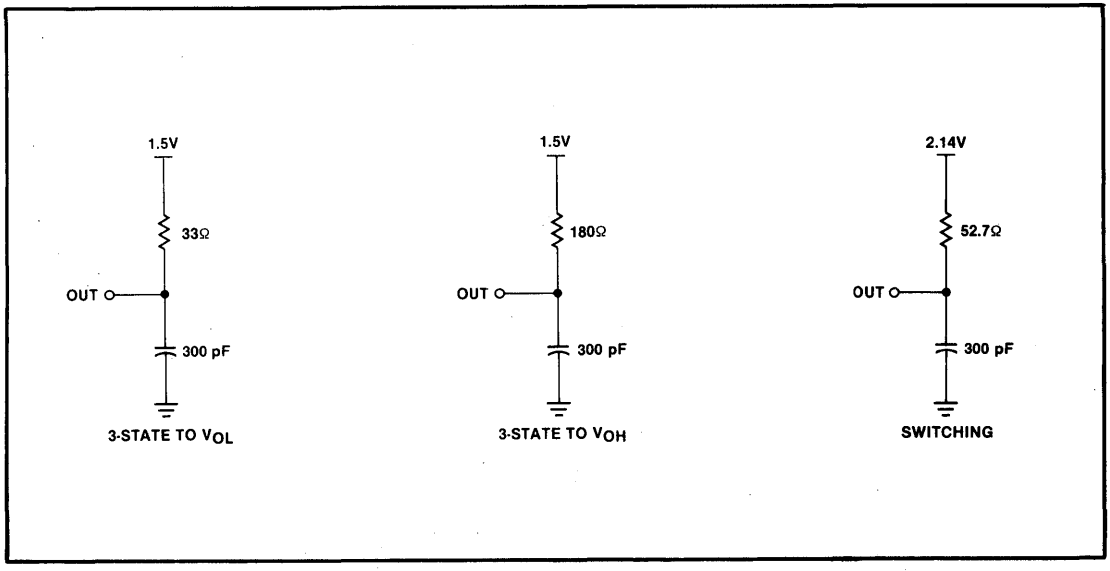


Figure 5. Output Test Load Circuits

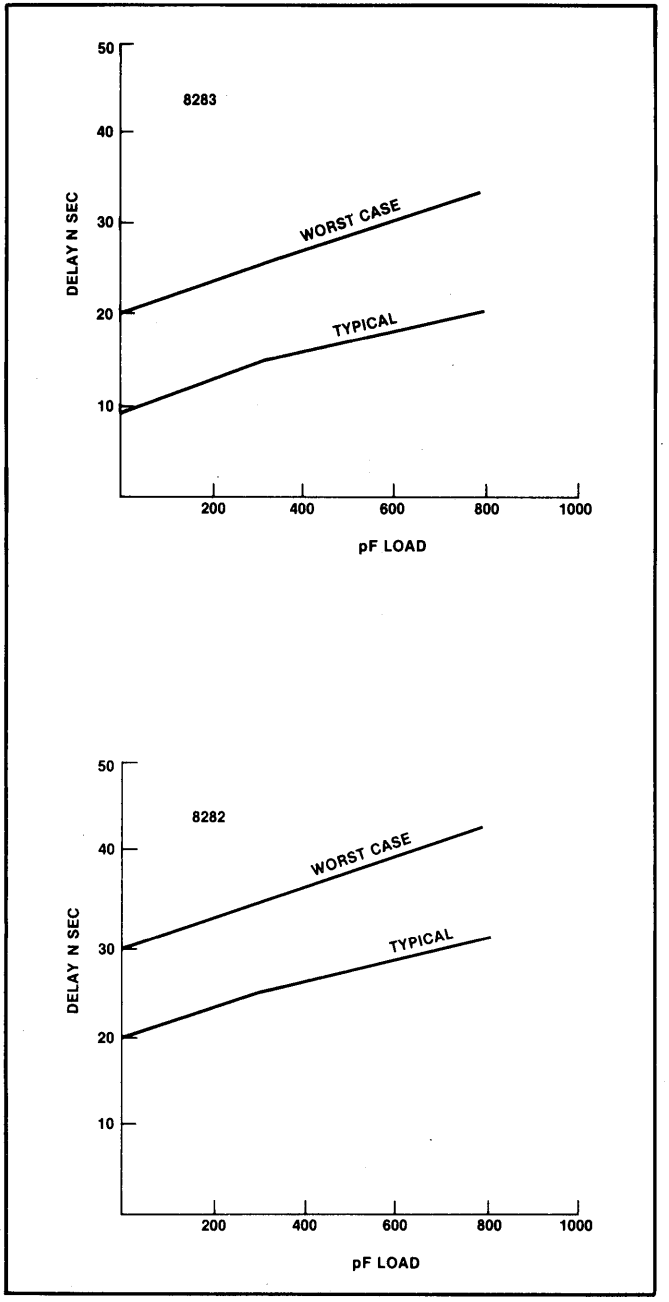


Figure 6. Output Delay vs. Capacitance

8286/8287
OCTAL BUS TRANSCEIVER

- **Data Bus Buffer Driver for iAPX 86,88, MCS-80™, MCS-85™, and MCS-48™ Families**
- **High Output Drive Capability for Driving System Data Bus**
- **Fully Parallel 8-Bit Transceivers**
- **3-State Outputs**
- **20-Pin Package with 0.3" Center**
- **No Output Low Noise when Entering or Leaving High Impedance State**

The 8286 and 8287 are 8-bit bipolar transceivers with 3-state outputs. The 8287 inverts the input data at its outputs while the 8286 does not. Thus, a wide variety of applications for buffering in microcomputer systems can be met.

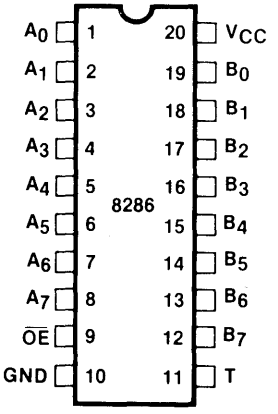


Figure 1. 8286 Pin Configuration

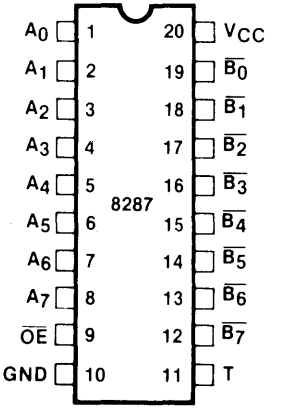


Figure 2. 8287 Pin Configuration

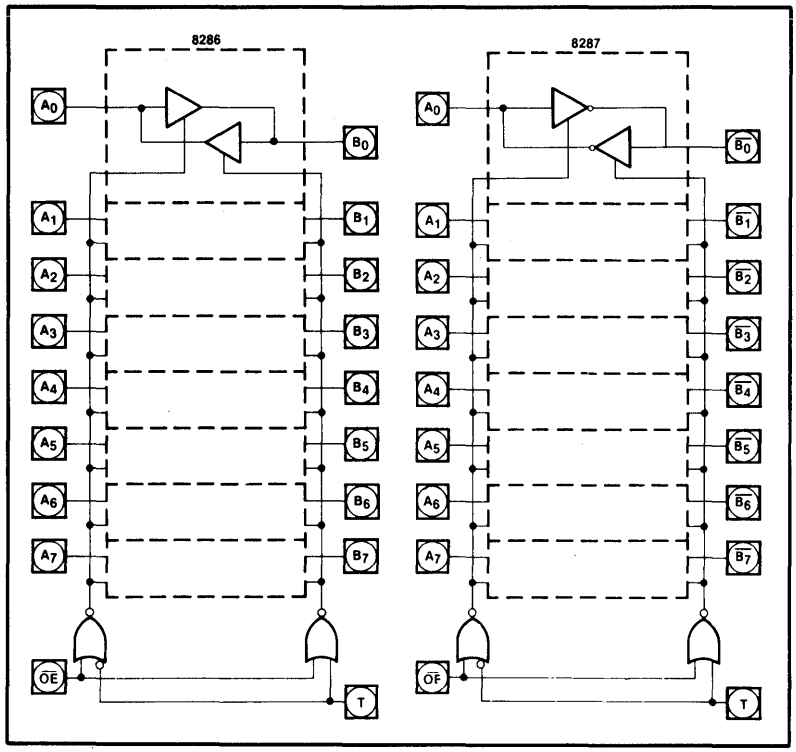


Figure 3. Logic Diagrams

Table 1. Pin Description

Pin	Description
T	TRANSMIT (Input). T is an input control signal used to control the direction of the transceivers. When HIGH, it configures the transceiver's B_0-B_7 as outputs with A_0-A_7 as inputs. T LOW configures A_0-A_7 as the outputs with B_0-B_7 serving as the inputs.
$\overline{OE}$	OUTPUT ENABLE (Input). $\overline{OE}$ is an input control signal used to enable the appropriate output driver (as selected by T) onto its respective bus. This signal is active LOW.
A_0-A_7	LOCAL BUS DATA PINS (Input/Output). These pins serve to either present data to or accept data from the processor's local bus depending upon the state of the T pin.
B_0-B_7 (8286) $\overline{B_0}$-$\overline{B_7}$ (8287)	SYSTEM BUS DATA PINS (Input/Output). These pins serve to either present data to or accept data from the system bus depending upon the state of the T pin.

FUNCTIONAL DESCRIPTION

The 8286 and 8287 transceivers are 8-bit transceivers with high impedance outputs. With T active HIGH and $\overline{OE}$ active LOW, data at the A_0-A_7 pins is driven onto the B_0-B_7 pins. With T inactive LOW and $\overline{OE}$ active LOW, data at the B_0-B_7 pins is driven onto the A_0-A_7 pins. No output low glitching will occur whenever the transceivers are entering or leaving the high impedance state.

AFN 01506A

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
All Output and Supply Voltages . . . . . −0.5V to +7V
All Input Voltages . . . . . −1.0V to +5.5V
Power Dissipation . . . . . 1 Watt

*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS FOR 8286/8287

Conditions: V_{CC} = 5V ±10% T_A = 0°C to 70°C

Symbol	Parameter	Min	Max	Units	Test Conditions
V_C	Input Clamp Voltage		-1	V	I_C = -5 mA
I_{CC}	Power Supply Current—8287 —8286		130 160	mA mA	
I_F	Forward Input Current		-0.2	mA	V_F = 0.45V
I_R	Reverse Input Current		50	μA	V_R = 5.25V
V_{OL}	Output Low Voltage —B Outputs —A Outputs		.45 .45	V V	I_{OL} = 32 mA I_{OL} = 16 mA
V_{OH}	Output High Voltage —B Outputs —A Outputs	2.4 2.4		V V	I_{OH} = -5 mA I_{OH} = -1 mA
I_{OFF} I_{OFF}	Output Off Current Output Off Current		I_F I_R		V_{OFF} = 0.45V V_{OFF} = 5.25V
V_{IL}	Input Low Voltage —A Side —B Side		0.8 0.9	V V	V_{CC} = 5.0V, See Note 1 V_{CC} = 5.0V, See Note 1
V_{IH}	Input High Voltage	2.0		V	V_{CC} = 5.0V, See Note 1
C_{IN}	Input Capacitance		12	pF	F = 1 MHz V_{BIAS} = 2.5V, V_{CC} = 5V T_A = 25°C

NOTE: 1. B Outputs — I_{OL} = 32 mA, I_{OH} = −5 mA, C_L = 300 pF; A Outputs — I_{OL} = 16 mA, I_{OH} = −1 mA, C_L = 100 pF.

A.C. CHARACTERISTICS FOR 8286/8287

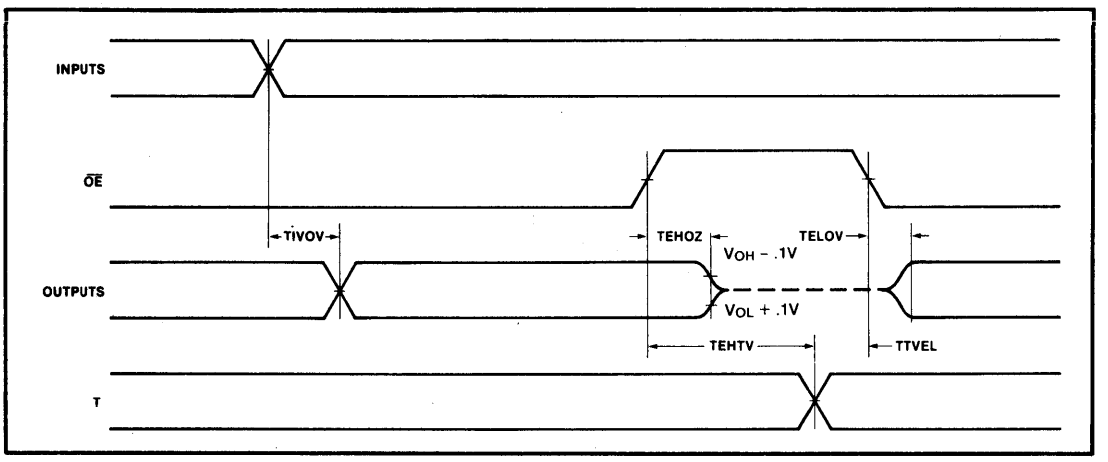
Conditions: V_{CC} = 5V ±10%, T_A = 0°C to 70°C

Loading: B Outputs — I_{OL} = 32 mA, I_{OH} = -5 mA, C_L = 300 pF
A Outputs — I_{OL} = 16 mA, I_{OH} = -1 mA, C_L = 100 pF

Symbol	Parameter	Min	Max	Units	Test Conditions
TIVOV	Input to Output Delay Inverting Non-Inverting	 5 5	 22 30	 ns ns	(See Note 1)
TEHTV	Transmit/Receive Hold Time	5		ns	
TTVEL	Transmit/Receive Setup	10		ns	
TEHOZ	Output Disable Time	5	18	ns	
TELOV	Output Enable Time	10	30	ns	

NOTE: 1. See waveforms and test load circuit on following page.

WAVEFORMS



NOTE: 1. All timing measurements are made at 1.5V unless otherwise noted.

Figure 4. 8286/8287 Timing

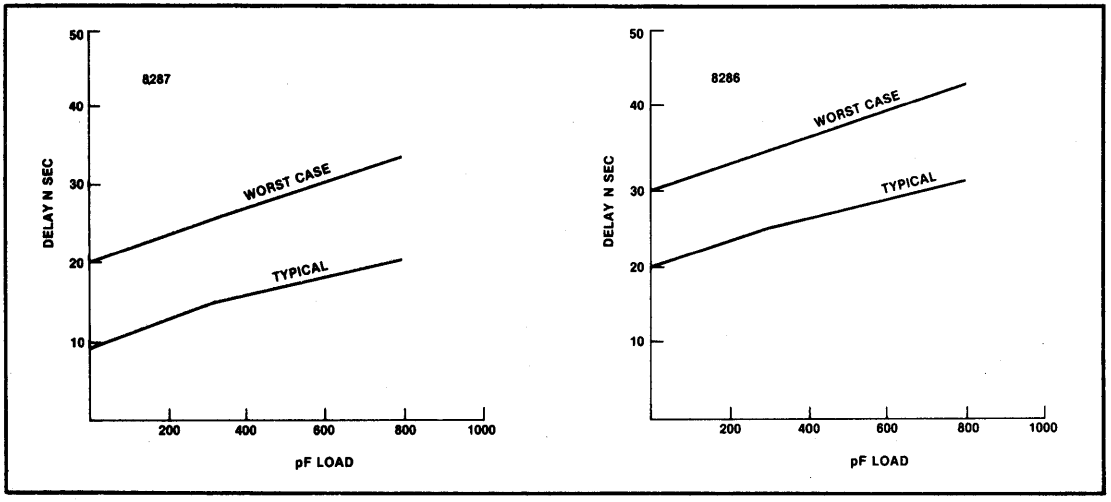


Figure 5. Output Delay vs. Capacitance

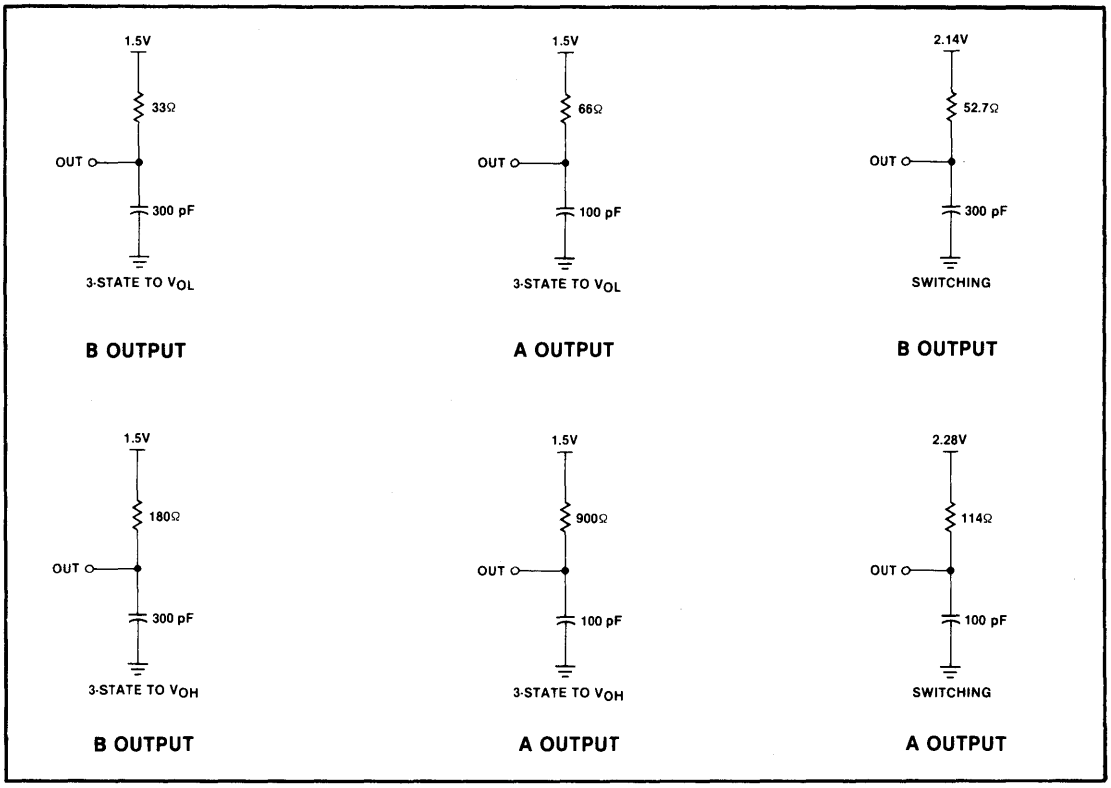


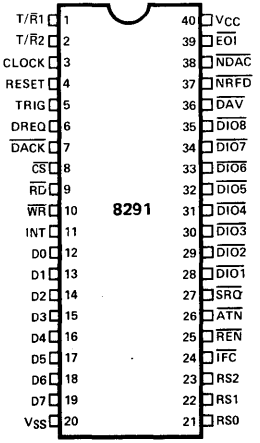
Figure 5. Test Load Circuits

8291
GPIB TALKER/LISTENER

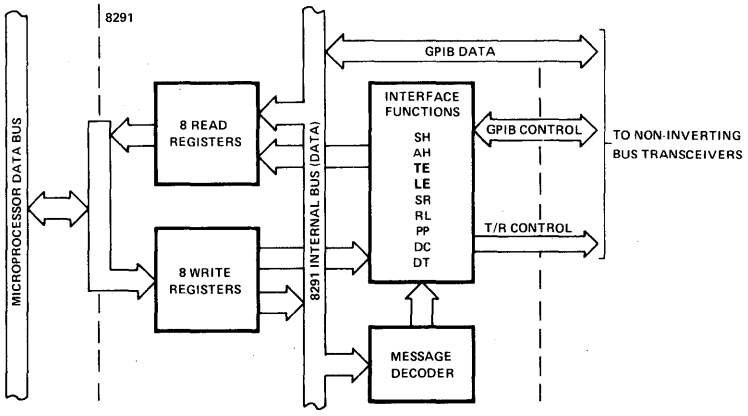
- Designed to Interface Microprocessors (e.g., 8080, 8085, 8086, 8048) to an IEEE Standard 488 Digital Interface Bus
- Programmable Data Transfer Rate
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with Extended Addressing
- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local Functions
- Selectable Interrupts
- On-Chip Primary and Secondary Address Recognition
- Automatic Handling of Addressing and Handshake Protocol
- Provision for Software Implementation of Additional Features
- 1 – 8 MHz Clock Range
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly Interfaces to External Non-Inverting Transceivers for Connection to the GPIB
- Provides Three Addressing Modes, Allowing the Chip to be Addressed Either as a Major or a Minor Talker/Listener with Primary or Secondary Addressing
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention
- Trigger Output Pin
- On-Chip EOS (End of Sequence) Message Recognition Facilitates Handling of Multi-Byte Transfers

The 8291 GPIB Talker/Listener is a microprocessor-controlled chip designed to interface microprocessors (e.g., 8048, 8080, 8085, 8086) to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's interface functions except for the controller.

PIN CONFIGURATION



BLOCK DIAGRAM

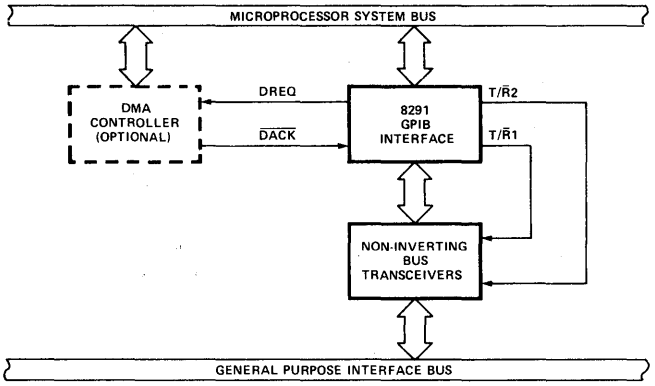


PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
D_0-D_7	I/O	12-19	Data bus port, to be connected to microprocessor data bus.
RS_0-RS_2	I	21-23	Register select inputs, to be connected to three non-multiplexed microprocessor address bus lines. Select which of the 8 internal read (write) registers will be read from (written into) with the execution of $\overline{RD}$ ($\overline{WR}$).
$\overline{CS}$	I	8	Chip select. When low, enables reading from or writing into the register selected by RS_0-RS_2.
$\overline{RD}$	I	9	Read strobe. When low, selected register contents are read by the CPU.
$\overline{WR}$	I	10	Write strobe. When low, data is written into the selected register.
INT ($\overline{INT}$)	O	11	Interrupt request to the microprocessor, set high for request and cleared when the appropriate register is accessed by the CPU. May be software configured to be active low.
DREQ	O	6	DMA request, normally low, set high to indicate byte output or byte input, in DMA mode; reset by $\overline{DACK}$.
$\overline{DACK}$	I	7	DMA acknowledge. When low, resets DREQ and selects data in/data out register for DMA data transfer (actual transfer done by $\overline{RD}/\overline{WR}$ pulse). Must be high if DMA is not used.
TRIG	O	5	Trigger output, normally low; generates a triggering pulse with 1μsec min. width in response to the GET bus command or Trigger auxiliary command.
CLOCK	I	3	External clock input, used only for T_1 delay generator. May be any speed in 1-8 MHz range.
RESET	I	4	Reset input. When high, forces the device into an "Idle" (initialization) mode. The device will remain at "Idle" until released by the microprocessor.
$\overline{DIO_1}$-$\overline{DIO_8}$	I/O	28-35	8-bit GPIB data port, used for bidirectional data byte transfer between 8291 and GPIB via non-inverting external line transceivers.
$\overline{DAV}$	I/O	36	Data valid; GPIB handshake control line. Indicates the availability and validity of information on the DIO lines.
$\overline{NRFD}$	I/O	37	Not ready for data; GPIB handshake control line. Indicates the condition of readiness of device(s) connected to the bus to accept data.
$\overline{NDAC}$	I/O	38	Not data accepted; GPIB handshake control line. Indicates the condition of acceptance of data by the device(s) connected to the bus.
$\overline{ATN}$	I	26	Attention; GPIB command line. Specifies how data on DIO lines are to be interpreted.
$\overline{IFC}$	I	24	Interface clear; GPIB command line. Places the interface functions in a known quiescent state.
$\overline{SRQ}$	O	27	Service request; GPIB command line. Indicates the need for attention and requests an interruption of the current sequence of events on the GPIB.
$\overline{REN}$	I	25	Remote enable; GPIB command line. Selects (in conjunction with other messages) remote or local control of the device.
$\overline{EOI}$	I/O	39	End or identify; GPIB command line. Indicates the end of a multiple byte transfer sequence or, in conjunction with ATN, addresses the device during a polling sequence.
T/$\overline{R}$1	O	1	External transceivers control line. Set high to indicate output data/signals on the DIO_1-DIO_8 and DAV lines and input signals on the NRFD and NDAC lines (active source handshake). Set low to indicate input data/signals on the DIO_1-DIO_8 and DAV lines and output signals on the NRFD and NDAC lines (active acceptor handshake).
T/$\overline{R}$2	O	2	External transceivers control line. Set high to indicate output signals on the EOI line. Set low to indicate expected input signal on the EOI line during parallel poll.
V_{CC}	P.S.	40	Positive power supply (5V ± 10%).
GND	P.S.	20	Potential ground circuit.

Note: All signals on the 8291 pins are specified with positive logic. However, IEEE 488 specifies negative logic on its 16 signal lines. Thus, the data is inverted once from D_0-D_7 to $\overline{DIO_1}$-$\overline{DIO_8}$ and non-inverting bus transcelvers should be used.

8291 SYSTEM DIAGRAM



THE GENERAL PURPOSE INTERFACE BUS (GPIB)

The General Purpose Interface Bus (GPIB) is defined in the IEEE Standard 488-1978 "Digital Interface for Programmable Instrumentation." Although a knowledge of this standard is assumed, Figure 1 provides the bus structure for quick reference. Also, Tables 1 and 2 reference the interface state mnemonics and the interface messages respectively. Modified state diagrams for the 8291 are presented in Appendix A.

GENERAL DESCRIPTION

The 8291 is a microprocessor controlled device designed to interface microprocessors e.g., 8048, 8080, 8085, 8086 to the GPIB. It implements all of the interface functions defined in the IEEE 488 Standard. If an implementation of the Standard's Controller function is desired, it can be connected with an Intel® 8292 to form a complete interface.

The 8291 handles communication between a microprocessor controlled device and the GPIB. Its capabilities include data transfer, handshake protocol, talker/listener addressing procedures, device clearing and triggering, service request, and both serial and parallel polling schemes. In most procedures, it does not disturb the microprocessor unless a byte is waiting on input or a byte sent on output (output buffer empty).

The 8291 architecture includes 16 registers. Eight of these registers may be written into by the microprocessor. The other eight registers may be read by the microprocessor. One each of these read and write registers is for direct data transfers. The rest of the write registers control the various features of the chip, while the rest of the read registers provide the microprocessor with a monitor of GPIB states, various bus conditions, and device conditions.

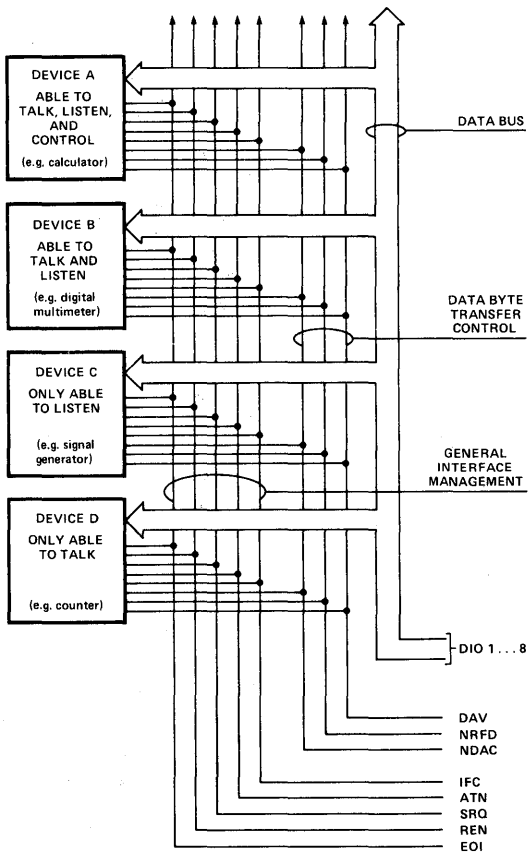


Figure 1. Interface Capabilities and Bus Structure.

GPIB Addressing

Each device connected to the GPIB must have at least one address whereby the controller device in charge of the bus can configure it to talk, listen, or send status. An 8291 implementation of the GPIB offers the user three addressing modes from which the device can be initialized for each application. The first of these modes allows for the device to have two separate primary addresses. The second mode allows the user to implement a single talker/listener with a two byte address (primary address + secondary address). The third mode again allows for two distinct addresses but in this instance, they can each have a two-byte address. However, this mode requires that the secondary addresses be passed to the microprocessor for verification. These three addressing schemes are described in more detail in the discussion of the Address registers.

TABLE 1.
IEEE 488 INTERFACE STATE MNEMONICS

Mnemonic	State Represented
ACDS	Accept Data State
ACRS	Acceptor Ready State
AIDS	Acceptor Idle State
ANRS	Acceptor Not Ready State
APRS	Affirmative Poll Response State
AWNS	Acceptor Wait for New Cycle State
CACS	Controller Active State
CADS	Controller Addressed State
CAWS	Controller Active Wait State
CIDS	Controller Idle State
CPPS	Controller Parallel Poll State
CPWS	Controller Parallel Poll Wait State
CSBS	Controller Standby State
CSNS	Controller Service Not Requested State
CSRS	Controller Service Requested State
CSWS	Controller Synchronous Wait State
CTRS	Controller Transfer State
DCAS	Device Clear Active State
DCIS	Device Clear Idle State
DTAS	Device Trigger Active State
DTIS	Device Trigger Idle State
LACS	Listener Active State
LADS	Listener Addressed State
LIDS	Listener Idle State
LOCS	Local State
LPAS	Listener Primary Addressed State
LPIS	Listener Primary Idle State
LWLS	Local With Lockout State
NPRS	Negative Poll Response State

Mnemonic	State Represented
PACS	Parallel Poll Addressed to Configure State
PPAS	Parallel Poll Active State
PPIS	Parallel Poll Idle State
PPSS	Parallel Poll Standby State
PUCS	Parallel Poll Unaddressed to Configure State
REMS	Remote State
RWLS	Remote With Lockout State
SACS	System Control Active State
SDYS	Source Delay State
SGNS	Source Generate State
SIAS	System Control Interface Clear Active State
SIDS	Source Idle State
SIIS	System Control Interface Clear Idle State
SINS	System Control Interface Clear Not Active State
SIWS	Source Idle Wait State
SNAS	System Control Not Active State
SPAS	Serial Poll Active State
SPIS	Serial Poll Idle State
SPMS	Serial Poll Mode State
SRAS	System Control Remote Enable Active State
SRIS	System Control Remote Enable Idle State
SRNS	System Control Remote Enable Not Active State
SRQS	Service Request State
STRS	Source Transfer State
SWNS	Source Wait for New Cycle State
TACS	Talker Active State
TADS	Talker Addressed State
TIDS	Talker Idle State
TPIS	Talker Primary Idle State

– – – – – – The Controller function is implemented on the Intel® 8292.

TABLE 2.
IEEE 488 INTERFACE MESSAGE REFERENCE LIST

Mnemonic	Message	Interface Function(s)
LOCAL MESSAGES RECEIVED (By Interface Functions)		
*gts	go to standby	C
ist	individual status	PP
lon	listen only	L, LE
lpe	local poll enable	PP
nba	new byte available	SH
pon	power on	SH,AH,T,TE,L,LE,SR,RL,PP,C
rdy	ready	AH
*rpp	request parallel poll	C
*rsc	request system control	C
rsv	request service	SR
rtl	return to local	RL
*sic	send interface clear	C
*sre	send remote enable	C
*tca	take control asynchronously	C
*tcs	take control synchronously	AH, C
ton	talk only	T, TE
REMOTE MESSAGES RECEIVED		
ATN	Attention	SH,AH,T,TE,L,LE,PP,C
DAB	Data Byte	(Via L, LE)
DAC	Data Accepted	SH
DAV	Data Valid	AH
DCL	Device Clear	DC
END	End	(via L, LE)
GET	Group Execute Trigger	DT
GTL	Go to Local	RL
IDY	Identify	L,LE,PP
IFC	Interface Clear	T,TE,L,LE,C
LLO	Local Lockout	RL
MLA	My Listen Address	L,LE,RL,T,TE
MSA	My Secondary Address	TE,LE,RL
MTA	My Talk Address	T,TE,L,LE
OSA	Other Secondary Address	TE
OTA	Other Talk Address	T, TE
PCG	Primary Command Group	TE,LE,PP
†PPC	Parallel Poll Configure	PP
†[PPD]	Parallel Poll Disable	PP
†[PPE]	Parallel Poll Enable	PP
*PPR_N	Parallel Poll Response N	(via C)
†PPU	Parallel Poll Unconfigure	PP
REN	Remote Enable	RL
RFD	Ready for Data	SH
RQS	Request Service	(via L, LE)
[SDC]	Select Device Clear	DC
SPD	Serial Poll Disable	T, TE
SPE	Serial Poll Enable	T, TE
*SQR	Service Request	(via C)
STB	Status Byte	(via L, LE)
*TCT or [TCT]	Take Control	C
UNL	Unlisten	L, LE

*These messages are handled only by Intel's 8292.
†Undefined commands which may be passed to the microprocessor.

TABLE 2. (Cont'd)
IEEE 488 INTERFACE MESSAGE REFERENCE LIST

Mnemonic	Message	** Interface Function(s)
REMOTE MESSAGES SENT		
ATN	Attention	C
DAB	Data Byte	(via T, TE)
DAC	Data Accepted	AH
DAV	Data Valid	SH
DCL	Device Clear	(via C)
END	End	(via T)
GET	Group Execute Trigger	(via C)
GTL	Go to Local	(via C)
IDY	Identify	C
IFC	Interface Clear	C
LLO	Local Lockout	(via C)
MLA or [MLA]	My Listen Address	(via C)
MSA or [MSA]	My Secondary Address	(via C)
MTA or [MTA]	My Talk Address	(via C)
OSA	Other Secondary Address	(via C)
OTA	Other Talk Address	(via C)
PCG	Primary Command Group	(via C)
PPC	Parallel Poll Configure	(via C)
[PPD]	Parallel Poll Disable	(via C)
[PPE]	Parallel Poll Enable	(via C)
PPR_N	Parallel Poll Response N	PP
PPU	Parallel Poll Unconfigure	(via C)
REN	Remote Enable	C
RFD	Ready for Data	AH
RQS	Request Service	T, TE
[SDC]	Selected Device Clear	(via C)
SPD	Serial Poll Disable	(via C)
SPE	Serial Poll Enable	(via C)
SRQ	Service Request	SR
STB	Status Byte	(via T, TE)
TCT	Take Control	(via C)
UNL	Unlisten	(via C)

**All Controller messages must be sent via Intel's 8292.

DEVICE ELECTRICAL CHARACTERISTICS

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5V ± 10%

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL}=2mA (4mA for TR1 pin)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA (-150μA for SRQ pin)
V_{OH-INT}	Interrupt Output High Voltage	2.4		V	I_{OH}=-400μA
		3.5		V	I_{OH}=-50μA
I_{IL}	Input Leakage		10	μA	V_{IN}=0V to V_{CC}
I_{LOL}	Output Leakage Current		-10	μA	V_{OUT}=0.45V
I_{LOH}	Output Leakage Current		10	μA	V_{OUT}=V_{CC}
I_{CC}	V_{CC} Supply Current		180	mA	T_A=0°C

A.C. CHARACTERISTICS

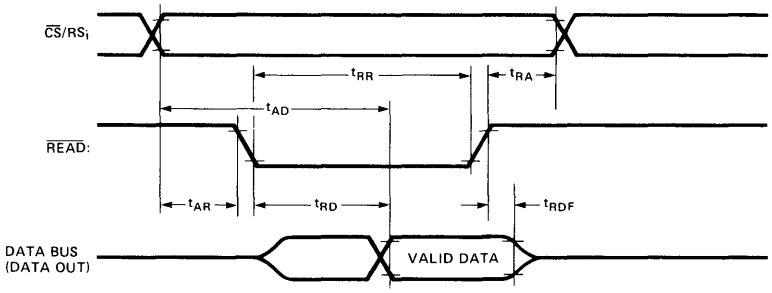
V_{CC} = 5V ± 10%, Commercial: T_A = 0°C to 70°C

Symbol	Parameter	Min.	Max.	Unit
t_{AR}	Address Stable Before $\overline{READ}$	0		nsec[1]
t_{RA}	Address Hold After $\overline{READ}$	0		nsec[1]
t_{RR}	$\overline{READ}$ width	140		nsec[2]
t_{AD}	Address Stable to Data Valid		250	nsec[1]
t_{RD}	$\overline{READ}$ to Data Valid		100	nsec[2]
t_{RDF}	Data Float After $\overline{READ}$	0	60[2]	nsec
t_{AW}	Address Stable Before $\overline{WRITE}$	0		nsec[1]
t_{WA}	Address Hold After $\overline{WRITE}$	0		
t_{WW}	$\overline{WRITE}$ Width	170		nsec[1]
t_{DW}	Data Set Up Time to the Trailing Edge of $\overline{WRITE}$	150		nsec[1]
t_{WD}	Data Hold Time After $\overline{WRITE}$	0		nsec[1]
t_{AKRQ}	$\overline{DACK}$↓ to DREQ↓		130	nsec
t_{DKDA6}	$\overline{DACK}$↓ to Up Data Valid		200	nsec

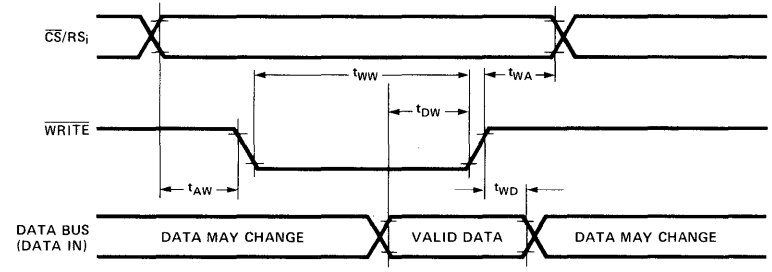
Notes:
1. 8080 System C_{Lmax} = 100pF; C_{Lmin} = 15pF; 3 MHz clock.
2. 8085 System C_L = 150pF; 4 MHz clock.

TIMING WAVEFORMS

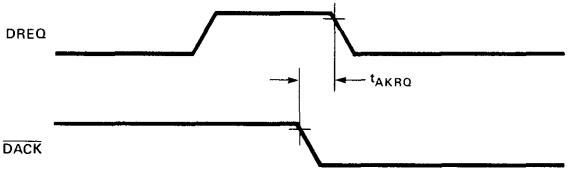
READ



WRITE



DMA



GPIB TIMINGS[1]

Symbol	Parameter	Max.	Unit	Test Conditions
TEOT13	$\overline{EOI}$↓ to TR1↑	135	nsec	PPSS, $\overline{ATN}$=0.45V
TEODI6	$\overline{EOI}$↓ to $\overline{DIO}$ Valid	155	nsec	PPSS, $\overline{ATN}$=0.45V
TEOT12	$\overline{EOI}$↓ to TR1↓	155	nsec	PPSS, $\overline{ATN}$=0.45V
TATND4	$\overline{ATN}$↓ to $\overline{NDAC}$↓	155	nsec	TACS, AIDS
TATT14	$\overline{ATN}$↓ to TR1↓	155	nsec	TACS, AIDS
TATT24	$\overline{ATN}$↓ to TR2↓	155	nsec	TACS, AIDS
TDVND3-C	$\overline{DAV}$↓ to $\overline{NDAC}$↑	650	nsec	AH, CACS
TNDDV1	$\overline{NDAC}$↑ to $\overline{DAV}$↑	350	nsec	SH, STRS
TNRDV2	$\overline{NRFD}$↑ to $\overline{DAV}$↓	350	nsec	SH, T1 True
TNDDR1	$\overline{NDAC}$↑ to DREQ↑	400	nsec	SH
TDVDR3	$\overline{DAV}$↓ to DREQ↑	600	nsec	AH, LACS, $\overline{ATN}$=2.4V
TDVND2-C	$\overline{DAV}$↑ to $\overline{NDAC}$↓	350	nsec	AH,LACS
TDVNR1-C	$\overline{DAV}$↑ to $\overline{NRFD}$↑	350	nsec	AH, LACS, rdy=True
TRDNR3	$\overline{RD}$↓ to $\overline{NRFD}$↑	500	nsec	AH, LACS
TWRDI5	$\overline{WR}$ to $\overline{DIO}$ Valid	250	nsec	SH, TACS, RS = 0.4V
TWRDV2	$\overline{WR}$↑ to $\overline{DAV}$↓	830 + t_{SYNC}	nsec	High Speed Transfers Enabled, $N_F = f_C$, $t_{SYNC} = 1/2 \cdot f_C$

Notes:
1. All GPIB timings are at the pins of the 8291.

Appendix A

MODIFIED STATE DIAGRAMS

Figure A.1 presents the interface function state diagrams. It is derived from IEEE Std. state diagrams, with the following changes:

A. The 8291 supports the complete set of IEEE-488 interface functions except for the controller. These include: SH1, AH1, T5, TE5, L3, LE3, SR1, RL1, PP1, DC1, DT1, and C0.

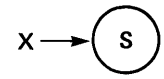
B. Addressing modes included in T,L state diagrams.

Note that in Mode 3, MSA, OSA are generated only after secondary address validity check by the microprocessor (APT interrupt).

C. In these modified state diagrams, the IEEE-488 convention of low true logic is followed. Thus, DAV is logically true at <0.8V and is equivalent to pin 36 on the 8291.

D. All remote multiline messages decoded are conditioned by ACDS. The multiplication by ACDS is not drawn to simplify the diagrams.

E. The symbol



indicates:

1. When event X occurs, the function will return to state S.
2. X overrides any other transition condition in the function.

Statement 2 simplifies the diagram, avoiding the explicit use of $\overline{X}$ to condition all transitions from S to other states.

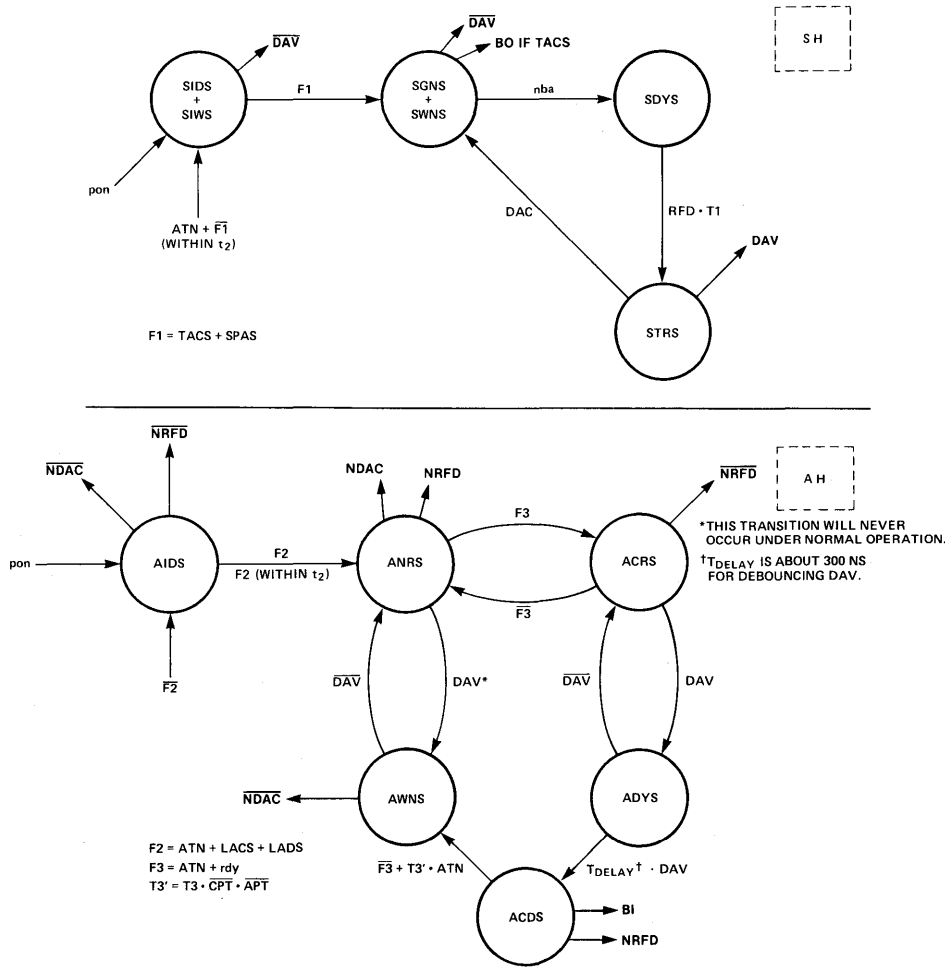


Figure A.1. 8291 State Diagrams (Continued next page)

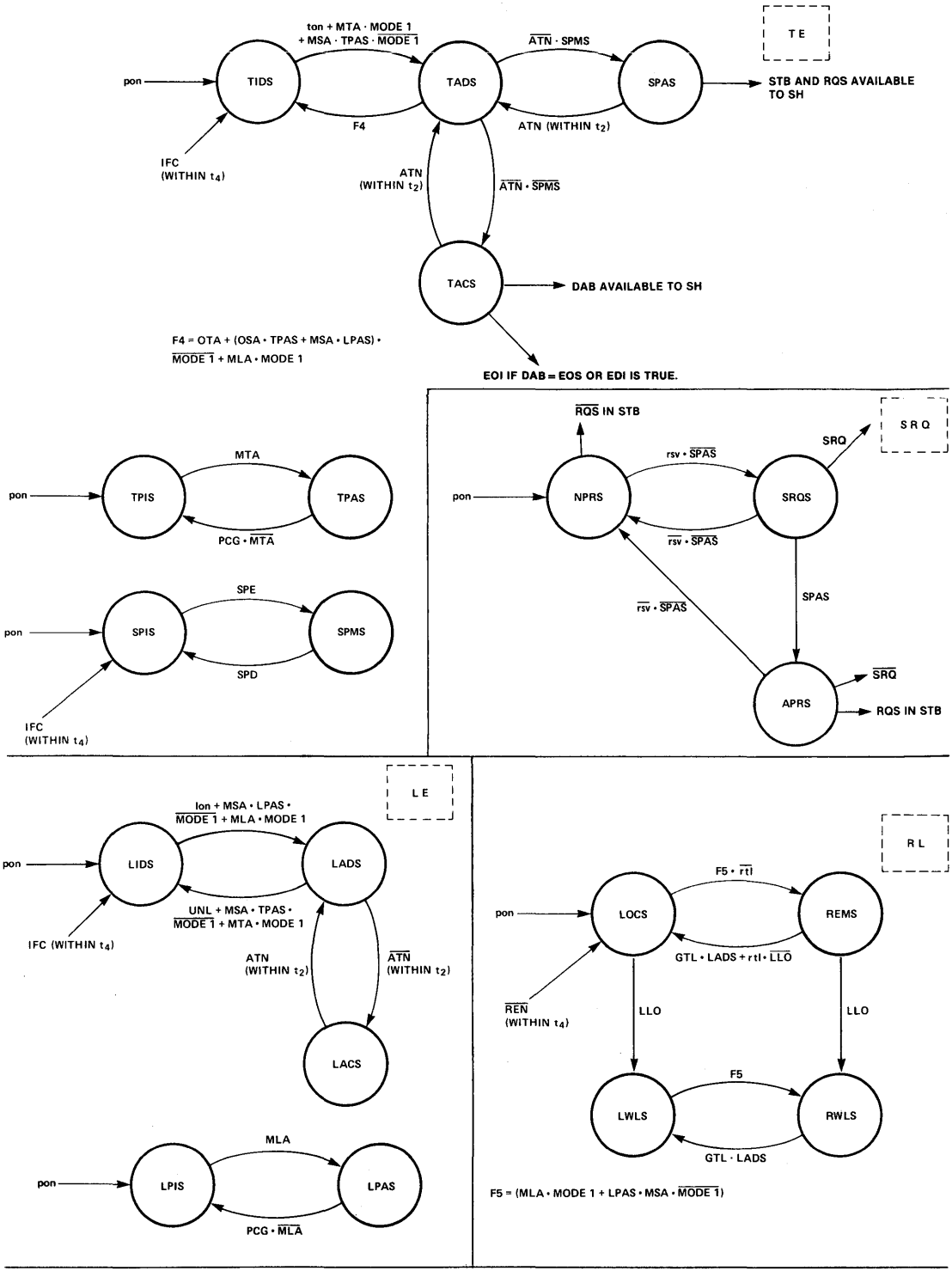


Figure A.1. 8291 State Diagrams (Continued next page)

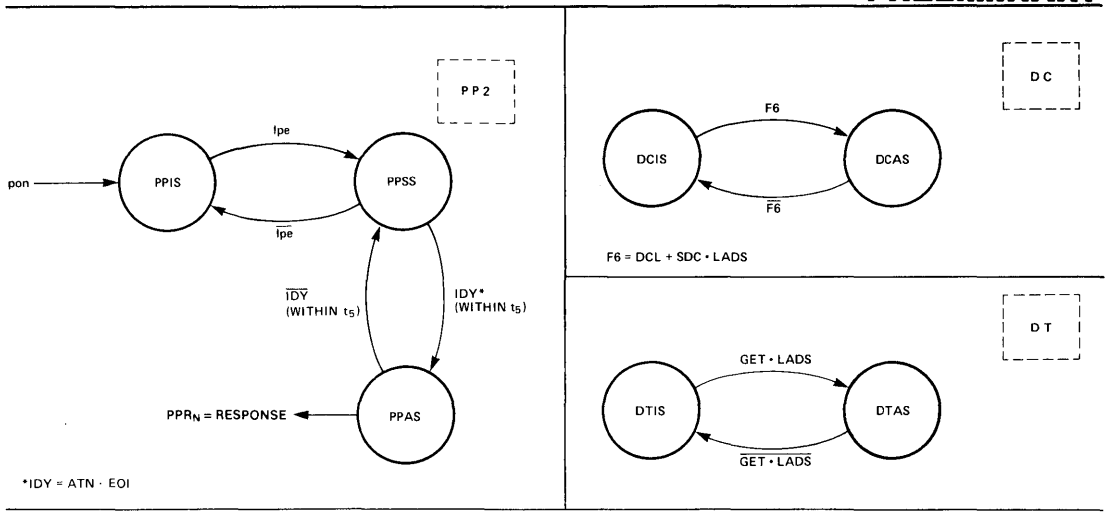


Figure A.1. 8291 State Diagrams

Appendix B

IEEE 488 TIME VALUES

Time Value Identifier*	Function (Applies to)	Description	Value
T_1	SH	Settling Time for Multiline Messages	≥ 2μs†
t_2	LC,$\overline{IC}$,SH,AH,T,L	Response to ATN	≤ 200ns
T_3	AH	Interface Message Accept Time ‡	> 0 δ
t_4	T,TE,L,LE,C,CE	Response to IFC or REN False	< 100μs
t_5	PP	Response to ATN+EOI	≤ 200ns
T_6	C	Parallel Poll Execution Time	≥ 2μs
T_7	C	Controller Delay to Allow Current Talker to see ATN Message	≥ 500ns
T_8	C	Length of IFC or REN False	> 100μs
T_9	C	Delay for EOI**	≥ 1.5μs††

* Time values specified by a lower case t indicate the maximum time allowed to make a state transition. Time values specified by an upper case T indicate the minimum time that a function must remain in a state before exiting.

† If three-state drivers are used on the DIO, DAV, and EOI lines, T_1 may be:
1. ≥ 1100ns
2. Or ≥ 700ns if it is known that within the controller ATN is driven by a three-state driver.
3. Or ≥ 500ns for all subsequent bytes following the first sent after each false transition of ATN (the first byte must be sent in accordance with (1) or (2).
4. Or ≥ 350ns for all subsequent bytes following the first sent after each false transition of ATN under conditions specified in Section 5.2.3 and warning note. See IEEE Standard 488.

‡ Time required for interface functions to accept, not necessarily respond to interface messages.

δ Implementation independent.

** Delay required for EOI, NDAC, and NRFD signal lines to indicate valid states.

†† ≥ 600ns for three-state drivers.

Appendix C
THE THREE WIRE HANDSHAKE

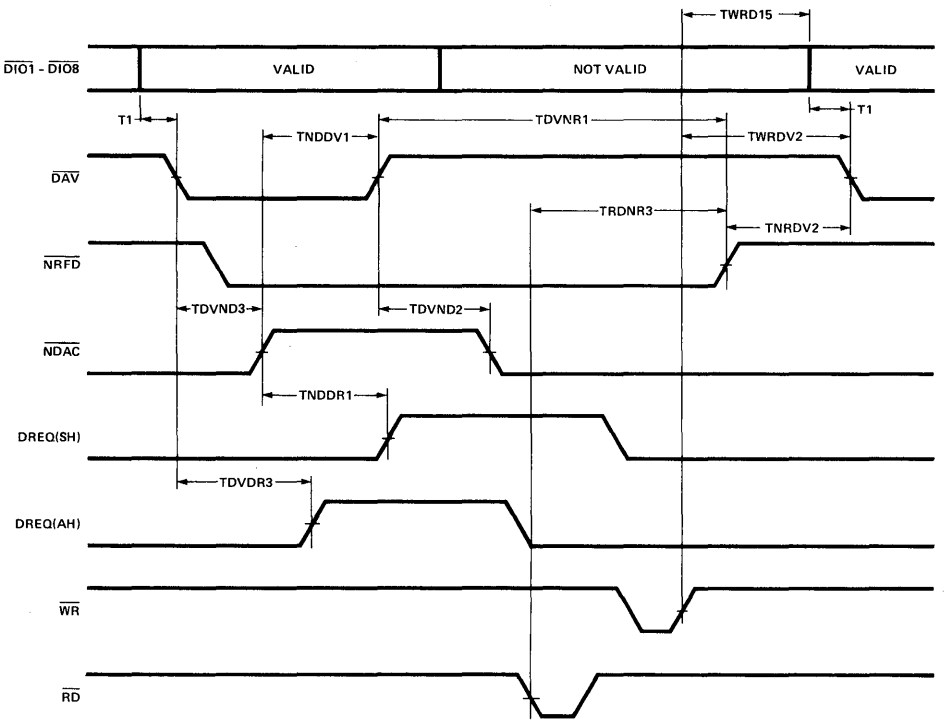
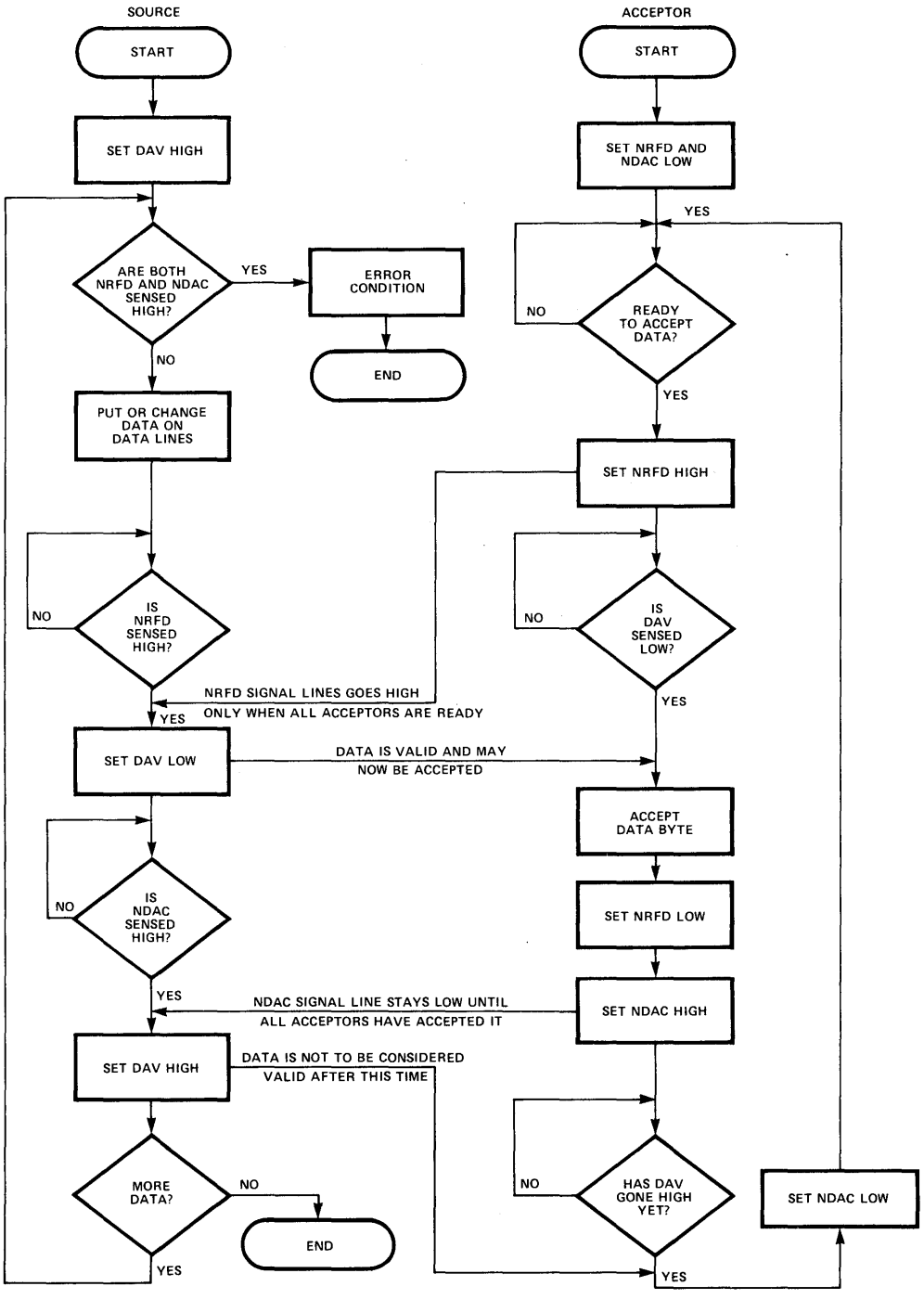


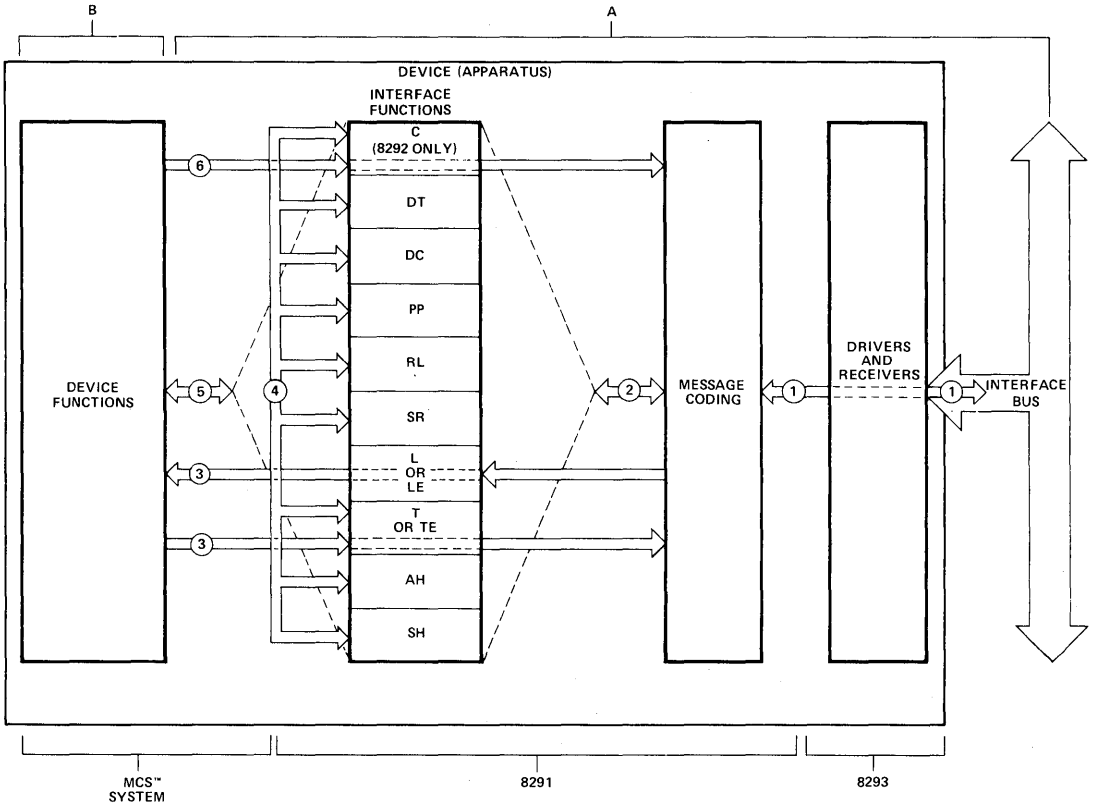
Figure C-1. 3-Wire Handshake Timing at 8291.



FLOW DIAGRAM OUTLINES SEQUENCE OF EVENTS DURING TRANSFER OF DATA BYTE. MORE THAN ONE LISTENER AT A TIME CAN ACCEPT DATA BECAUSE OF LOGICAL AND CONNECTION OF NRFD AND NDAC LINES.

Figure C.2. Handshake Flowchart.

Appendix D
FUNCTIONAL PARTITIONS



A – CAPABILITY DEFINED BY THE 488-1978 STANDARD.
B – CAPABILITY DEFINED BY THE DESIGNER.
1 – INTERFACE BUS SIGNAL LINES.
2 – REMOTE INTERFACE MESSAGES TO AND FROM INTERFACE FUNCTIONS.
3 – DEVICE DEPENDENT MESSAGES TO AND FROM DEVICE FUNCTIONS.
4 – STATE LINKAGES BETWEEN INTERFACE FUNCTIONS.
5 – LOCAL MESSAGES BETWEEN DEVICE FUNCTIONS AND INTERFACE FUNCTIONS (MESSAGES TO INTERFACE FUNCTIONS ARE DEFINED, MESSAGES FROM INTERFACE FUNCTIONS EXIST ACCORDING TO THE DESIGNER'S CHOICE).
6 – CONTROL MESSAGES (8292 ONLY).

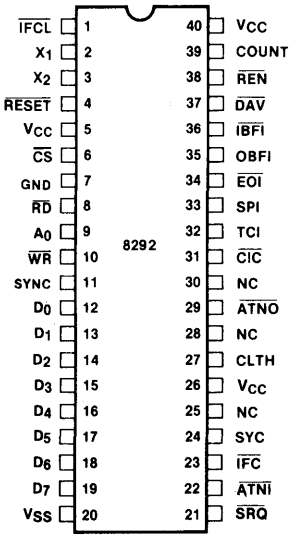
Figure D.1. Functional Partition Within a Device.

8292
GPIB CONTROLLER

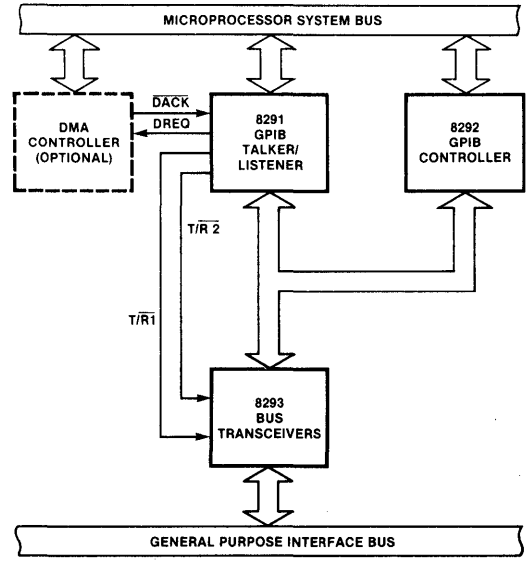
- **Complete IEEE Standard 488 Controller Function**
- **Interface Clear (IFC) Sending Capability Allows Seizure of Bus Control and/or Initialization of the Bus**
- **Responds to Service Requests (SRQ)**
- **Sends Remote Enable (REN), Allowing Instruments to Switch to Remote Control**
- **Complete Implementation of Transfer Control Protocol**
- **Synchronous Control Seizure Prevents the Destruction of Any Data Transmission in Progress**
- **Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller**

The 8292 GPIB Controller is a microprocessor-controlled chip designed to function with the 8291 GPIB Talker/Listener to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a pre-programmed Intel® 8041A.

PIN CONFIGURATION



8291, 8292 SYSTEM DIAGRAM



00741C

PIN DESCRIPTION

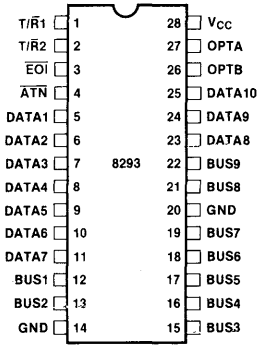
Symbol	I/O	Pin No.	Function
$\overline{IFCL}$	I	1	IFC Received (latched) — The 8292 monitors the IFC Line (when not system controller) through this pin.
X_1, X_2	I	2, 3	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
$\overline{RESET}$	I	4	Used to initialize the chip to a known state during power on.
$\overline{CS}$	I	6	Chip Select Input — Used to select the 8292 from other devices on the common data bus.
$\overline{RD}$	I	8	I/O write input which allows the master CPU to read from the 8292.
A_0	I	9	Address Line — Used to select between the data bus and the status register during read operations and to distinguish between data and commands written into the 8292 during write operations.
$\overline{WR}$	I	10	I/O read input which allows the master CPU to write to the 8292.
SYNC	O	11	8041A instruction cycle synchronization signal; it is an output clock with a frequency of XTAL ÷ 15.
D_0-D_7	I/O	12-19	8 bidirectional lines used for communication between the central processor and the 8292's data bus buffers and status register.
V_{SS}	P.S.	7, 20	Circuit ground potential.
$\overline{SRQ}$	I	21	Service Request — One of the IEEE control lines. Sampled by the 8292 when it is controller in charge. If true, SPI interrupt to the master will be generated.
$\overline{ATNI}$	I	22	Attention In — Used by the 8292 to monitor the GPIB ATN control line. It is used during the transfer control procedure.
$\overline{IFC}$	I/O	23	Interface Clear — One of the GPIB management lines, as defined by IEEE Std. 488-1978, places all devices in a known quiescent state.
SYC	I	24	System Controller — Monitors the system controller switch.
CLTH	O	27	CLEAR LATCH Output — Used to clear the $\overline{IFCR}$ latch after being recognized by the 8292. Usually low (except after hardware Reset), it will be pulsed high when $\overline{IFCR}$ is recognized by the 8292.
$\overline{ATNO}$	O	29	Attention Out — Controls the ATN control line of the bus through external logic for tcs and tca procedures. (ATN is a GPIB control line, as defined by IEEE Std. 488-1978.)
V_{CC}	P.S.	5, 26, 40	+5V supply input. ±10%.
COUNT	I	39	Count Input — When enabled by the proper command the internal counter will count external events through this pin. High to low transition will increment the internal counter by one. The pin is sampled once per three internal instruction cycles (7.5μsec sample period when using 6 MHz XTAL). It can be used for byte counting when connected to NDAC, or for block counting when connected to the EOI.
$\overline{REN}$	O	38	The Remote Enable bus signal selects remote or local control of the device on the bus. A GPIB bus management line, as defined by IEEE Std. 488-1978.
$\overline{DAV}$	I/O	37	DAV Handshake Line — Used during parallel poll to force the 8291 to accept the parallel poll status bits. It is also used during the tcs procedure.
$\overline{IBFI}$	O	36	Input Buffer Not Full — Used to interrupt the central processor while the input buffer of the 8292 is empty. This feature is enabled and disabled by the interrupt mask register.
OBFI	O	35	Output Buffer Full — Used as an interrupt to the central processor while the output buffer of the 8292 is full. The feature can be enabled and disabled by the interrupt mask register.
$\overline{EOI2}$	I/O	34	End Or Identify — One of the GPIB management lines, as defined by IEEE Std. 488-1978. Used with ATN as Identify Message during parallel poll.
SPI	O	33	Special Interrupt — Used as an interrupt on events not initiated by the central processor.
TCI	O	32	Task Complete Interrupt — Interrupt to the control processor used to indicate that the task requested was completed by the 8292 and the information requested is ready in the data bus buffer.
CIC	O	31	Controller In Charge — Controls the S/R input of the SRQ bus transceiver. It can also be used to indicate that the 8292 is in charge of the GPIB bus.

8293
GPIB TRANSCEIVER

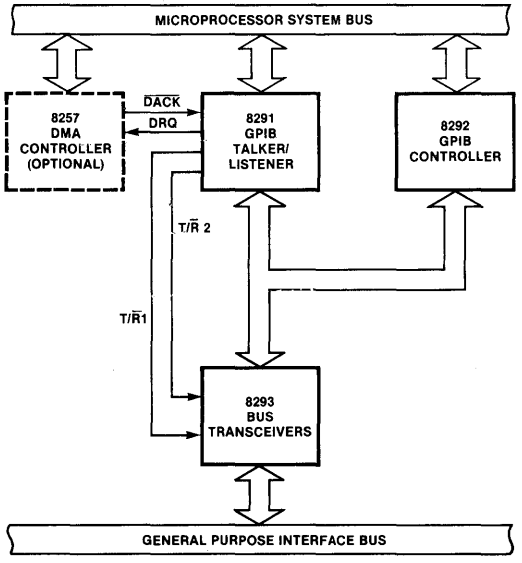
- Nine Open-collector or Three-state Line Drivers
- 48 mA Sink Current Capability on Each Line Driver
- Nine Schmitt-type Line Receivers
- High Capacitance Load Drive Capability
- Single 5V Power Supply
- 28-Pin Package
- Low Power HMOS Design
- On-chip Decoder for Mode Configuration
- Power Up/Power Down Protection to Prevent Disrupting the IEEE Bus
- Connects with the 8291 and 8292 to Form an IEEE Standard 488 Interface Talker/Listener/Controller with no Additional Components
- Only Two 8293's Required per GPIB Interface
- On-Chip IEEE-488 Bus Terminations

The Intel® 8293 GPIB Transceiver is a high current, non-inverting buffer chip designed to interface the 8291 GPIB Talker/Listener or the 8292 GPIB Controller with the 8291 to the IEEE Standard 488-1978 Instrumentation Interface Bus. Each GPIB interface would contain two 8293 Bus Transceivers. In addition, the 8293 can also be used as a general purpose bus driver.

PIN CONFIGURATION



8291, 8292, 8293 SYSTEM DIAGRAM



PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
BUS1-BUS9	I/O	12, 13, 15-19, 21, 22	These are the IEEE-488 bus interface driver/receivers. Using the mode select pins, they can be configured differently to allow direct connections between the 8291 GPIB Talker/Listener and the 8292 GPIB Controller.
DATA1-DATA10	I/O	5-11, 23-25	These are the pins to be connected to the 8291 and 8292 to interface with the GPIB bus. Their use is programmed by the two mode select pins, OPTA and OPTB. All these pins are TTL compatible.
T/$\overline{R}$1	I	1	Transmit receive 1; this pin controls the direction for NDAC, NRFD, DAV, and DIO1-DIO8. Input is TTL compatible.
T/$\overline{R}$2	I	2	Transmit receive 2; this pin controls the direction for EOI. Input is TTL compatible.
$\overline{EOI}$	I/O	3	End or Identify; this pin indicates the end of a multiple byte transfer or, in conjunction with ATN, addresses the device during a polling sequence. It connects to the 8291 and is switched between transmit and receive by T/$\overline{R}$2. This pin is TTL compatible.
$\overline{ATN}$	O	4	Attention; this pin is used by the 8291 to monitor the GPIB ATN control line. It specifies how data on the DIO lines is to be interpreted. This output is TTL compatible.
OPTA OPTB	I I	27 26	These two pins are to control the function of the 8293. A truth table of how this programs the various modes is in Table 1.
V_{CC}	P.S.	28	Positive power supply (5V ± 10%).
GND	P.S.	14, 20	Circuit ground potential.

Table 1. 8293 Mode Selection Pin Mapping

Pin Name	Pin No.	IEEE Implementation Name			
		Mode 0	Mode 1	Mode 2	Mode 3
OPTA	27	0	1	0	1
OPTB	26	0	0	1	1
DATA1	5	$\overline{IFC}$	$\overline{DIO8}$	$\overline{IFC}$	$\overline{DIO8}$
BUS1	12	IFC*	DIO8*	IFC*	DIO8*
DATA2	6	$\overline{REN}$	$\overline{DIO7}$	$\overline{REN}$	$\overline{DIO7}$
BUS2	13	REN*	DIO7*	REN*	DIO7*
DATA3	7	NC	$\overline{DIO6}$	$\overline{EOI2}$	$\overline{DIO6}$
BUS3	15	EOI*	DIO6*	EOI*	DIO6*
DATA4	8	$\overline{SRQ}$	$\overline{DIO5}$	$\overline{SRQ}$	$\overline{DIO5}$
BUS4	16	SRQ*	DIO5*	SRQ*	DIO5*
DATA5	9	$\overline{NRFD}$	$\overline{DIO4}$	$\overline{NRFD}$	$\overline{DIO4}$
BUS5	17	NRFD*	DIO4*	NRFD*	DIO4*
DATA6	10	$\overline{NDAC}$	$\overline{DIO3}$	$\overline{NDAC}$	$\overline{DIO3}$
BUS6	18	NDAC*	DIO3*	NDAC*	DIO3*
DATA7	11	T/$\overline{R}$IO1	NC	$\overline{ATNI}$	$\overline{ATNO}$
DATA8	23	T/$\overline{R}$IO2	$\overline{DIO2}$	$\overline{ATNO}$	$\overline{DIO2}$
BUS7	19	ATN*	DIO2*	ATN*	DIO2*
DATA9	24	$\overline{GIO1}$	$\overline{DAV}$	$\overline{CIC}$	$\overline{DAV}$
BUS8	21	GIO1*	DAV*	CLTH	DAV*
DATA10	25	$\overline{GIO2}$	$\overline{DIO1}$	$\overline{IFCL}$	$\overline{DIO1}$
BUS9	22	GIO2*	DIO1*	SYC	DIO1*
T/$\overline{R}$1	1	T/$\overline{R}$1	T/$\overline{R}$1	T/$\overline{R}$1	T/$\overline{R}$1
T/$\overline{R}$2	2	T/$\overline{R}$2	NC	T/$\overline{R}$2	$\overline{IFCL}$
$\overline{EOI}$	3	$\overline{EOI}$	$\overline{EOI}$	$\overline{EOI}$	$\overline{EOI}$
$\overline{ATN}$	4	$\overline{ATN}$	$\overline{ATN}$	$\overline{ATN}$	$\overline{ATN}$

***Note:** These pins are the IEEE-488 bus non-inverting driver/receivers. They include all the bus terminations required by the Standard and may be connected directly to the GPIB bus connector.

GENERAL DESCRIPTION

The 8293 is a bidirectional transceiver. It was designed to interface the Intel 8291 GPIB Talker/Listener and the Intel® 8292 GPIB Controller to the IEEE Standard 488-1978. Instrumentation Bus (also referred to as the GPIB Bus). The Intel GPIB Bus Transceiver meets or exceeds all of the electrical specifications defined in the IEEE Standard 488-1978, Section 3.3-3.5, including the required bus termination specifications.

The 8293 can be hardware programmed to one of four modes of operation. These modes allow the 8293 to be configured to support both a Talker/Listener/Controller environment and Talker/Listener environment. In addition, the 8293 can be used as a general purpose three-state (push-pull) or open-collector bus transceiver with nine receiver/drivers. Two modes are used to support a Talker/Listener environment (see Figure 1), and to support a Talker/Listener/Controller environment (see Figure 2). Mode 1 is the general purpose mode.

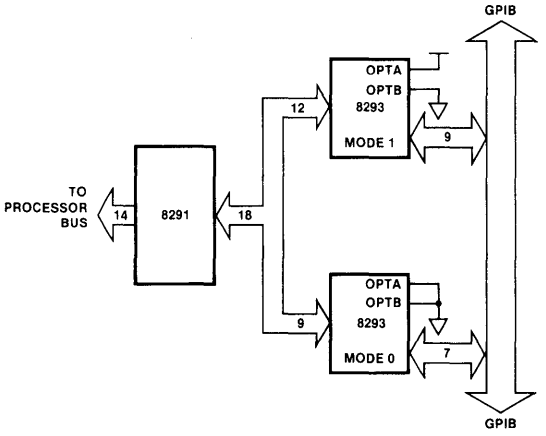


Figure 1. Talker/Listener Configuration

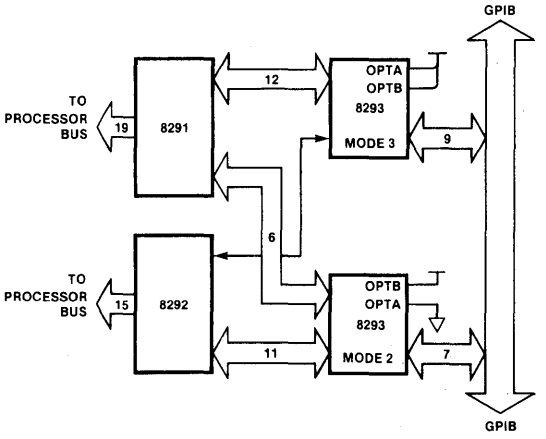


Figure 2. Talker/Listener/Controller Configuration

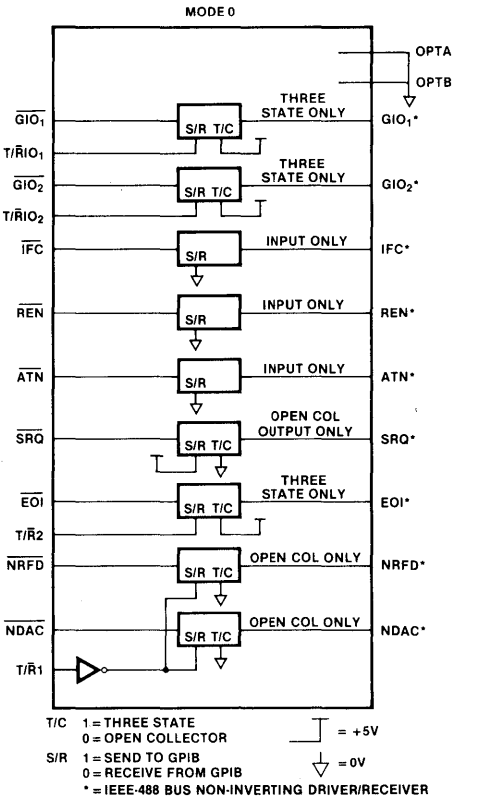


Figure 3. Talker/Listener Control Configuration

MODE 0 PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
T/R̄1	I	1	Transmit receive 1; direction control for NDAC and NRFD. If T/R̄1 is high, then NDAC* and NRFD* are receiving. Input is TTL compatible.
$\overline{NDAC}$	I/O	10	Not Data Accepted; processor GPIB bus handshake control line; used to indicate the condition of acceptance of data by device(s). It is TTL compatible.
NDAC*	I/O	18	Not Data Accepted; IEEE GPIB bus handshake control line. When an input, it is a TTL compatible Schmitt-trigger. When an output, it is an open-collector driver with 48 mA sinking capability.
$\overline{NRFD}$	I/O	9	Not Ready For Data; processor GPIB handshake control line; used to indicate the condition of readiness of device(s) to accept data. This pin is TTL compatible.

Symbol	I/O	Pin No.	Function
NRFD*	I/O	17	Not Ready For Data; IEEE GPIB bus handshake control line. When an input, it is a TTL compatible Schmitt-trigger. When an output, it is an open-collector driver with a 48 mA current sinking capability.
T/$\overline{R}$2	I	2	Transmit receive 2; direction control for EOI. If T/$\overline{R}$2 is high, EOI* is sending. Input is TTL compatible.
$\overline{EOI}$	I/O	3	End or Identify; processor GPIB bus control line; is used by a talker by indicate the end of a multiple byte transfer. This pin is TTL compatible.
EOI*	I/O	15	End or Identify; IEEE GPIB bus control line; is used by a talker to indicate the end of a multiple byte transfer. This pin is a three-state (push-pull) driver capable of sinking 48 mA and a TTL compatible receiver with hysteresis.
$\overline{SRQ}$	I	8	Service Request; processor GPIB bus control line; used by a device to indicate the need for service and to request an interruption of the current sequence of events on the GPIB. It is a TTL compatible input.
SRQ*	O	16	Service Request; IEEE GPIB bus control line; it is an open collector driver capable of sinking 48 mA.
$\overline{REN}$	O	6	Remote Enable; processor GPIB bus control line; used by a controller (in conjunction with other messages) to select between two alternate sources of device programming data (remote or local control). This output is TTL compatible.
REN*	I	13	Remote Enable; IEEE GPIB bus control line. This input is a TTL compatible Schmitt-trigger.
$\overline{ATN}$	O	4	Attention; processor GPIB bus control line; used by the 8291 to determine how data on the DIO signal lines are to be interpreted. This is a TTL compatible output.
ATN*	I	19	Attention; IEEE GPIB bus control line; this input is a TTL compatible Schmitt-trigger.

Symbol	I/O	Pin No.	Function
$\overline{IFC}$	O	5	Interface Clear; processor GPIB bus control line; used by a controller to place the interface system into a known quiescent state. It is a TTL compatible output.
IFC*	I	12	Interface Clear; IEEE GPIB bus control line. This input is a TTL compatible Schmitt-trigger.
T/$\overline{R}$IO1 T/$\overline{R}$IO2	I I	11 23	Transmit receive General IO; direction control for the two spare transceivers. Input is TTL compatible.
$\overline{GIO1}$ $\overline{GIO2}$	I/O I/O	24 25	General IO; this is the TTL side of the two spare transceivers. These pins are TTL compatible.
GIO1* GIO2*	I/O I/O	21 22	General IO; these are spare three-state (push-pull) drivers/ Schmitt-trigger receivers. The drivers can sink 48 mA.

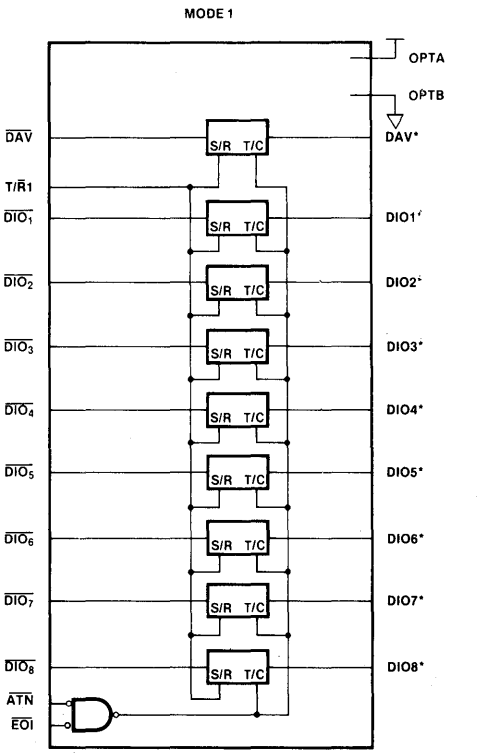


Figure 4. Talker/Listener Data Configuration

MODE 1 PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
T/$\overline{R}$1	I	1	Transmit receive 1; controls the direction for DAV and the DIO lines. If T/$\overline{R}$1 is high, then all these lines are sending information to the IEEE GPIB lines. This input is TTL compatible.
$\overline{EOI}$	I	3	End of Sequence and Attention; processor GPIB control lines. These two control signals are ANDed together to determine whether all the transceivers in the 8293 are three-state (push-pull) or open-collector. When both signals are low (true), then the controller is performing a parallel poll and the transceivers are all open-collector. These inputs are TTL compatible.
$\overline{ATN}$	I	4	
$\overline{DAV}$	I/O	24	Data Valid; processor GPIB bus handshake control line; used to indicate the condition (availability and validity) of information on the DIO signals. It is TTL compatible.
DAV*	I/O	21	Data Valid; IEEE GPIB bus handshake control line. When an input, it is a TTL compatible Schmitt-trigger. When DAV* is an output, it can sink 48 mA.
$\overline{DIO1}$-$\overline{DIO8}$	I/O	25, 23, 10, 9, 8, 7, 6, 5	Data Input/Output; processor GPIB bus data lines; used to carry message and data bytes in a bit-parallel byte-serial form controlled by the three handshake signals. These lines are TTL compatible.
DIO1* DIO8*	I/O	22, 19, 18, 17, 16, 15, 13, 12	Data Input/Output; IEEE GPIB bus data lines. They are TTL compatible Schmitt-triggers when used for input and can sink 48 mA when used for output. See $\overline{ATN}$ and $\overline{EOI}$ description for output mode.

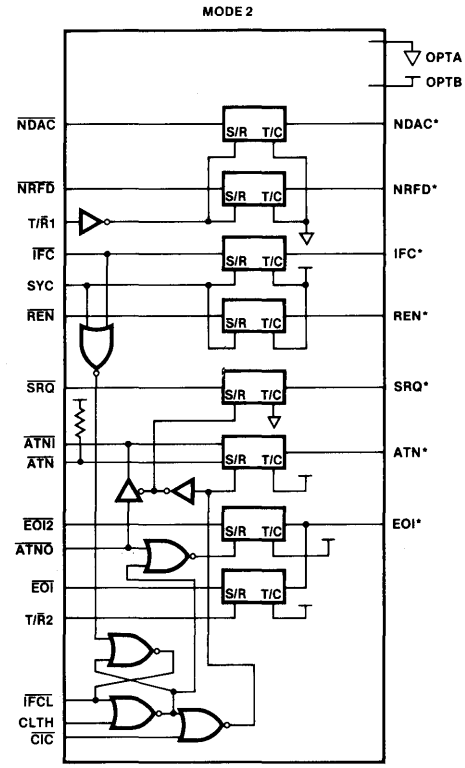


Figure 5. Talker/Listener/Controller Control Configuration

MODE 2 PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
T/$\overline{R}$1	I	1	Transmit receive 1; direction control for NDAC and NRFD. If T/$\overline{R}$1 is high, then NDAC and NRFD are receiving. Input is TTL compatible.
$\overline{NDAC}$	I/O	10	Not Data Accepted; processor GPIB bus handshake control line; used to indicate the condition of acceptance of data by device(s). This pin is TTL compatible.
NDAC*	I/O	18	Not Data Accepted; IEEE GPIB bus handshake control line. It is a TTL compatible Schmitt-trigger when used for input and an open-collector driver with a 48 mA current sink capability when used for output.

Symbol	I/O	Pin No.	Function
$\overline{NRFD}$	I/O	9	Not Ready For Data; processor GPIB bus handshake control line; used to indicate the condition of readiness of device(s) to accept data. This pin is TTL compatible.
NRFD*	I/O	17	Not Ready For Data; IEEE GPIB bus handshake control line. It is a TTL compatible Schmitt-trigger when used for input and an open-collector driver with a 48 mA current sink capability when used for output.
SYC	I	22	System Controller; used to monitor the system controller switch and control the direction for IFC and REN. This pin is a TTL compatible input.
$\overline{REN}$	I/O	6	Remote Enable; processor GPIB control line; used by the active controller (in conjunction with other messages) to select between two alternate sources of device programming data (remote or local control). This pin is TTL compatible.
REN*	I/O	13	Remote Enable; IEEE GPIB bus control line. When used as an input, this is a TTL compatible Schmitt-trigger. When an output, it is a three-state driver with a 48 mA current sinking capability.
$\overline{IFC}$	I/O	5	Interface Clear; processor GPIB bus control line; used by the active controller to place the interface system into a known quiescent state. This pin is TTL compatible.
IFC*	I/O	12	Interface Clear; IEEE GPIB bus control line. This is a TTL compatible Schmitt-trigger when used for input and a three-state driver capable of sinking 48 mA current when used for output.
$\overline{CIC}$	I	24	Controller in Charge; used to control the direction of the SRQ and to indicate that the 8292 is in charge of the bus. $\overline{CIC}$ is a TTL compatible input.
CLTH	I	21	Clear Latch; used to clear the IFC Received latch after it has been recognized by the 8292. Normally low (except after a hardware reset), it will be pulsed low when IFC Received is recognized by the 8292. This input is TTL compatible.
$\overline{IFCL}$	O	25	IFC Received Latched; the 8292 monitors the IFC line when it is not the active controller through this pin.
$\overline{SRQ}$	I/O	8	Service Request; processor GPIB control line; indicates the need for attention and requests the active controller to interrupt the current sequence of events on the GPIB bus. This pin is TTL compatible.
SRQ*	I/O	16	Service Request; IEEE GPIB bus control line. When used as an input, this pin is a TTL compatible Schmitt-trigger. When used as an output, it is an open-collector driver with a 48 mA current sinking capability.
T/$\overline{R}$2	I	2	Transmit receive 2; controls the direction for EOI. This input is TTL compatible.
$\overline{ATNO}$	I	23	Attention Out; processor GPIB bus control line; used by the 8292 for ATN control of the IEEE bus during "take control synchronously" operations. A low on this input causes ATN to be asserted if $\overline{CIC}$ indicates that this 8292 is in charge. $\overline{ATNO}$ is a TTL compatible input.
$\overline{ATNI}$	O	11	Attention In; processor GPIB bus control line; used by the 8292 to monitor the ATN line. This output is TTL compatible.
$\overline{ATN}$	O	4	Attention; processor GPIB bus control live; used by the 8292 to monitor the ATN line. This output is TTL compatible.
ATN*	I/O	19	Attention; IEEE GPIB bus control line; used by a controller to specify how data on the DIO signal lines are to be interpreted and which devices must respond to data. When used as an output, this pin is a three-state driver capable of sinking 48 mA current. As an input, it is a TTL compatible Schmitt-trigger.
$\overline{EOI2}$	I/O	7	End or Identify 2; processor GPIB bus control line; used in conjunction with ATN by the active controller (the 8292) to execute a polling sequence. This pin is TTL compatible.

Symbol	I/O	Pin No.	Function
$\overline{EOI}$	I/O	3	End or Identify; processor GPIB bus control line; used by a talker to indicate the end of a multiple byte transfer sequence. This pin is TTL compatible.
EOI*	I/O	15	End or Identify; IEEE GPIB bus control line; used by a talker to indicate the end of a multiple byte transfer sequence or, by a controller in conjunction with ATN, to execute a polling sequence. When an output, this pin can sink 48 mA current. When an input, it is a TTL compatible Schmitt-trigger.

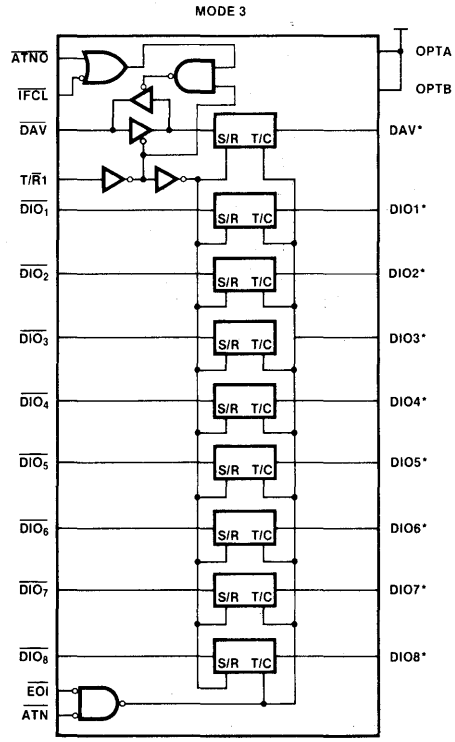


Figure 6. Talker/Listener/Controller Data Configuration

MODE 3 PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
T/$\overline{R}$1	I	1	Transmit receive 1; controls the direction for DAV and the DIO lines. If T/$\overline{R}$1 is high, then all these lines are sending information to the IEEE GPIB lines. This input is TTL compatible.
$\overline{EOI}$	I	3	End of Sequence and Attention; processor GPIB control lines. These two control lines are ANDed together to determine whether all the transceivers in the 8293 are push-pull or open-collector. When both signals are low (true), then the controller is performing a parallel poll and the transceivers are all open-collector. These inputs are TTL compatible.
$\overline{ATN}$	I	4	
$\overline{ATNO}$	I	23	Attention Out; processor GPIB control line; used by the 8292 during "take control synchronously" operations. This pin is TTL compatible.
$\overline{IFCL}$	I	2	Interface Clean Latched; used to make DAV received after the system controller asserts IFC. This input is TTL compatible.
$\overline{DAV}$	I/O	24	Data Valid; processor GPIB handshake control line; used to indicate the condition (availability and validity) of information on the DIO signals. This pin is TTL compatible.
DAV*	I/O	21	Data Valid; IEEE GPIB handshake control line. When an input, this pin is a TTL compatible Schmitt-trigger. When DAV* is an output, it can sink 48 mA.
$\overline{DIO1}$-$\overline{DIO8}$	I/O	25, 23, 10, 9, 8, 7, 6, 5	Data Input/Output; processor GPIB bus data lines; used to carry message and data bytes in a bit-parallel byte-serial form controlled by the three handshake signals. These lines are TTL compatible.
DIO1*-DIO8*	I/O	22, 19, 18, 17, 16, 15, 13, 12	Data Input/Output; IEEE GPIB bus data lines. They are TTL compatible Schmitt-triggers when used for input and can sink 48 mA when used for output.

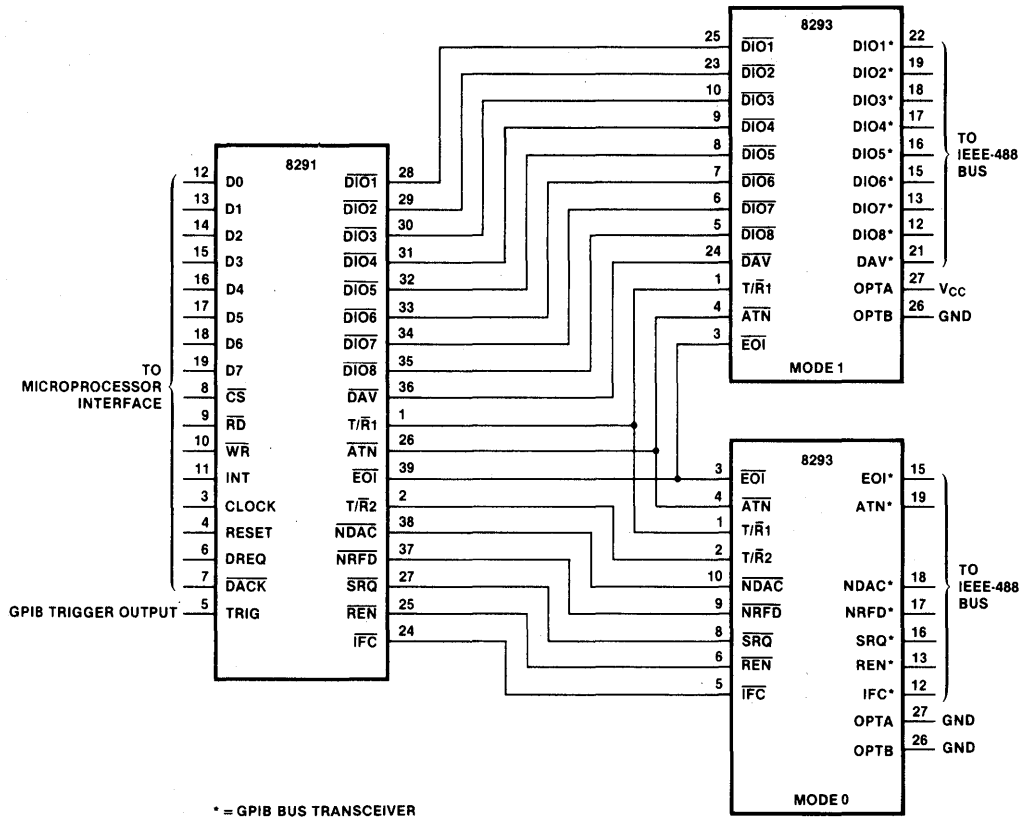


Figure 7. 8291 and 8293 System Configuration

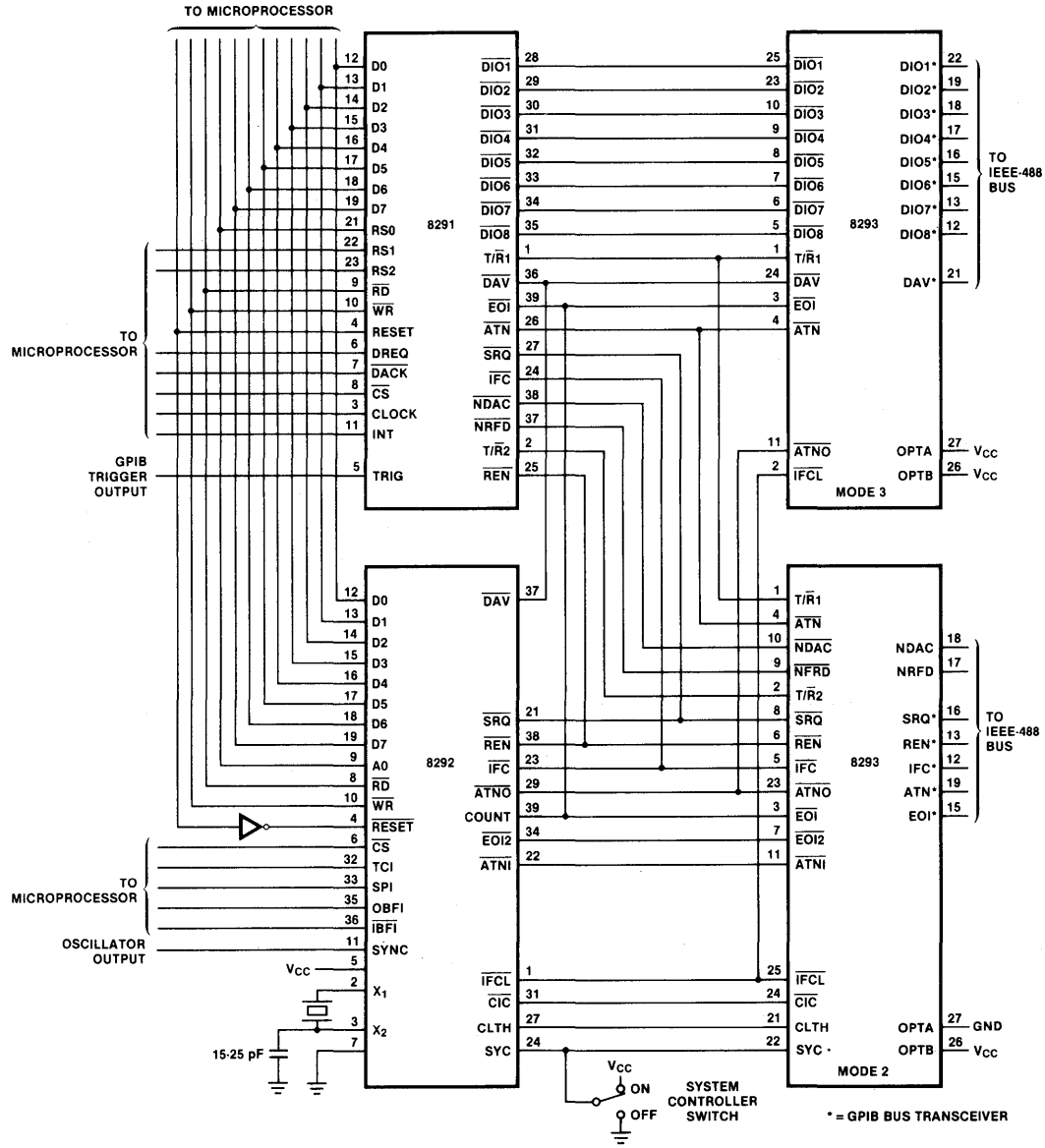


Figure 8. 8291, 8292, and 8293 System Configuration

Absolute Maximum Ratings*

Ambient Temperature Under Bias.....0°C to 70°C
Storage Temperature.....−65°C to +150°C
Voltage on any Pin with
Respect to Ground.....−1.0V to +7V
Power Dissipation.....1 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. and Operating Characteristics

$T_A = 0°C$ to 70°C; $V_{CC} = 5.0V \pm 10\%$; GND = 0V

SYMBOL	PARAMETER	LIMITS			UNIT	TEST CONDITIONS
		MIN.	TYP.	MAX.		
V_{IL1}	Input Low Voltage (GPIB Bus Pins)			0.8	V	
V_{IL2}	Input Low Voltage (Option Pins)	−0.1		0.1	V	
V_{IL3}	Input Low Voltage (All Others)			0.8	V	
V_{IH1}	Input High Voltage (GPIB Bus Pins)	2.0			V	
V_{IH2}	Input High Voltage (Option Pins)	4.5		5.5	V	
V_{IH3}	Input High Voltage (All Others)	2.0			V	
V_{OL1}	Output Low Voltage (GPIB Bus Pins)			0.5	V	$I_{OL} = 48$ mA
V_{OL2}	Output Low Voltage (All Others)			0.5	V	$I_{OL} = 16$ mA
V_{OH1}	Output High Voltage (GPIB Bus Pins)	2.4			V	$I_{OH} = -5.2$ mA
V_{OH2}	Output High Voltage (All Others)	2.4			V	$I_{OH} = -400\ \mu A$
V_{IH4}	Receiver Input Hysteresis	400	600		mV	
V_{IT}	Receiver Input Threshold High to Low	0.8	1.0		V	
	Receiver Input Threshold Low to High		1.6	2.0		
I_{LI1}	Low Input Load Current (GPIB Bus Pins)	−3.2		0.0	mA	$V_{IL} = 0.8V$
I_{LI2}	Low Input Load Current (All Others)			10	μA	$V_{IL} = 0.8V$
I_{PD}	Bus Power Down Leakage Current			10	μA	$V_{CC} = 0V$
I_{CC}	Power Supply Current			100	mA	

Capacitance

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance		5	10	pF	$V_{IN} = V_{CC}$
C_{OUT}	Output Capacitance		10	20	pF	$V_{OUT} = V_{CC}$

A.C. Characteristics

$T_A = 0°C$ to $70°C$; $V_{CC} = 5.0V \pm 10\%$; $GND = 0V$

SYMBOL	PARAMETER	TYP.*	MAX.	UNITS
t_{PLH1}	Driver Propagation Delay (Low to High)	20	35	ns
t_{PHL1}	Driver Propagation Delay (High to Low)	17	30	ns
t_{PLH2}	Receiver Propagation Delay (Low to High)	22	35	ns
t_{PHL2}	Receiver Propagation Delay (High to Low)	18	30	ns
t_{PHZ1}	Driver Enable Delay (High to 3-State)	20	35	ns
t_{PZH1}	Driver Enable Delay (3-State to High)	15	30	ns
t_{PLZ1}	Driver Enable Delay (Low to 3-State)	20	35	ns
t_{PZL1}	Driver Enable Delay (3-State to Low)	15	30	ns
t_{PHZ2}	Receiver Enable Delay (High to 3-State)	25	40	ns
t_{PZH2}	Receiver Enable Delay (3-State to High)	20	35	ns
t_{PLZ2}	Receiver Enable Delay (Low to 3-State)	25	40	ns
t_{PZL2}	Receiver Enable Delay (3-State to Low)	20	35	ns

*Typical @ $T_A = 25°C$.

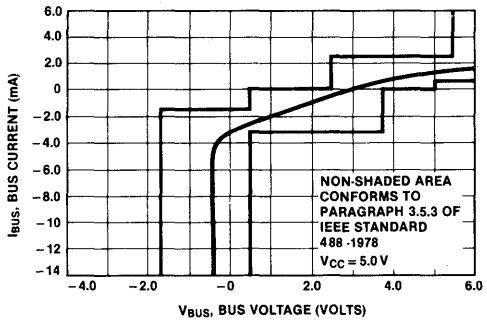


Figure 9. Typical Bus Load Line

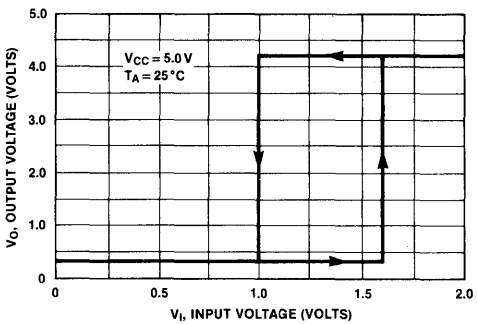


Figure 10. Typical Receiver Hysteresis Characteristics

OUTPUT LOADING TEST CIRCUITS

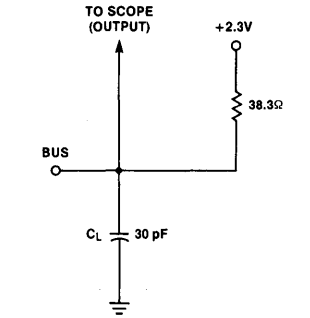


Figure 11. Data Input to Bus Output (Driver)

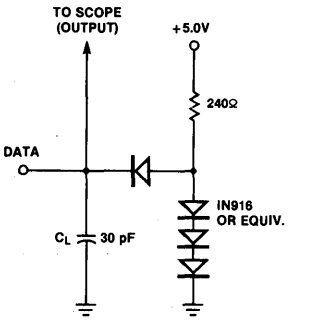


Figure 12. Bus Input to Data Output (Receiver)

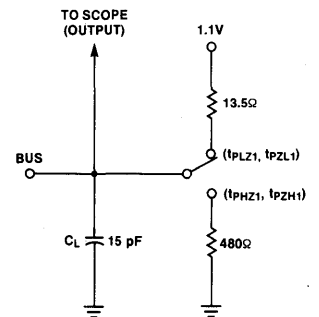


Figure 13. Send/Receive Input to Bus Output (Driver)

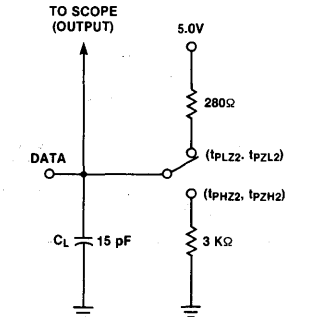
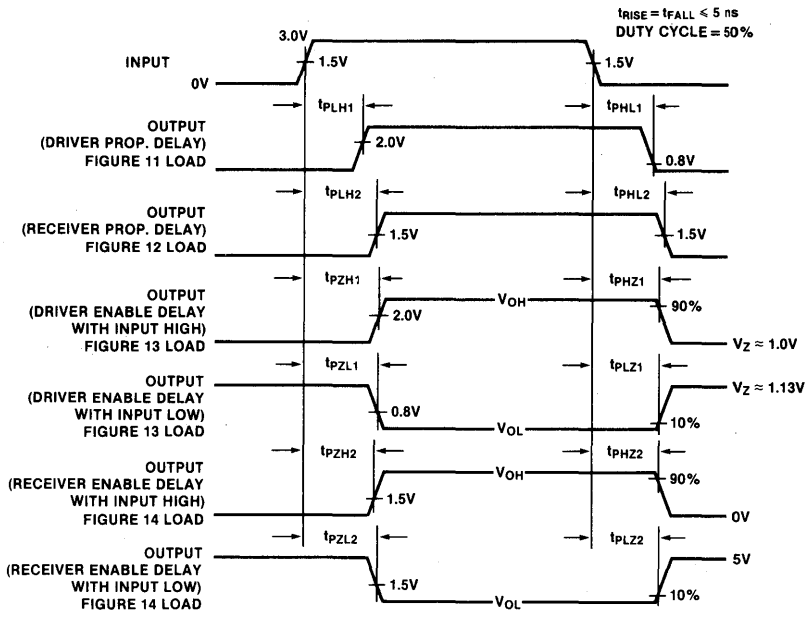


Figure 14. Send/Receive Input to Data Output (Receiver)

8293 WAVEFORMS



8294
DATA ENCRYPTION UNIT

- **Certified by National Bureau of Standards**
- **80 Byte/Sec Data Conversion Rate**
- **64-Bit Data Encryption Using 56-Bit Key**
- **DMA Interface**
- **3 Interrupt Outputs to Aid in Loading and Unloading Data**
- **7-Bit User Output Port**
- **Single 5V ± 10% Power Supply**
- **Peripheral to MCS-86™, MCS-85™, MCS-80™ and MCS-48™ Processors**
- **Implements Federal Information Processing Data Encryption Standard**
- **Encrypt and Decrypt Modes Available**

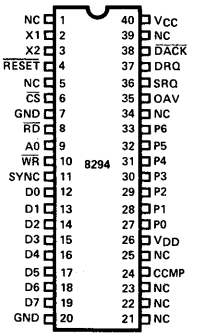
DESCRIPTION

The Intel® 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

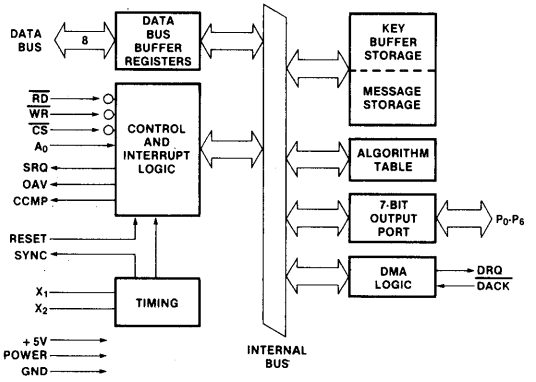
The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 80 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

PIN CONFIGURATION



BLOCK DIAGRAM



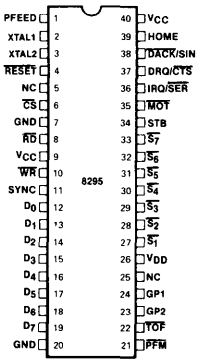
8295
DOT MATRIX PRINTER CONTROLLER

- **Interfaces Dot Matrix Printers to MCS-48™, MCS-80/85™, MCS-86™ Systems**
- **40 Character Buffer On Chip**
- **Serial or Parallel Communication with Host**
- **DMA Transfer Capability**
- **Programmable Character Density (10 or 12 Chararcters/Inch)**
- **Programmable Print Intensity**
- **Single or Double Width Printing**
- **Programmable Multiple Line Feeds**
- **3 Tabulations**
- **2 General Purpose Outputs**

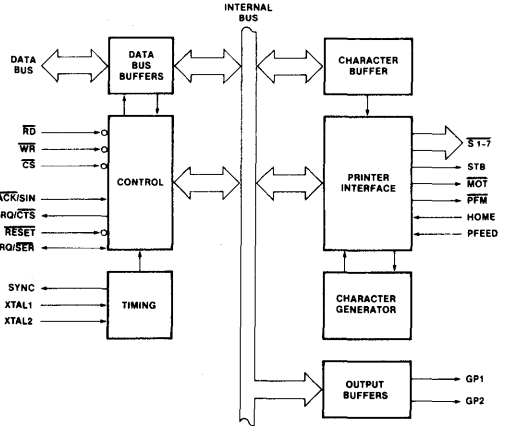
The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. In parallel mode, data transfers are based on polling, interrupts, or DMA. Furthermore, it provides internal buffering of up to 40 characters and contains a 7 x 7 matrix character generator accommodating 64 ASCII characters.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN DESCRIPTION

Name	I/O	Pin #	Description
PFEED	I	1	Paper feed input switch.
XTAL1	I	2	Inputs for a crystal to set internal oscillator frequency. For proper operation use 6 MHz crystal.
XTAL2		3	
$\overline{RESET}$	I	4	Reset input, active low. After reset the 8295 will be set for 12 characters/inch single width printing, solenoid strobe at 320 msec.
NC	—	5	No connection or tied high.
$\overline{CS}$	I	6	Chip select input used to enable the $\overline{RD}$ and $\overline{WR}$ inputs except during DMA.
GND	—	7	This pin must be tied to ground.
$\overline{RD}$	I	8	Read input which enables the master CPU to read data and status. In the serial mode this pin must be tied to V_{CC}.
V_{CC}	—	9	+5 volt power input: +5V ± 10%.
$\overline{WR}$	I	10	Write input which enables the master CPU to write data and commands to the 8295. In the serial mode this pin must be tied to V_{SS}.
SYNC	O	11	2.5 μs clock output. Can be used as a strobe for external circuitry.
D_0	I/O	12	Three-state bidirectional data bus buffer lines used to interface the 8295 to the host processor in the parallel mode. In the serial mode $D_0 - D_2$ sets up the baud rate.
D_1		13	
D_2		14	
D_3		15	
D_4		16	
D_5		17	
D_6		18	
D_7		19	
GND	—	20	This pin must be tied to ground.
V_{CC}	—	40	+5 volt power input: +5V ± 10%.

Name	I/O	Pin #	Description
HOME	I	39	Home input switch, used by the 8295 to detect that the print head is in the home position.
$\overline{DACK}$/SIN	I	38	In the parallel mode used as DMA acknowledgement; in the serial mode, used as input for data.
DRQ/$\overline{CTS}$	O	37	In the parallel mode used as DMA request output pin to indicate to the 8257 that a DMA transfer is requested; in the serial mode used as clear-to-send signal.
IRQ/$\overline{SER}$	O	36	In parallel mode it is an interrupt request input to the master CPU; in serial mode it should be strapped to V_{SS}.
$\overline{MOT}$	O	35	Main motor drive, active low.
STB	O	34	Solenoid strobe output. Used to determine duration of solenoids activation.
$\overline{S_7}$	O	33	Solenoid drive outputs; active low.
$\overline{S_6}$		32	
$\overline{S_5}$		31	
$\overline{S_4}$		30	
$\overline{S_3}$		29	
$\overline{S_2}$		28	
$\overline{S_1}$		27	
V_{DD}	—	26	+5V power input (+5V ± 10%). Low power standby pin.
NC	—	25	No connection.
GP1	O	24	General purpose output pins.
GP2	O	23	
$\overline{TOF}$	I	22	Top of form input, used to sense top of form signal for type T printer.
$\overline{PFM}$	O	21	Paper feed motor drive, active low.

FUNCTIONAL DESCRIPTION

The 8295 interfaces microcomputers to the LRC 7040 Series dot matrix impact printers, and to other similar printers. It provides internal buffering of up to 40 characters. Printing begins automatically when the buffer is full or when a carriage return character is received. It provides a modified 7x7 matrix character generator. The character set includes 64 ASCII characters.

Communication between the 8295 and the host processor can be implemented in either a serial or parallel mode. The parallel mode allows for character transfers into the buffer via DMA cycles. The serial mode features selectable data rates from 110 to 4800 baud.

The 8295 also offers two general purpose output pins which can be set or cleared by the host processor. They can be used with various printers to implement such functions as ribbon color selection, enabling form release solenoid, and reverse document feed.

COMMAND SUMMARY

Hex Code	Description
00	Set GP1. This command brings the GP1 pin to a logic high state. After power on it is automatically set high.
01	Set GP2. Same as the above but for GP2.
02	Clear GP1. Sets GP1 pin to logic low state, inverse of command 00.
03	Clear GP2. Same as above but for GP2. Inverse command 01.
04	Software Reset. This is a pacify command. This command is not effective immediately after commands requiring a parameter, as the Reset command will be interpreted as a parameter.
05	Print 10 characters/in. density.
06	Print 12 characters/in. density.
07	Print double width characters. This command prints characters at twice the normal width, that is, at either 17 or 20 characters per line.
08	Enable DMA mode; must be followed by two bytes specifying the number of data characters to be fetched. Least significant byte accepted first.
09	Tab character.
0A	Line feed.
0B	Multiple Line Feed; must be followed by a byte specifying the number of line feeds.
0C	Top of Form. Enables the line feed output until the Top of Form input is activated.
0D	Carriage Return. Signifies end of a line and enables the printer to start printing.
0E	Set Tab #1, followed by tab position byte.
0F	Set Tab #2, followed by tab position byte. Should be greater than Tab #1.
10	Set Tab #3, followed by tab position byte. Should be greater than Tab #2.
11	Print Head Home on Right. On some printers the print head home position is on the right. This command would enable normal left to right printing with such printers.
12	Set Strobe Width; must be followed by strobe width selection byte. This command adjusts the duration of the strobe activation.

PROGRAMMABLE PRINTING OPTIONS

CHARACTER DENSITY

The character density is programmable at 10 or 12 characters/inch (32 or 40 characters/line). The 8295 is automatically set to 12 characters/inch at power-up. Invoking the Print Double-Width command halves the character density (5 or 6 characters/inch). The 10 char/in or 12 char/in command must be re-issued to cancel the Double-Width mode. Different character density modes may not be mixed within a single line of printing.

PRINT INTENSITY

The intensity of the printed characters is determined by the amount of time during which the solenoid is on. This on-time is programmable via the Set Strobe-Width command. A byte following this command sets the solenoid on-time according to Table 1. Note that only the three least significant bits of this byte are important.

D7—D3	D2	D1	D0	Solenoid On (microsec)
x	0	0	0	200
x	0	0	1	240
x	0	1	0	280
x	0	1	1	320
x	1	0	0	360
x	1	0	1	400
x	1	1	0	440
x	1	1	1	480

Table 1.

TABULATIONS

Up to three tabulation positions may be specified with the 8295. The column position of each tabulation is selected by issuing the Set Tab commands, each fol-

lowed by a byte specifying the column. The tab positions will then remain valid until new Set Tab commands are issued.

Sending a tab character (09H) will automatically fill the character buffer with blanks up to the next tab position. The character sent immediately after the tab character will thus be stored and printed at that position.

CPU TO 8295 INTERFACE

Communication between the CPU and the 8295 may take place in either a serial or parallel mode. However, the selection of modes is inherent in the system hardware; it is not software programmable. Thus, the two modes cannot be mixed in a single 8295 application.

PARALLEL INTERFACE

Two internal registers on the 8295 are addressable by the CPU: one for input, one for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Register
1	0	0	Input Data Register
0	1	0	Output Status Register

Input Data Register—Data written to this register is interpreted in one of two ways, depending on how the data is coded.

1. A command to be executed (0XH or 1XH).
2. A character to be stored in the character buffer for printing (2XH, 3XH, 4XH, or 5XH). See the character set, Table 2.

Output Status Register—8295 status is available in this register at all times.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	x	x	PA	DE	x	x	IBF	x

PA—Parameter Required; PA = 1 indicates that a command requiring a parameter has been received. After the necessary parameters have been received by the 8295, the PA flag is cleared.

DE—DMA Enabled; DE = 1 whenever the 8295 is in DMA mode. Upon completion of the required DMA transfers, the DE flag is cleared.

IBF—Input Buffer Full; IBF = 1 whenever data is written to the Input Data Register. No data should be written to the 8295 when IBF = 1.

A flow chart describing communication with the 8295 is shown in Figure 1.

The interrupt request output (IRQ, Pin 36) is available on the 8295 for interrupt driven systems. This output is asserted true whenever the 8295 is ready to receive data.

To improve bus efficiency and CPU overhead, data may be transferred from main memory to the 8295 via DMA cycles. Sending the Enable DMA command (08H) activates the DMA channel of the 8295. This command must be followed by two bytes specifying the length of the data string to be transferred (least significant byte first). The 8295 will then assert the required DMA requests to the 8257 DMA controller without further CPU intervention. Figure 2 shows a block diagram of the 8295 in DMA mode.

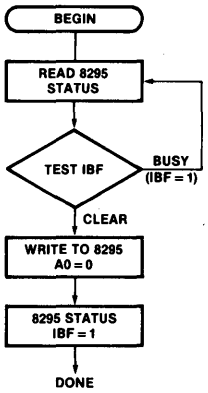


Figure 1. Host to 8295 Protocol Flowchart

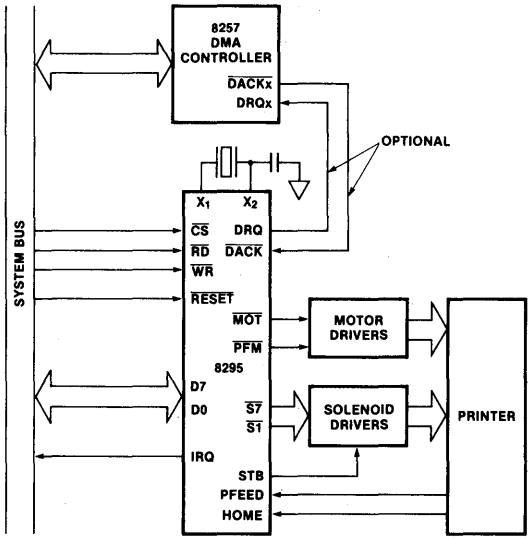


Figure 2. Parallel System Interface

Data transferred in the DMA mode may be either commands or characters or a mixture of both. The procedure is as follows:

1. Set up the 8257 DMA controller channel by sending a starting address and a block length.
2. Set up the 8295 by issuing the "Enable DMA" command (08H) followed by two bytes specifying the block length (least significant byte first).

The DMA enabled flag (DE) will be true until the assigned data transfer is completed. Upon completion of the transfer, the flag is cleared and the interrupt request (IRQ) signal is asserted. The 8295 then returns to the non-DMA mode of operation.

SERIAL INTERFACE

The 8295 may be hardware programmed to operate in a serial mode of communication. By connecting the IRQ/$\overline{SER}$ pin (pin 36) to logic zero, the serial mode is enabled immediately upon power-up. The serial Baud rate is also hardware programmable; by strapping pins 14, 13, and 12 according to Table 2, the rate is selected. $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ must be strapped as shown in Figure 3.

Pin 14	Pin 13	Pin 12	Baud Rate
0	0	0	110
0	0	1	150
0	1	0	300
0	1	1	600
1	0	0	1200
1	0	1	2400
1	1	0	4800
1	1	1	4800

Table 2.

The serial data format is shown in Figure 3. The CPU should wait for a clear to send signal ($\overline{CTS}$) from the 8295 before sending data.

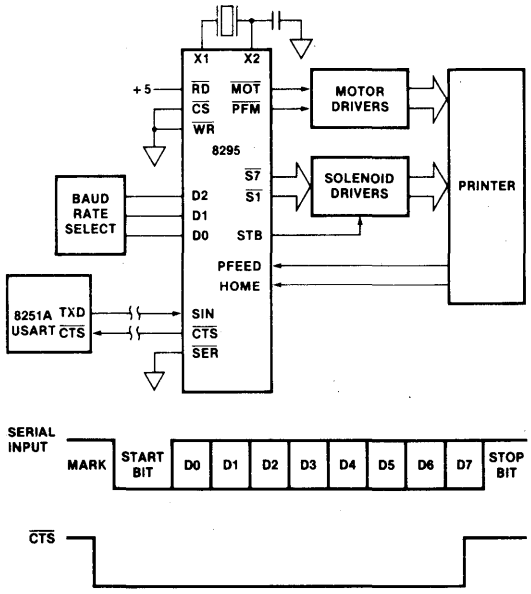


Figure 3. Serial Interface to UART (8251A)

8295 TO PRINTER INTERFACE

The strobe output signal of the 8295 determines the duration of the solenoid outputs, which hold the data to the printer. These solenoid outputs cannot drive the printer solenoids directly. They should be buffered through solenoid drivers as shown in Figure 4. Recommended solenoid and motor driver circuits may be found in the printer manufacturer's interface guide.

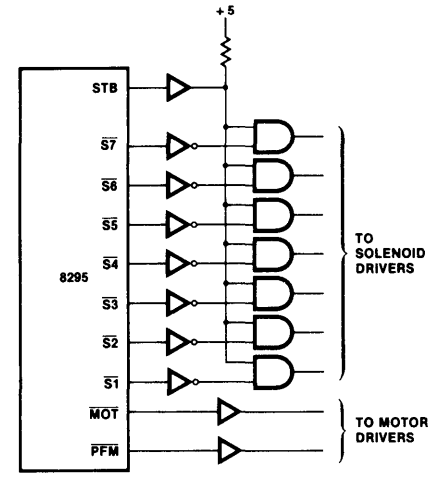


Figure 4. 8295 To Printer Solenoid Interface

OSCILLATOR AND TIMING CIRCUITS

The 8295's internal timing generation is controlled by a self-contained oscillator and timing circuit. A 6 MHz crystal is used to derive the basic oscillator frequency. The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 5. The recommended crystal connection is shown in Figure 6.

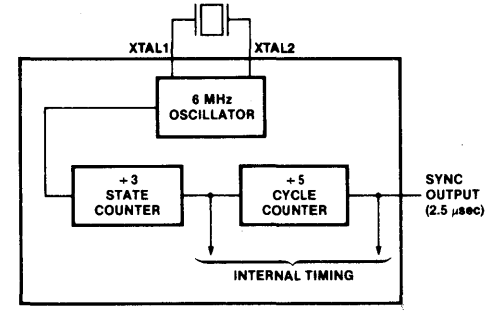


Figure 5. Oscillator Configuration

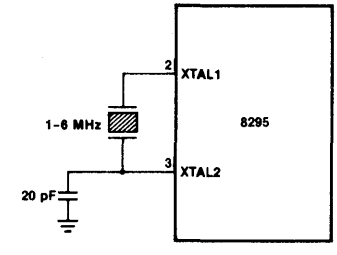


Figure 6. Recommended Crystal Connection

8295 CHARACTER SET

Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.
20	space	30	0	40	@	50	P
21	!	31	1	41	A	51	Q
22	"	32	2	42	B	52	R
23	#	33	3	43	C	53	S
24	$	34	4	44	D	54	T
25	%	35	5	45	E	55	U
26	&	36	6	46	F	56	V
27	'	37	7	47	G	57	W
28	(	38	8	48	H	58	X
29	)	39	9	49	I	59	Y
2A	*	3A	:	5A	J	5A	Z
2B	+	3B	;	4B	K	5B	[
2C	,	3C	<	4C	L	5C	\
2D	-	3D	=	4D	M	5D	]
2E	.	3E	>	4E	N	5E	↑
2F	/	3F	?	4F	O	5F	—

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65° to +150°C
Voltage on Any Pin With
Respect to Ground . . . . . 0.5V to +7V
Power Dissipation . . . . . 1.5 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to 70°C, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5		0.8	V	
V_{IL1}	Input Low Voltage (X_1, X_2, $\overline{RESET}$)	−0.5		0.6	V	
V_{IH}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2		V_{CC}	V	
V_{IH1}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (D_0-D_7)			0.45	V	$I_{OL} = 2.0$ mA
V_{OL1}	Output Low Voltage (All Other Outputs)			0.45	V	$I_{OL} = 1.6$ mA
V_{OH}	Output High Voltage (D_0-D_7)	2.4			V	$I_{OH} = -400\mu A$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			V	$I_{OH} = -50\mu A$
I_{IL}	Input Leakage Current ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0)			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)			±10	μA	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		5	15	mA	
$I_{DD} + I_{CC}$	Total Supply Current		60	125	mA	
I_{LI}	Low Input Load Current (Pins 24, 27-38)			0.5	mA	$V_{IL} = 0.8V$
I_{LI1}	Low Input Load Current ($\overline{RESET}$)			0.2	mA	$V_{IL} = 0.8V$

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}\downarrow$	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}\downarrow$ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{DF}	$\overline{RD}\uparrow$ to Data Float Delay		100	ns	
t_{CY}	Cycle Time	2.5	15	μs	

DBB WRITE

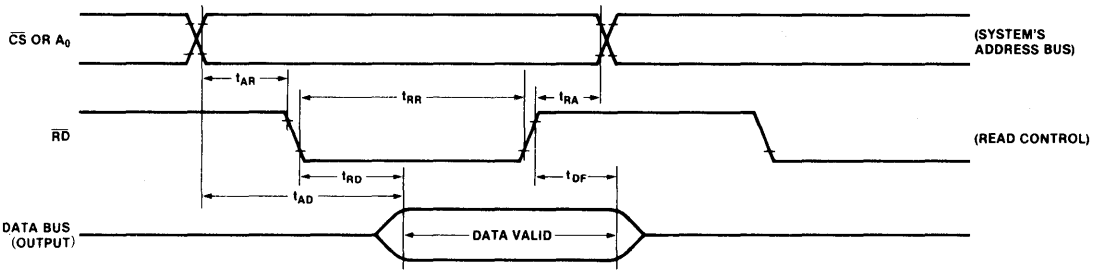
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}\downarrow$	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}\uparrow$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}\uparrow$	150		ns	
t_{WD}	Data Hold to $\overline{WR}\uparrow$	0		ns	

DMA AND INTERRUPT TIMING

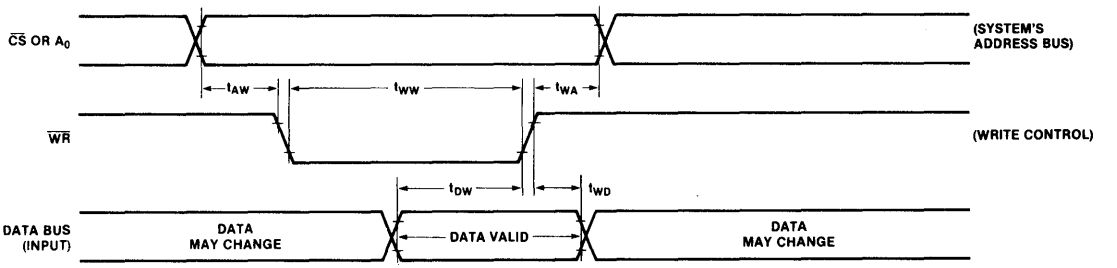
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{ACC}	$\overline{DACK}$ Setup to Control	0		ns	
t_{CAC}	$\overline{DACK}$ Hold After Control	0		ns	
t_{CRQ}	$\overline{WR}$ to DRQ Cleared		200	ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		225	ns	

WAVEFORMS

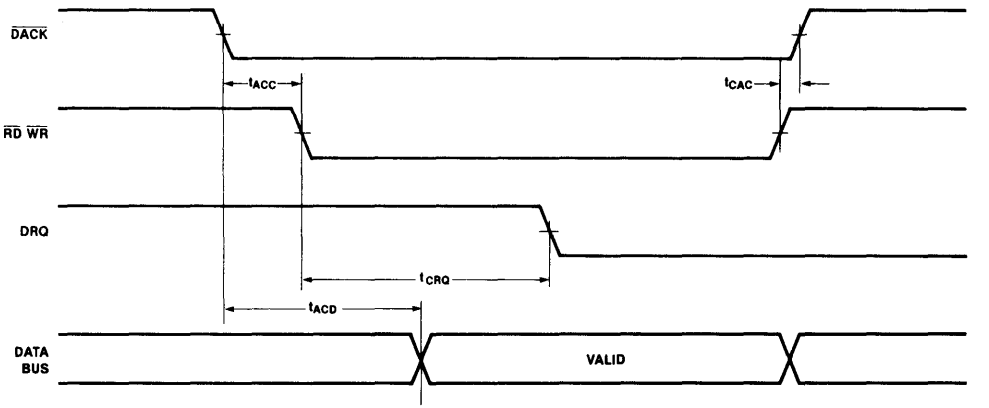
1. READ OPERATION — OUTPUT BUFFER REGISTER.



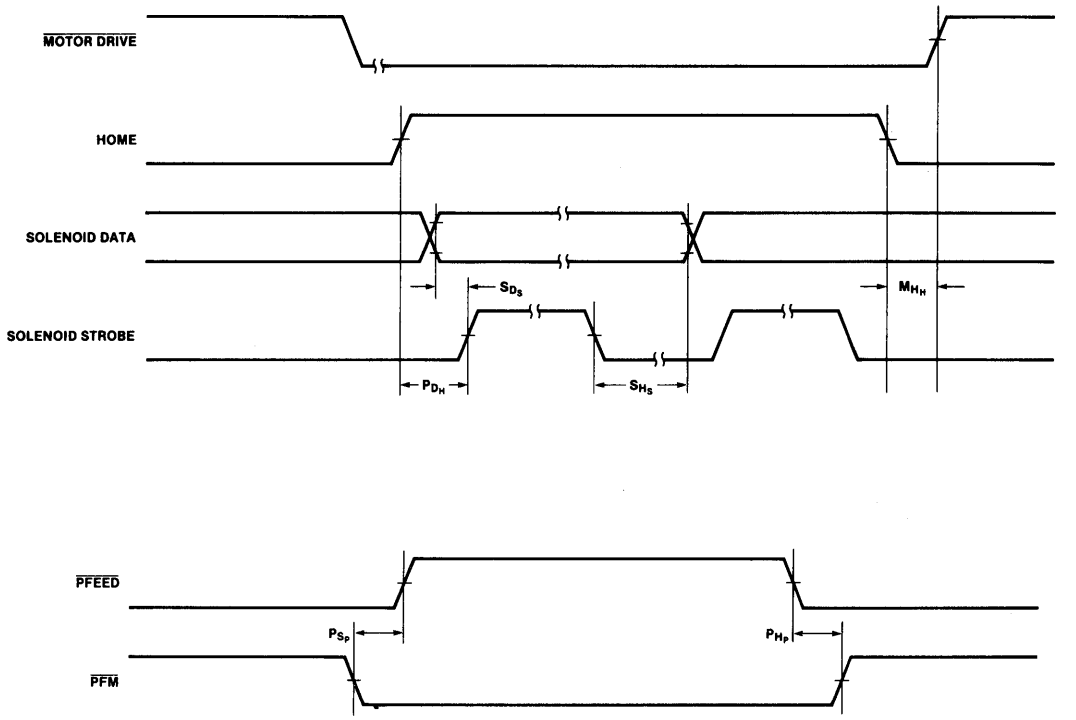
2. WRITE OPERATION — INPUT BUFFER REGISTER.



DMA AND INTERRUPT TIMING



PRINTER INTERFACE TIMING AND WAVEFORMS



Symbol	Parameter	Typical
P_{DH}	Print delay from home inactive	1.8 ms
S_{DS}	Solenoid data setup time before strobe active	25 μs
S_{HS}	Solenoid data hold after strobe inactive	>1 ms
M_{HA}	Motor hold time after home active	3.2 ms
P_{SP}	PFEED setup time after PFM active	58 ms
P_{HP}	PFM hold time after PFEED active	9.75 ms

2114A
1024 X 4 BIT STATIC RAM

	2114AL-1	2114AL-2	2114AL-3	2114AL-4	2114A-4	2114A-5
Max. Access Time (ns)	100	120	150	200	200	250
Max. Current (mA)	40	40	40	40	70	70

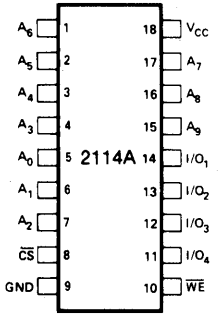
- **HMOS Technology**
- **Low Power, High Speed**
- **Identical Cycle and Access Times**
- **Single +5V Supply ±10%**
- **High Density 18 Pin Package**
- **Completely Static Memory - No Clock or Timing Strobe Required**
- **Directly TTL Compatible: All Inputs and Outputs**
- **Common Data Input and Output Using Three-State Outputs**
- **2114 Upgrade**

The Intel® 2114A is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using HMOS, a high performance MOS technology. It uses fully DC stable (static) circuitry throughout, in both the array and the decoding, therefore it requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

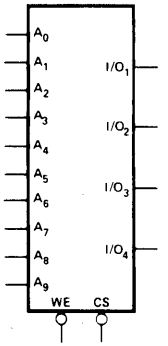
The 2114A is designed for memory applications where the high performance and high reliability of HMOS, low cost, large bit storage, and simple interfacing are important design objectives. The 2114A is placed in an 18-pin package for the highest possible density.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. A separate Chip Select ($\overline{CS}$) lead allows easy selection of an individual package when outputs are or-tied.

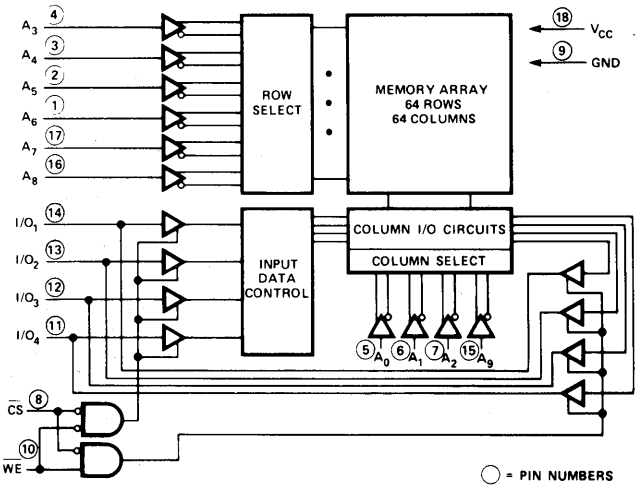
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0–A_9	ADDRESS INPUTS	V_{CC}	POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND	GROUND
$\overline{CS}$	CHIP SELECT		
I/O_1–I/O_4	DATA INPUT/OUTPUT		

DECEMBER, 1979

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... -10°C to 80°C
Storage Temperature ..... -65°C to 150°C
Voltage on any Pin
With Respect to Ground ..... -3.5V to +7V
Power Dissipation 1.0W
D.C. Output Current 5mA

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, unless otherwise noted.

SYMBOL	PARAMETER	2114AL-1/L-2/L-3/L-4 Min.	Typ.[1]	Max.	2114A-4/-5 Min.	Typ.[1]	Max.	UNIT	CONDITIONS
I_{LI}	Input Load Current (All Input Pins)			10			10	μA	V_{IN} = 0 to 5.5V
$\|I_{LO}\|$	I/O Leakage Current			10			10	μA	$\overline{CS}$ = V_{IH} $V_{I/O}$ = GND to VCC
I_{CC}	Power Supply Current		25	40		50	70	mA	V_{CC} = max, $I_{I/O}$ = 0 mA, T_A = 0°C
V_{IL}	Input Low Voltage	-3.0		0.8	-3.0		0.8	V	
V_{IH}	Input High Voltage	2.0		6.0	2.0		6.0	V	
I_{OL}	Output Low Current	2.1	9.0		2.1	9.0		mA	V_{OL} = 0.4V
I_{OH}	Output High Current	-1.0	-2.5		-1.0	-2.5		mA	V_{OH} = 2.4V
I_{OS}[2]	Output Short Circuit Current			40			40	mA	

NOTE: 1. Typical values are for T_A = 25°C and V_{CC} = 5.0V.
2. Duration not to exceed 30 seconds.

CAPACITANCE

T_A = 25°C, f = 1.0 MHz

SYMBOL	TEST	MAX	UNIT	CONDITIONS
$C_{I/O}$	Input/Output Capacitance	5	pF	$V_{I/O}$ = OV
C_{IN}	Input Capacitance	5	pF	V_{IN} = OV

NOTE: This parameter is periodically sampled and not 100% tested.

A.C. CONDITIONS OF TEST

Input Pulse Levels ..... 0.8 Volt to 2.0 Volt
Input Rise and Fall Times ..... 10 nsec
Input and Output Timing Levels ..... 1.5 Volts
Output Load ..... 1 TTL Gate and C_L = 100 pF

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, unless otherwise noted.

READ CYCLE [1]

SYMBOL	PARAMETER	2114AL-1 Min.	2114AL-1 Max.	2114AL-2 Min.	2114AL-2 Max.	2114AL-3 Min.	2114AL-3 Max.	2114A-4/L-4 Min.	2114A-4/L-4 Max.	2114A-5 Min.	2114A-5 Max.	UNIT
t_{RC}	Read Cycle Time	100		120		150		200		250		ns
t_A	Access Time		100		120		150		200		250	ns
t_{CO}	Chip Selection to Output Valid		70		70		70		70		85	ns
t_{CX}	Chip Selection to Output Active	10		10		10		10		10		ns
t_{OTD}	Output 3-state from Deselection		30		35		40		50		60	ns
t_{OHA}	Output Hold from Address Change	15		15		15		15		15		ns

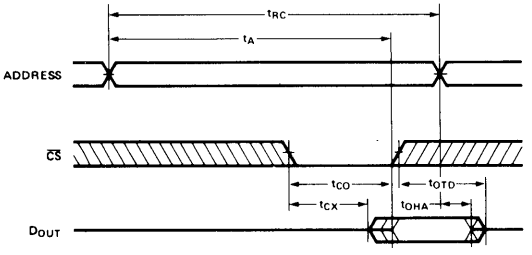
WRITE CYCLE [2]

SYMBOL	PARAMETER	2114AL-1 Min.	2114AL-1 Max.	2114AL-2 Min.	2114AL-2 Max.	2114AL-3 Min.	2114AL-3 Max.	2114A-4/L-4 Min.	2114A-4/L-4 Max.	2114A-5 Min.	2114A-5 Max.	UNIT
t_{WC}	Write Cycle Time	100		120		150		200		250		ns
t_W	Write Time	75		75		90		120		135		ns
t_{WR}	Write Release Time	0		0		0		0		0		ns
t_{OTW}	Output 3-state from Write		30		35		40		50		60	ns
t_{DW}	Data to Write Time Overlap	70		70		90		120		135		ns
t_{DH}	Data Hold from Write Time	0		0		0		0		0		ns

NOTES:
1. A Read occurs during the overlap of a low $\overline{CS}$ and a high $\overline{WE}$.
2. A Write occurs during the overlap of a low $\overline{CS}$ and a low $\overline{WE}$. t_W is measured from the latter of $\overline{CS}$ or $\overline{WE}$ going low to the earlier of $\overline{CS}$ or $\overline{WE}$ going high.

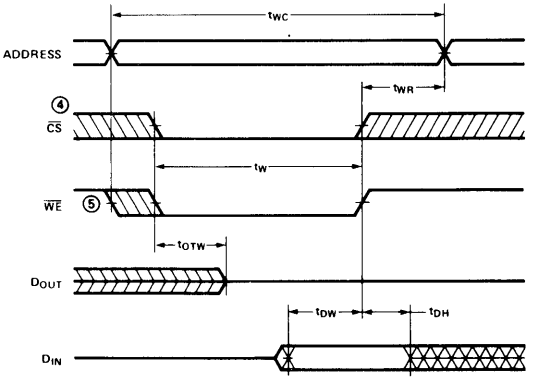
WAVEFORMS

READ CYCLE ③



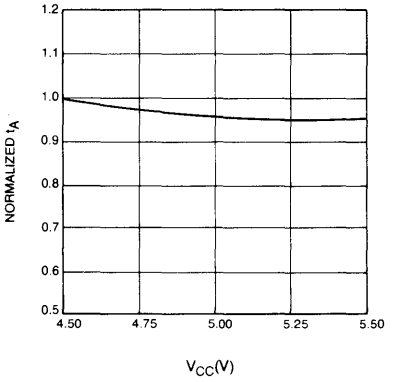
NOTES:
3. $\overline{WE}$ is high for a Read Cycle.
4. If the $\overline{CS}$ low transition occurs simultaneously with the $\overline{WE}$ low transition, the output buffers remain in a high impedance state.
5. $\overline{WE}$ must be high during all address transitions.

WRITE CYCLE

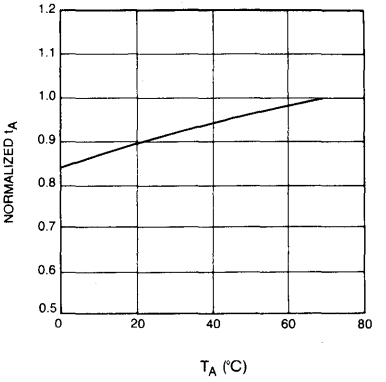


TYPICAL D.C. AND A.C. CHARACTERISTICS

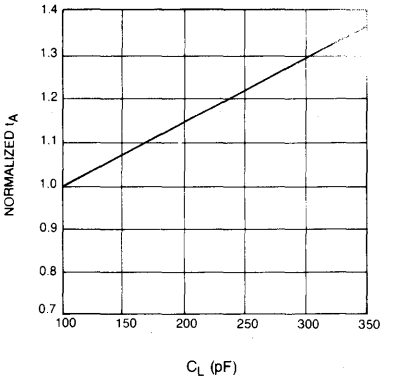
NORMALIZED ACCESS TIME VS. SUPPLY VOLTAGE



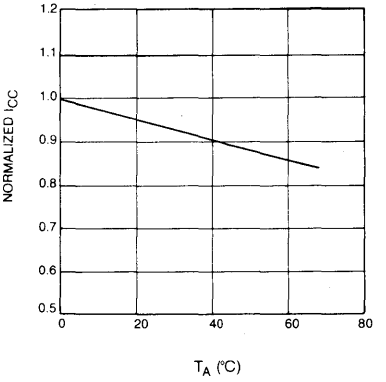
NORMALIZED ACCESS TIME VS. AMBIENT TEMPERATURE



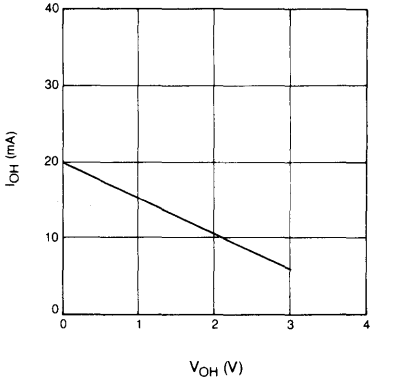
NORMALIZED ACCESS TIME VS. OUTPUT LOAD CAPACITANCE



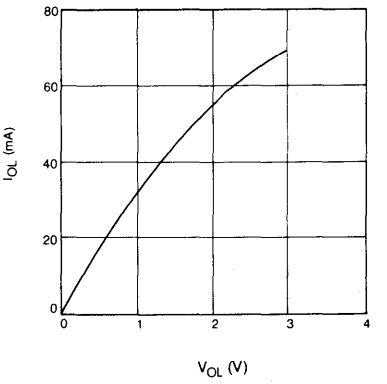
NORMALIZED POWER SUPPLY CURRENT VS. AMBIENT TEMPERATURE



OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



2142
1024 X 4 BIT STATIC RAM

	2142-2	2142-3	2142	2142L2	2142L3	2142L
Max. Access Time (ns)	200	300	450	200	300	450
Max. Power Dissipation (mw)	525	525	525	370	370	370

- **High Density 20 Pin Package**
- **Access Time Selections From 200-450ns**
- **Identical Cycle and Access Times**
- **Low Operating Power Dissipation .1mW/Bit Typical**
- **Single +5V Supply**
- **No Clock or Timing Strobe Required**
- **Completely Static Memory**
- **Directly TTL Compatible: All Inputs and Outputs**
- **Common Data Input and Output Using Three-State Outputs**

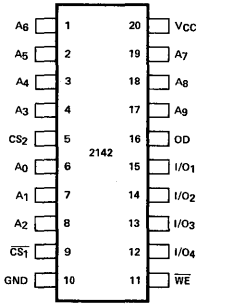
The Intel® 2142 is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using N-channel Silicon-Gate MOS technology. It uses fully DC stable (static) circuitry throughout — in both the array and the decoding — and therefore requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 2142 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives. It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply.

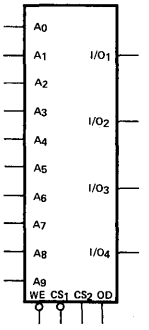
The 2142 is placed in a 20-pin package. Two Chip Selects ($\overline{CS}_1$ and CS_2) are provided for easy and flexible selection of individual packages when outputs are OR-tied. An Output Disable is included for direct control of the output buffers.

The 2142 is fabricated with Intel's N-channel Silicon-Gate technology — a technology providing excellent protection against contamination permitting the use of low cost plastic packaging.

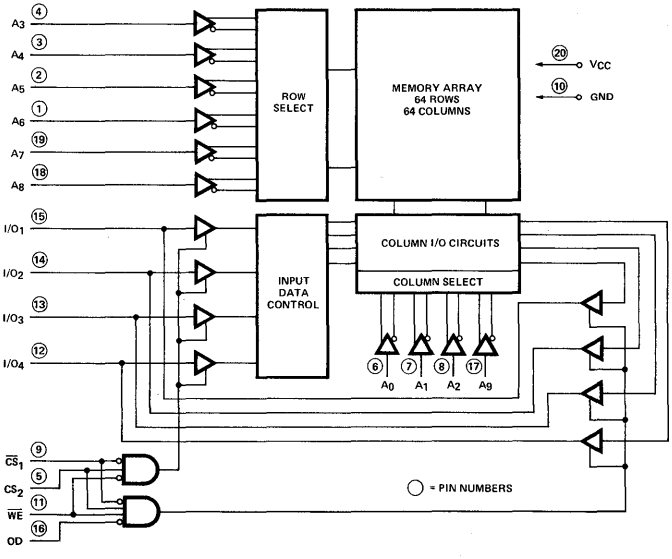
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0–A_9	ADDRESS INPUTS	OD	OUTPUT DISABLE
$\overline{WE}$	WRITE ENABLE	V_{CC}	POWER (+5V)
$\overline{CS}_1$, CS_2	CHIP SELECT	GND	GROUND
I/O_1–I/O_4	DATA INPUT/OUTPUT		

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... -10°C to 80°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin
With Respect to Ground ..... -0.5V to +7V
Power Dissipation 1.0W
D.C. Output Current 10mA

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 5%, unless otherwise noted.

SYMBOL	PARAMETER	2142-2, 2142-3, 2142 Min.	Typ.[1]	Max.	2142L2, 2142L3, 2142L Min.	Typ.[1]	Max.	UNIT	CONDITIONS
I_{LI}	Input Load Current (All Input Pins)			10			10	µA	V_{IN} = 0 to 5.25V
$\|I_{LO}\|$	I/O Leakage Current			10			10	µA	$\overline{CS}$ = 2.4V, $V_{I/O}$ = 0.4V to V_{CC}
I_{CC1}	Power Supply Current		80	95			65	mA	V_{IN} = 5.25V, $I_{I/O}$ = 0 mA, T_A = 25°C
I_{CC2}	Power Supply Current			100			70	mA	V_{IN} = 5.25V, $I_{I/O}$ = 0 mA, T_A = 0°C
V_{IL}	Input Low Voltage	-0.5		0.8	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		6.0	2.0		6.0	V	
I_{OL}	Output Low Current	2.1	6.0		2.1	6.0		mA	V_{OL} = 0.4V
I_{OH}	Output High Current	-1.0	-1.4		-1.0	-1.4		mA	V_{OH} = 2.4V
I_{OS}[2]	Output Short Circuit Current			40			40	mA	$V_{I/O}$ = GND to V_{CC}

NOTE: 1. Typical values are for T_A = 25°C and V_{CC} = 5.0V.
2. Duration not to exceed 30 seconds.

CAPACITANCE

T_A = 25°C, f = 1.0 MHz

SYMBOL	TEST	MAX	UNIT	CONDITIONS
$C_{I/O}$	Input/Output Capacitance	5	pF	$V_{I/O}$ = OV
C_{IN}	Input Capacitance	5	pF	V_{IN} = OV

NOTE: This parameter is periodically sampled and not 100% tested.

A.C. CONDITIONS OF TEST

Input Pulse Levels ..... 0.8 Volt to 2.4 Volt
Input Rise and Fall Times ..... 10 nsec
Input and Output Timing Levels ..... 1.5 Volts
Output Load ..... 1 TTL Gate and C_L = 100 pF

TEST NOTE: This circuit employs a self starting oscillator and a charge pump which require a certain amount of time after POWER ON to start functioning properly. This 2142 circuit is conservatively specified as requiring 500 µsec after V_{CC} reaches its specified limit (4.75V).

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$, unless otherwise noted.

READ CYCLE [1]

SYMBOL	PARAMETER	2142-2, 2142L2 Min.	2142-2, 2142L2 Max.	2142-3, 2142L3 Min.	2142-3, 2142L3 Max.	2142, 2142L Min.	2142, 2142L Max.	UNIT
t_{RC}	Read Cycle Time	200		300		450		ns
t_A	Access Time		200		300		450	ns
t_{OD}	Output Enable to Output Valid		70		100		120	ns
t_{ODX}	Output Enable to Output Active	20		20		20		ns
t_{CO}	Chip Selection to Output Valid		70		100		120	ns
t_{CX}	Chip Selection to Output Active	20		20		20		ns
t_{OTD}	Output 3-state from Disable		60		80		100	ns
t_{OHA}	Output Hold from Address Change	50		50		50		ns

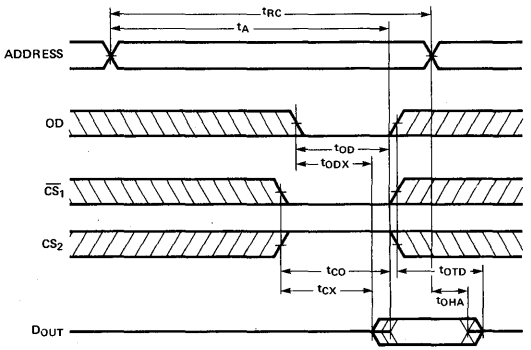
WRITE CYCLE [2]

SYMBOL	PARAMETER	2142-2, 2142L2 Min.	2142-2, 2142L2 Max.	2142-3, 2142L3 Min.	2142-3, 2142L3 Max.	2142, 2142L Min.	2142, 2142L Max.	UNIT
t_{WC}	Write Cycle Time	200		300		450		ns
t_W	Write Time	120		150		200		ns
t_{WR}	Write Release Time	0		0		0		ns
t_{OTD}	Output 3-state from Disable		60		80		100	ns
t_{DW}	Data to Write Time Overlap	120		150		200		ns
t_{DH}	Data Hold From Write Time	0		0		0		ns

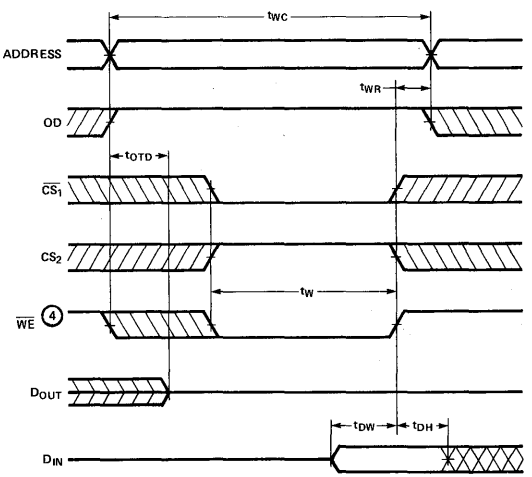
NOTES:
1. A Read occurs during the overlap of a low $\overline{CS}$ and a high $\overline{WE}$.
2. A Write occurs during the overlap of a low $\overline{CS}$ and a low $\overline{WE}$.

WAVEFORMS

READ CYCLE ③



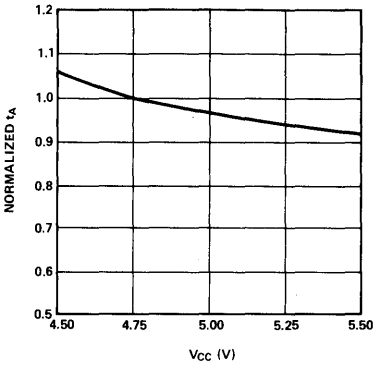
WRITE CYCLE



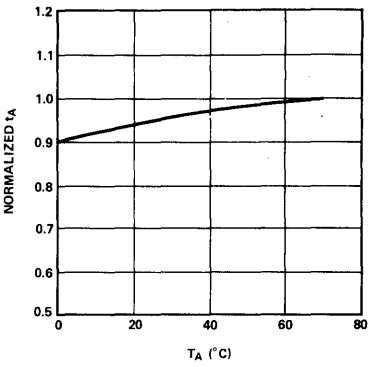
NOTES:
③ $\overline{WE}$ is high for a Read Cycle.
④ $\overline{WE}$ must be high during all address transitions.

TYPICAL D.C. AND A.C. CHARACTERISTICS

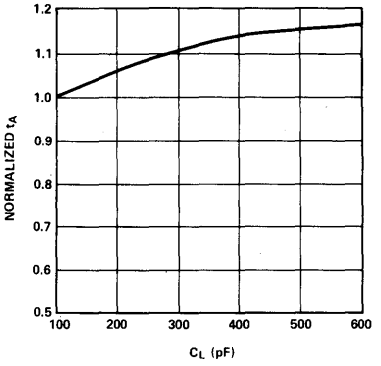
NORMALIZED ACCESS TIME VS. SUPPLY VOLTAGE



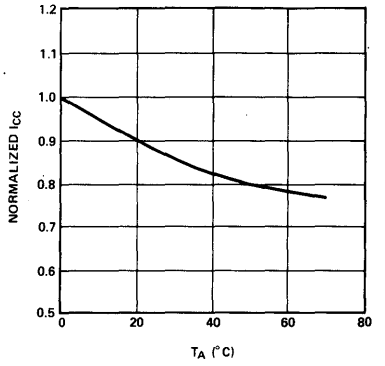
NORMALIZED ACCESS TIME VS. AMBIENT TEMPERATURE



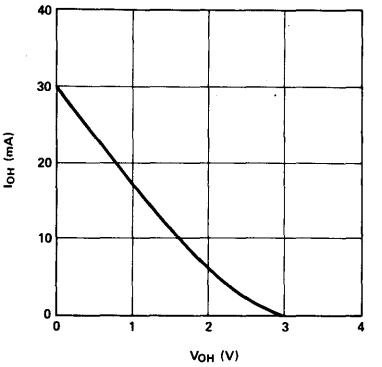
NORMALIZED ACCESS TIME VS. OUTPUT LOAD CAPACITANCE



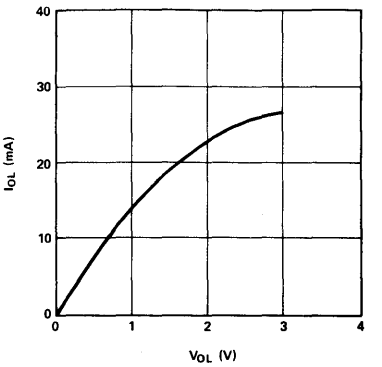
NORMALIZED POWER SUPPLY CURRENT VS. AMBIENT TEMPERATURE



OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



2148
1024 x 4 BIT STATIC RAM

	2148-3	2148	2148-6
Max. Access Time (ns)	55	70	85
Max. Active Current (mA)	125	125	125
Max. Standby Current (mA)	30	30	30

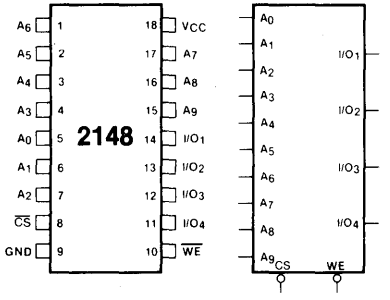
- **HMOS Technology**
- **Completely Static Memory — No Clock or Timing Strobe Required**
- **Equal Access and Cycle Times**
- **Single +5V Supply**
- **Automatic Power-Down**
- **High Density 18-Pin Package**
- **Directly TTL Compatible — All Inputs and Outputs**
- **Common Data Input and Output**
- **Three-State Output**

The Intel® 2148 is a 4096-bit static Random Access Memory organized as 1024 words by 4 bits using HMOS, a high-performance MOS technology. It uses a uniquely innovative design approach which provides the ease-of-use features associated with non-clocked static memories and the reduced standby power dissipation associated with clocked static memories. To the user this means low standby power dissipation without the need for clocks, address setup and hold times, nor reduced data rates due to cycle times that are longer than access times.

$\overline{CS}$ controls the power-down feature. In less than a cycle time after $\overline{CS}$ goes high — disabling the 2148 — the part automatically reduces its power requirements and remains in this low power standby mode as long as $\overline{CS}$ remains high. This device feature results in system power savings as great as 85% in larger systems, where the majority of devices are disabled.

The 2148 is assembled in an 18-pin package configured with the industry standard 1K x 4 pinout. It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. The data is read out nondestructively and has the same polarity as the input data.

PIN CONFIGURATION



LOGIC SYMBOL

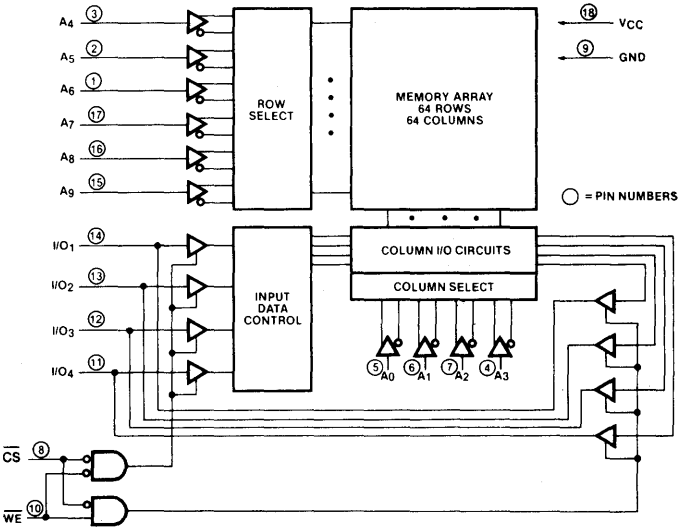
PIN NAMES

A_0-A_9	ADDRESS INPUTS
$\overline{WE}$	WRITE ENABLE
$\overline{CS}$	CHIP SELECT
I/O_1-I/O_4	DATA INPUT/OUTPUT
V_{CC}	POWER (+5V)
GND	GROUND

TRUTH TABLE

$\overline{CS}$	$\overline{WE}$	MODE	I/O	POWER
H	X	NOT SELECTED	HIGH-Z	STANDBY
L	L	WRITE	D_{IN}	ACTIVE
L	H	READ	D_{OUT}	ACTIVE

BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage on Any Pin with
Respect to Ground ..... −3.5V to +7V
D.C. Output Current 20mA
Power Dissipation 1.2W

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS[1]

$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 10\%$ unless otherwise noted.

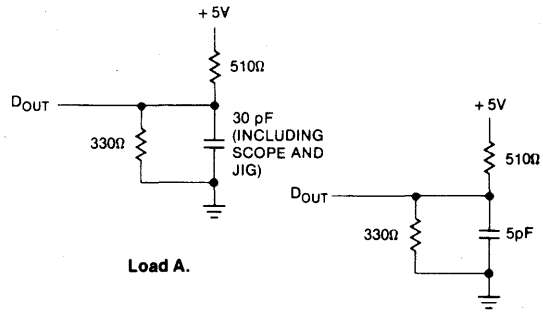
Symbol	Parameter	2148, 2148-3, 2148-6 Min.	Typ.[2]	Max.	Unit	Test Conditions
I_{LI}	Input Load Current (All Input Pins)		0.01	10	µA	V_{CC} = max, V_{IN} = GND to V_{CC}
$\|I_{LO}\|$	Output Leakage Current		0.1	50	µA	$\overline{CS} = V_{IH}$, V_{CC} = max, V_{OUT} = GND to 4.5V
I_{CC}	Operating Current		75	115	mA	$T_A = 25°C$; V_{CC} = max, $\overline{CS} = V_{IL}$, Outputs Open
				125	mA	$T_A = 0°C$
I_{SB}	Standby Current		12	30	mA	V_{CC} = min to max, $\overline{CS} = V_{IH}$
I_{PO}[3]	Peak Power-On Current		25	50	mA	V_{CC} = GND to V_{CC}min, $\overline{CS}$ = Lower of V_{CC} or V_{IH} min
V_{IL}	Input Low Voltage	−3.0		0.8	V	
V_{IH}	Input High Voltage	2.0		6.0	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −2.0mA
I_{OS}	Output Short Circuit Current	TBD		TBD	mA	V_{OUT} = GND to V_{CC}

Notes:

1. The operating ambient temperature range is guaranteed with transverse air flow exceeding 400 linear feet per minute.
2. Typical limits are at $V_{CC} = 5V$, $T_A = +25°C$, and Load A.
3. A pull-up resistor to V_{CC} on the $\overline{CS}$ input is required to keep the device deselected; otherwise, power-on current approaches I_{CC} active.

A.C. TEST CONDITIONS

Input Pulse Levels	GND to 3.0 Volts
Input Rise and Fall Times	10 nsec
Input and Output Timing Reference Levels	1.5 Volts
Output Load	See Load A.



Load A.

Load B.

CAPACITANCE [4]

$T_A = 25°C$, f = 1.0MHz

Symbol	Parameter	Max.	Unit	Conditions
C_{IN}	Input Capacitance	5	pF	V_{IN} = 0V
C_{OUT}	Output Capacitance	7	pF	V_{OUT} = 0V

Note 4. This parameter is sampled and not 100% tested.

A.C. CHARACTERISTICS

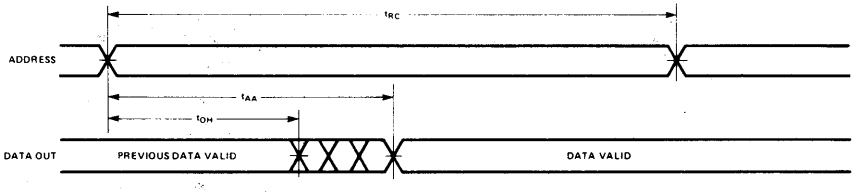
$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 10\%$ unless otherwise noted.

READ CYCLE

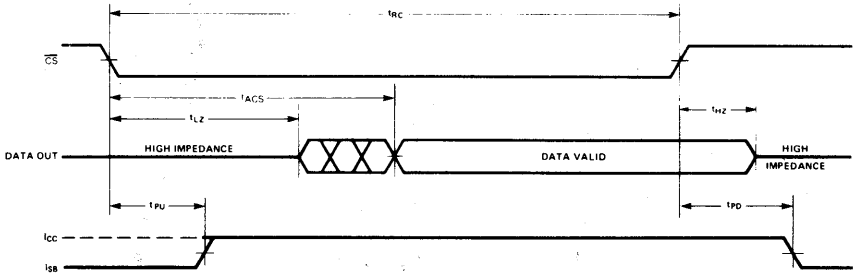
Symbol	Parameter	2148-3 Min.	2148-3 Max.	2148 Min.	2148 Max.	2148-6 Min.	2148-6 Max.	Unit	Test Conditions
t_{RC}	Read Cycle Time	55		70		85		ns	
t_{AA}	Address Access Time		55		70		85	ns	
t_{ACS1}	Chip Select Access Time		55		70		85	ns	Note 1
t_{ACS2}	Chip Select Access Time		65		80		95	ns	Note 2
t_{OH}	Output Hold from Address Change	5		5		5		ns	
t_{LZ}[6]	Chip Selection Output in Low Z	10		10		10		ns	Note 7
t_{HZ}[6]	Chip Deselection to Output in High Z	0	40	0	40	0	40	ns	Note 7
t_{PU}	Chip Selection to Power Up Time	0		0		0		ns	
t_{PD}	Chip Deselection to Power Down Time		30		30		30	ns	

WAVEFORMS

READ CYCLE NO. 1[3, 4]



READ CYCLE NO. 2[3, 5]



Notes:

1. Chip deselected for greater than 55ns prior to $\overline{CS}$ transition low.
2. Chip deselected for a finite time that is less than 55ns prior to $\overline{CS}$ transition low. (If the deselect time is 0ns, the chip is by definition selected and access occurs according to Read Cycle No. 1.)
3. $\overline{WE}$ is high for Read Cycles.
4. Device is continuously selected, $\overline{CS} = V_{IL}$.
5. Addresses valid prior to or coincident with $\overline{CS}$ transition low.
6. At any given temperature and voltage condition, t_{HZ} max. is less than t_{LZ} min. both for a given device and from device to device.
7. Transition is measured ±500mV from high impedance voltage with Load B. This parameter is sampled and not 100% tested.

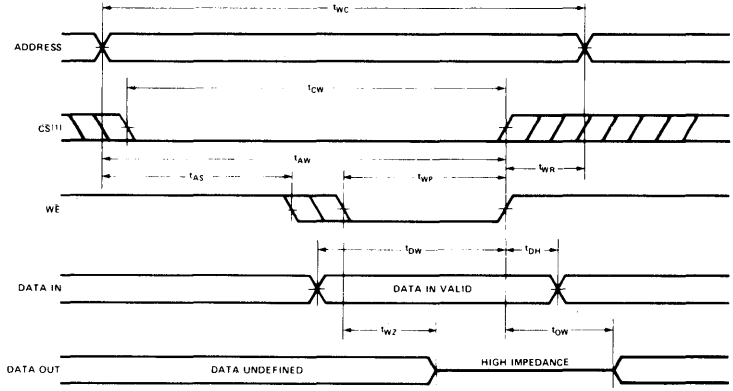
A.C. CHARACTERISTICS (continued)

WRITE CYCLE

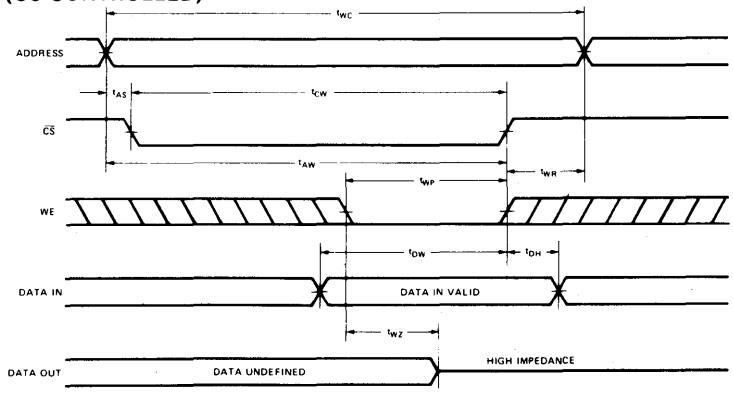
Symbol	Parameter	2148-3 Min.	2148-3 Max.	2148 Min.	2148 Max.	2148-6 Min.	2148-6 Max.	Unit	Test Conditions
t_{WC}	Write Cycle Time	55		70		85		ns	
t_{CW}	Chip Selection to End of Write	50		65		80		ns	
t_{AW}	Address Valid to End of Write	50		65		80		ns	
t_{AS}	Address Setup Time	0		0		0		ns	
t_{WP}	Write Pulse Width	40		50		60		ns	
t_{WR}	Write Recovery Time	5		5		5		ns	
t_{DW}	Data Valid to End of Write	25		25		30		ns	
t_{DH}	Data Hold Time	5		5		5		ns	
t_{WZ}	Write Enabled to Output in High Z	0	15	0	25	0	30	ns	Note 2
t_{OW}	Output Active from End of Write	0		0		0		ns	Note 2

WAVEFORMS

WRITE CYCLE #1 ($\overline{WE}$ CONTROLLED)



WRITE CYCLE #2 ($\overline{CS}$ CONTROLLED)



Notes: 1. If $\overline{CS}$ goes high simultaneously with $\overline{WE}$ high, the output remains in a high impedance state.
2. Transition is measured ±500 mV from high impedance voltage with Load B. This parameter is sampled and not 100% tested.

2148H
1024 x 4-BIT STATIC RAM

	2148H-3	2148H	2148HL-3	2148HL
Maximum Access Time (ns)	55	70	55	70
Maximum Active Current (mA)	180	180	125	125
Maximum Standby Current (mA)	30	30	20	20

- **Automatic Power-Down**
- **Industry Standard 2114A and 2148 Pinout**
- **HMOS II Technology**
- **Functionally Compatible to the 2148**
- **Completely Static Memory—No Clock or Timing Strobe Required**
- **Equal Access and Cycle Times**
- **High Density 18-Pin Package**
- **Common Data Input and Output**
- **Three-State Output**
- **Single +5V Supply**
- **Fast Chip Select Access 2149H Available**

The Intel® 2148H is a 4096-bit static Random Access Memory organized as 1024 words by 4 bits using HMOS II, a high performance MOS technology. It uses a uniquely innovative design approach which provides the ease-of-use features associated with non-clocked static memories and the reduced standby power dissipation associated with clocked static memories. To the user this means low standby power dissipation without the need for clocks, address setup and hold times, or reduced data rates due to cycle times that are longer than access times.

$\overline{CS}$ controls the power-down feature. In less than a cycle time after $\overline{CS}$ goes high—disabling the 2148H—the part automatically reduces its power requirements and remains in this low power standby mode as long as $\overline{CS}$ remains high. This device feature results in system power savings as great as 85% in larger systems, where the majority of devices are disabled. A non-power-down companion, the 2149H, is available to provide a fast chip select access time for speed critical applications.

The 2148H is assembled in an 18-pin package configured with the industry standard 1K x 4 pinout. It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. The data is read out non-destructively and has the same polarity as the input data.

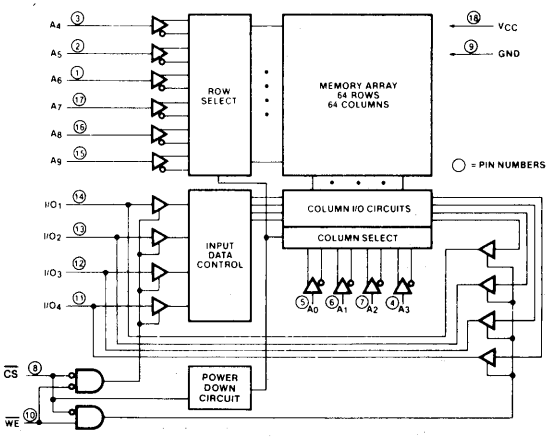
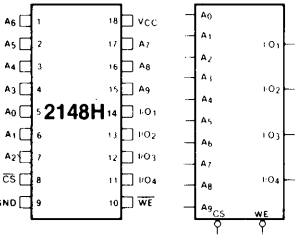


Figure 1. 2148H Block Diagram



PIN NAMES

A_0-A_9	ADDRESS INPUTS
$\overline{WE}$	WRITE ENABLE
$\overline{CS}$	CHIP SELECT
I/O_1-I/O_4	DATA INPUT/OUTPUT
V_{CC}	POWER (+5V)
GND	GROUND

TRUTH TABLE

$\overline{CS}$	$\overline{WE}$	MODE	I/O	POWER
H	X	NOT SELECTED	HIGH-Z	STANDBY
L	L	WRITE	D_{IN}	ACTIVE
L	H	READ	D_{OUT}	ACTIVE

Figure 2. 2148H Pin Diagram

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage on Any Pin with
Respect to Ground ..... −3.5V to +7V
D.C. Continuous Output Current ..... 20 mA
Power Dissipation 1.2W

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. AND OPERATING CHARACTERISTICS[1]

$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 10\%$ unless otherwise noted.

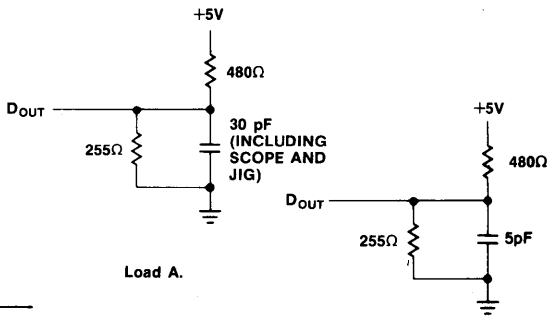
Symbol	Parameter	2148H/H-3 Min.	2148H/H-3 Typ[2]	2148H/H-3 Max.	2148HL/HL-3 Min.	2148HL/HL-3 Typ[2]	2148HL/HL-3 Max.	Unit	Test Conditions
I_{LI}	Input Load Current (All Input Pins)		0.01	10		0.01	10	μA	V_{CC} = max, V_{IN} = GND to V_{CC}
$\|I_{LO}\|$	Output Leakage Current		0.1	50		0.1	50	μA	$\overline{CS}$ = V_{IH}, V_{CC} = max, V_{OUT} = GND to 4.5V
I_{CC}	Operating Current		120	180		90	125	mA	V_{CC} = max, $\overline{CS}$ = V_{IL}, Outputs Open
I_{SB}	Standby Current		15	30		10	20	mA	V_{CC} = min to max, $\overline{CS}$ = V_{IH}
I_{PO}[3]	Peak Power-On Current		25	50		15	30	mA	V_{CC} = GND to V_{CC} min, $\overline{CS}$ = Lower of V_{CC} or V_{IH} min
V_{IL}	Input Low Voltage	−3.0		0.8	−3.0		0.8	V	
V_{IH}	Input High Voltage	2.1		6.0	2.1		6.0	V	
V_{OL}	Output Low Voltage			0.4			0.4	V	I_{OL} = 8 mA
V_{OH}	Output High Voltage	2.4			2.4			V	I_{OH} = −4.0 mA
I_{OS}	Output Short Circuit Current		±150	±200		±150	±200	mA	V_{OUT} = GND to V_{CC}

Notes:

1. The operating ambient temperature range is guaranteed with transverse air flow exceeding 400 linear feet per minute. Typical thermal resistance values of the package at maximum temperatures are:
   θ_{JA} (@ 400 f_{PM} air flow) = 40° C/W
   θ_{JA} (still air) = 70° C/W
   θ_{JC} = 25° C/W
2. Typical limits are at V_{CC} = 5V, T_A = +25°C, and Load A.
3. A pull-up resistor to V_{CC} on the $\overline{CS}$ input is required to keep the device deselected during power-on. Otherwise, power-on current approaches I_{CC} active.

A.C. TEST CONDITIONS

Input Pulse Levels	GND to 3.0 Volts
Input Rise and Fall Times	5 nsec
Input and Output Timing Reference Levels	1.5 Volts
Output Load	See Load A.



Load A. Load B.

CAPACITANCE [4]

$T_A = 25°C$, f = 1.0 MHz

Symbol	Parameter	Max.	Unit	Conditions
C_{IN}	Address/Control Capacitance	5	pF	V_{IN} = 0V
C_{IO}	Input/Output Capacitance	7	pF	V_{OUT} = 0V

Note 4. This parameter is sampled and not 100% tested.

A.C. CHARACTERISTICS

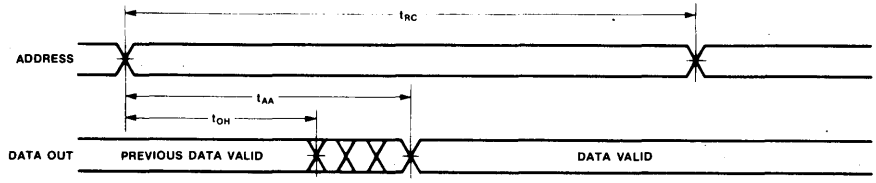
$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 10\%$ unless otherwise noted.

READ CYCLE

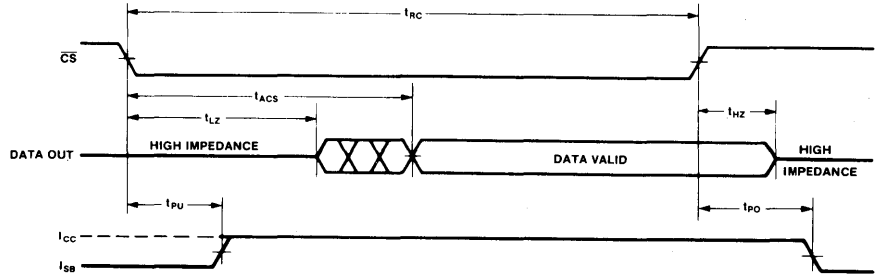
Symbol	Parameter	2148H-3/HL-3 Min.	2148H-3/HL-3 Max.	2148H/HL Min.	2148H/HL Max.	Unit	Test Conditions
t_{RC}	Read Cycle Time	55		70		ns	
t_{AA}	Address Access Time		55		70	ns	
t_{ACS1}	Chip Select Access Time		55		70	ns	Note 1
t_{ACS2}	Chip Select Access Time		65		80	ns	Note 2
t_{OH}	Output Hold from Address Change	5		5		ns	
t_{LZ}	Chip Selection Output in Low Z	20		20		ns	Note 6
t_{HZ}	Chip Deselection to Output in High Z	0	20	0	20	ns	Note 6
t_{PU}	Chip Selection to Power Up Time	0		0		ns	
t_{PD}	Chip Deselection to Power Down Time		30		30	ns	

WAVEFORMS

READ CYCLE NO. 1[3, 4]



READ CYCLE NO. 2[3, 5]



Notes:

1. Chip deselected for greater than 55 ns prior to $\overline{CS}$ transition low.
2. Chip deselected for a finite time that is less than 55 ns prior to $\overline{CS}$ transition low. (If the deselect time is 0 ns, the chip is by definition selected and access occurs according to Read Cycle No. 1.)
3. $\overline{WE}$ is high for Read Cycles.
4. Device is continuously selected, $\overline{CS} = V_{IL}$.
5. Addresses valid prior to or coincident with $\overline{CS}$ transition low.
6. Transition is measured ±500mV from high impedance voltage with Load B. This parameter is sampled and not 100% tested.

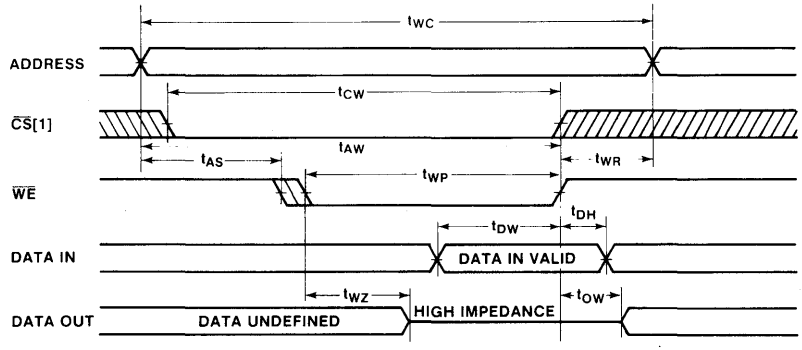
A.C. CHARACTERISTICS (continued)

WRITE CYCLE

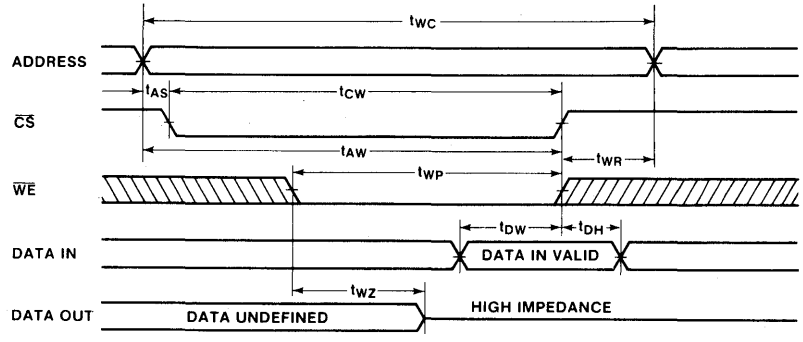
Symbol	Parameter	2148H-3/HL-3 Min.	2148H-3/HL-3 Max.	2148H/HL Min.	2148H/HL Max.	Unit	Test Conditions
t_{WC}	Write Cycle Time	55		70		ns	
t_{CW}	Chip Selection to End of Write	50		65		ns	
t_{AW}	Address Valid to End of Write	50		65		ns	
t_{AS}	Address Setup Time	0		0		ns	
t_{WP}	Write Pulse Width	40		50		ns	
t_{WR}	Write Recovery Time	5		5		ns	
t_{DW}	Data Valid to End of Write	20		25		ns	
t_{DH}	Data Hold Time	0		0		ns	
t_{WZ}	Write Enabled to Output in High Z	0	20	0	25	ns	Note 2
t_{OW}	Output Active from End of Write	0		0		ns	Note 2

WAVEFORMS

WRITE CYCLE No. 1 ($\overline{WE}$ CONTROLLED)



WRITE CYCLE No. 2 ($\overline{CS}$ CONTROLLED)



Notes: 1. If $\overline{CS}$ goes high simultaneously with $\overline{WE}$ high, the output remains in a high impedance state.

2. Transition is measured ±500mV from high impedance voltage with Load B. This parameter is sampled and not 100% tested.

2118 FAMILY
16,384 x 1 BIT DYNAMIC RAM

	2118-3	2118-4	2118-7
Maximum Access Time (ns)	100	120	150
Read, Write Cycle (ns)	235	270	320
Read-Modify-Write Cycle (ns)	285	320	410

- **Single +5V Supply, ±10% Tolerance**
- **HMOS Technology**
- **Low Power: 150 mW Max. Operating**
  11 mW Max. Standby
- **Low V_{DD} Current Transients**
- **All Inputs, Including Clocks, TTL Compatible**
- **$\overline{CAS}$ Controlled Output is Three-State, TTL Compatible**
- **$\overline{RAS}$ Only Refresh**
- **128 Refresh Cycles Required Every 2ms**
- **Page Mode and Hidden Refresh Capability**
- **Allows Negative Overshoot V_{IL} min = -2V**

The Intel® 2118 is a 16,384 word by 1-bit Dynamic MOS RAM designed to operate from a single +5V power supply. The 2118 is fabricated using HMOS — a production proven process for high performance, high reliability, and high storage density.

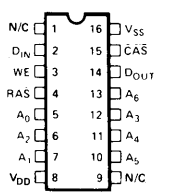
The 2118 uses a single transistor dynamic storage cell and advanced dynamic circuitry to achieve high speed with low power dissipation. The circuit design minimizes the current transients typical of dynamic RAM operation. These low current transients contribute to the high noise immunity of the 2118 in a system environment.

Multiplexing the 14 address bits into the 7 address input pins allows the 2118 to be packaged in the industry standard 16-pin DIP. The two 7-bit address words are latched into the 2118 by the two TTL clocks, Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$). Non-critical timing requirements for $\overline{RAS}$ and $\overline{CAS}$ allow use of the address multiplexing technique while maintaining high performance.

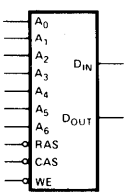
The 2118 three-state output is controlled by $\overline{CAS}$, independent of $\overline{RAS}$. After a valid read or read-modify-write cycle, data is latched on the output by holding $\overline{CAS}$ low. The data out pin is returned to the high impedance state by returning $\overline{CAS}$ to a high state. The 2118 hidden refresh feature allows $\overline{CAS}$ to be held low to maintain latched data while $\overline{RAS}$ is used to execute $\overline{RAS}$-only refresh cycles.

The single transistor storage cell requires refreshing for data retention. Refreshing is accomplished by performing $\overline{RAS}$-only refresh cycles, hidden refresh cycles, or normal read or write cycles on the 128 address combinations of A_0 through A_6 during a 2ms period. A write cycle will refresh stored data on all bits of the selected row except the bit which is addressed.

PIN CONFIGURATION

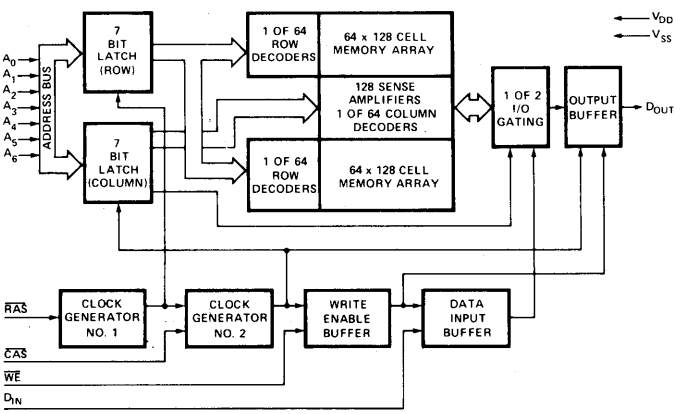


LOGIC SYMBOL



A_0-A_6	ADDRESS INPUTS
$\overline{CAS}$	COLUMN ADDRESS STROBE
D_{IN}	DATA IN
D_{OUT}	DATA OUT
$\overline{WE}$	WRITE ENABLE
$\overline{RAS}$	ROW ADDRESS STROBE
V_{DD}	POWER (+5V)
V_{SS}	GROUND

BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ... -10°C to +80°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin Relative to V_{SS} 7.5V
Data Out Current 50mA
Power Dissipation 1.0W

COMMENT:

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS[1]

T_A = 0°C to 70°C, V_{DD} = 5V ±10%, V_{SS} = 0V, unless otherwise noted.

Symbol	Parameter	Limits Min.	Typ.[2]	Max.	Unit	Test Conditions	Notes
$\|I_{LI}\|$	Input Load Current (any input)		0.1	10	µA	V_{IN}=V_{SS} to V_{DD}	
$\|I_{LO}\|$	Output Leakage Current for High Impedance State		0.1	10	µA	Chip Deselected: $\overline{CAS}$ at V_{IH}, V_{OUT} = 0 to 5.5V	
I_{DD1}	V_{DD} Supply Current, Standby		1.2	2	mA	$\overline{CAS}$ and $\overline{RAS}$ at V_{IH}	
I_{DD2}	V_{DD} Supply Current, Operating		23	27	mA	2118-3, t_{RC} = t_{RCMIN}	3
			21	25	mA	2118-4, t_{RC} = t_{RCMIN}	3
			19	23	mA	2118-7, t_{RC} = t_{RCMIN}	3
I_{DD3}	V_{DD} Supply Current; $\overline{RAS}$-Only Cycle		16	18	mA	2118-3, t_{RC} = t_{RCMIN}	3
			14	16	mA	2118-4, t_{RC} = t_{RCMIN}	3
			12	14	mA	2118-7, t_{RC} = t_{RCMIN}	3
I_{DD5}	V_{DD} Supply Current, Standby, Output Enabled		2	4	mA	$\overline{CAS}$ at V_{IL}, $\overline{RAS}$ at V_{IH}	3
V_{IL}	Input Low Voltage (all inputs)	-2.0		0.8	V		
V_{IH}	Input High Voltage (all inputs)	2.4		7.0	V		
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 4.2mA	
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -5mA	

NOTES:
1. All voltages referenced to V_{SS}.
2. Typical values are for T_A = 25°C and nominal supply voltages.
3. I_{DD} is dependent on output loading when the device output is selected. Specified $I_{DD\ MAX}$ is measured with the output open.

CAPACITANCE[1]

T_A = 25°C, V_{DD} = 5V ±10%, V_{SS} = 0V, unless otherwise noted.

Symbol	Parameter	Typ.	Max.	Unit
C_{I1}	Address, Data In	3	5	pF
C_{I2}	$\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, Data Out	4	7	pF

NOTES:
1. Capacitance measured with Boonton Meter or effective capacitance calculated from the equation:
$C = \frac{I\Delta t}{\Delta V}$ with ΔV equal to 3 volts and power supplies at nominal levels.

A.C. CHARACTERISTICS[1,2,3]

T_A = 0°C to 70°C, V_{DD} = 5V ±10%, V_{SS} = 0V, unless otherwise noted.

READ, WRITE, READ-MODIFY-WRITE AND REFRESH CYCLES

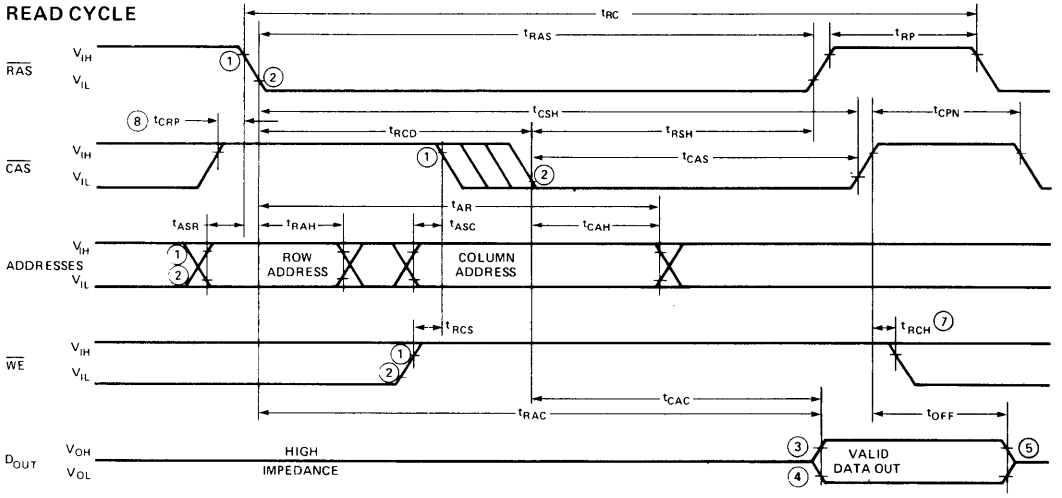
Symbol	Parameter	2118-3 Min.	2118-3 Max.	2118-4 Min.	2118-4 Max.	2118-7 Min.	2118-7 Max.	Unit	Notes
t_{RAC}	Access Time From $\overline{RAS}$		100		120		150	ns	4,5
t_{CAC}	Access Time From $\overline{CAS}$		55		65		80	ns	4,5,6
t_{REF}	Time Between Refresh		2		2		2	ms	
t_{RP}	$\overline{RAS}$ Precharge Time	110		120		135		ns	
t_{CPN}	$\overline{CAS}$ Precharge Time(non-page cycles)	50		55		70		ns	
t_{CRP}	$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	0		0		0		ns	
t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	25	45	25	55	25	70	ns	7
t_{RSH}	$\overline{RAS}$ Hold Time	70		85		105		ns	
t_{CSH}	$\overline{CAS}$ Hold Time	100		120		165		ns	
t_{ASR}	Row Address Set-Up Time	0		0		0		ns	
t_{RAH}	Row Address Hold Time	15		15		15		ns	
t_{ASC}	Column Address Set-Up Time	0		0		0		ns	
t_{CAH}	Column Address Hold Time	15		15		20		ns	
t_{AR}	Column Address Hold Time, to $\overline{RAS}$	60		70		90		ns	
t_T	Transition Time (Rise and Fall)	3	50	3	50	3	50	ns	8
t_{OFF}	Output Buffer Turn Off Delay	0	45	0	50	0	60	ns	
READ AND REFRESH CYCLES									
t_{RC}	Random Read Cycle Time	235		270		320		ns	
t_{RAS}	$\overline{RAS}$ Pulse Width	115	10000	140	10000	175	10000	ns	
t_{CAS}	$\overline{CAS}$ Pulse Width	55	10000	65	10000	95	10000	ns	
t_{RCS}	Read Command Set-Up Time	0		0		0		ns	
t_{RCH}	Read Command Hold Time	0		0		0		ns	
WRITE CYCLE									
t_{RC}	Random Write Cycle Time	235		270		320		ns	
t_{RAS}	$\overline{RAS}$ Pulse Width	115	10000	140	10000	175	10000	ns	
t_{CAS}	$\overline{CAS}$ Pulse Width	55	10000	65	10000	95	10000	ns	
t_{WCS}	Write Command Set-Up Time	0		0		0		ns	9
t_{WCH}	Write Command Hold Time	25		30		45		ns	
t_{WCR}	Write Command Hold Time, to $\overline{RAS}$	70		85		115		ns	
t_{WP}	Write Command Pulse Width	25		30		50		ns	
t_{RWL}	Write Command to $\overline{RAS}$ Lead Time	60		65		110		ns	
t_{CWL}	Write Command to $\overline{CAS}$ Lead Time	45		50		100		ns	
t_{DS}	Data-In Set-Up Time	0		0		0		ns	
t_{DH}	Data-In Hold Time	25		30		45		ns	
t_{DHR}	Data-In Hold Time, to $\overline{RAS}$	70		85		115		ns	
READ-MODIFY-WRITE CYCLE									
t_{RWC}	Read-Modify-Write Cycle Time	285		320		410		ns	
t_{RRW}	RMW Cycle $\overline{RAS}$ Pulse Width	165	10000	190	10000	265	10000	ns	
t_{CRW}	RMW Cycle $\overline{CAS}$ Pulse Width	105	10000	120	10000	185	10000	ns	
t_{RWD}	$\overline{RAS}$ to $\overline{WE}$ Delay	100		120		150		ns	9
t_{CWD}	$\overline{CAS}$ to $\overline{WE}$ Delay	55		65		80		ns	9

NOTES:

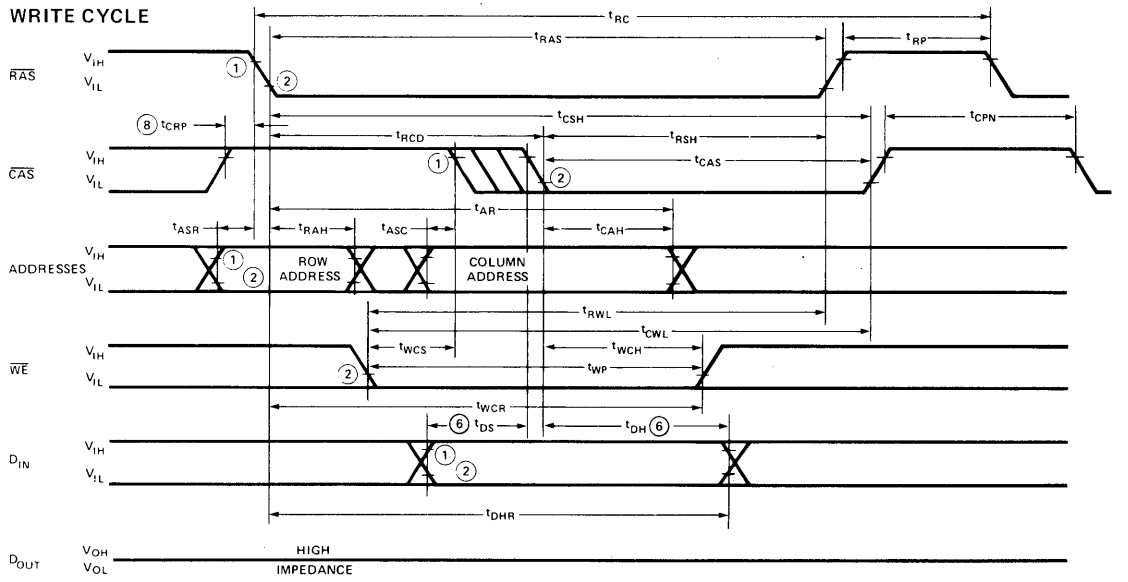
1. All voltages referenced to V_{SS}.
2. Eight cycles are required after power-up or prolonged periods (greater than 2ms) of $\overline{RAS}$ inactivity before proper device operation is achieved. Any 8 cycles which perform refresh are adequate for this purpose.
3. A.C. Characteristics assume t_T = 5ns.
4. Assume that $t_{RCD} \leq t_{RCD}$ (max.). If t_{RCD} is greater than t_{RCD} (max.) then t_{RAC} will increase by the amount that t_{RCD} exceeds t_{RCD} (max.).
5. Load = 2 TTL loads and 100pF.
6. Assumes $t_{RCD} \geq t_{RCD}$ (max.).
7. t_{RCD} (max.) is specified as a reference point only: if t_{RCD} is less than t_{RCD} (max.) access time is t_{RAC}. If t_{RCD} is greater than t_{RCD} (max.) access time is t_{RCD} + t_{CAC}.
8. t_T is measured between V_{IH} (min.) and V_{IL} (max.).
9. t_{WCS}, t_{CWD} and t_{RWD} are specified as reference points only. If $t_{WCS} \geq t_{WCS}$ (min.) the cycle is an early write cycle and the data out pin will remain high impedance throughout the entire cycle. If $t_{CWD} \geq t_{CWD}$ (min.) and $t_{RWD} \geq t_{RWD}$ (min.), the cycle is a read-modify-write cycle and the data out will contain the data read from the selected address. If neither of the above conditions is satisfied, the condition of the data out is indeterminate.

WAVEFORMS

READ CYCLE



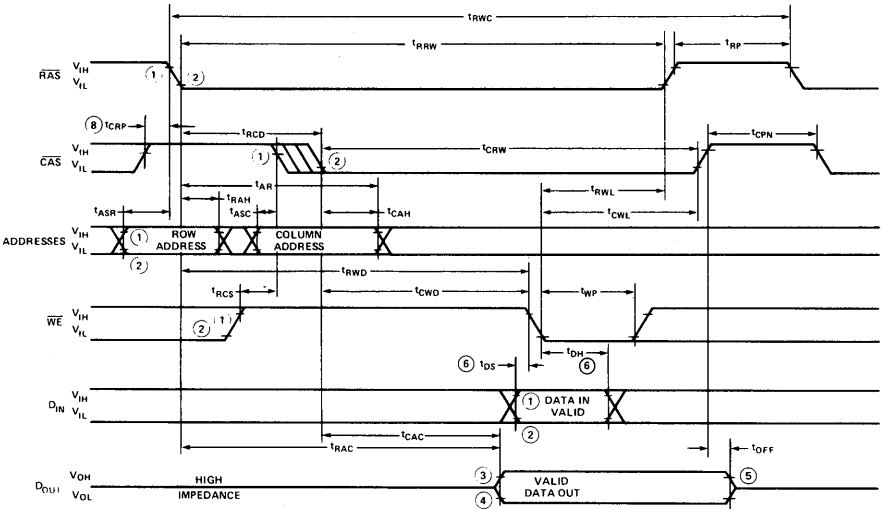
WRITE CYCLE



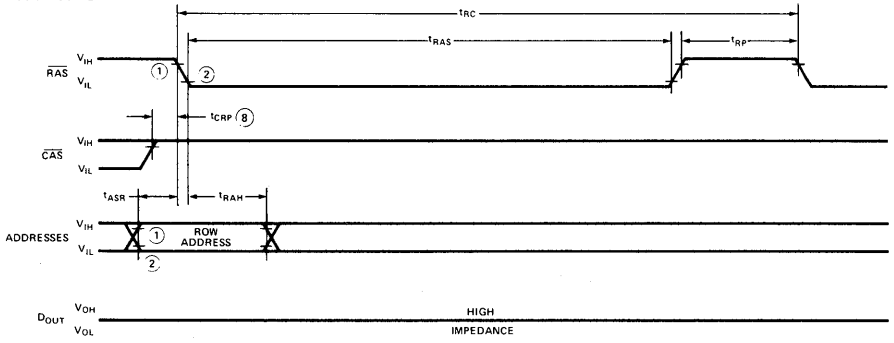
NOTES: 1,2. $V_{IH\ MIN}$ AND $V_{IL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF INPUT SIGNALS.
3,4. $V_{OH\ MIN}$ AND $V_{OL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF D_{OUT}.
5. t_{OFF} IS MEASURED TO $|I_{OUT}| \leq |I_{LO}|$.
6. t_{DS} AND t_{DH} ARE REFERENCED TO $\overline{CAS}$ OR $\overline{WE}$, WHICHEVER OCCURS LAST.
7. t_{RCH} IS REFERENCED TO THE TRAILING EDGE OF $\overline{CAS}$ OR $\overline{RAS}$, WHICHEVER OCCURS FIRST.
8. t_{CRP} REQUIREMENT IS ONLY APPLICABLE FOR $\overline{RAS}$/$\overline{CAS}$ CYCLES PRECEEDED BY A $\overline{CAS}$-ONLY CYCLE (i.e., FOR SYSTEMS WHERE $\overline{CAS}$ HAS NOT BEEN DECODED WITH $\overline{RAS}$).

WAVEFORMS

READ-MODIFY-WRITE CYCLE

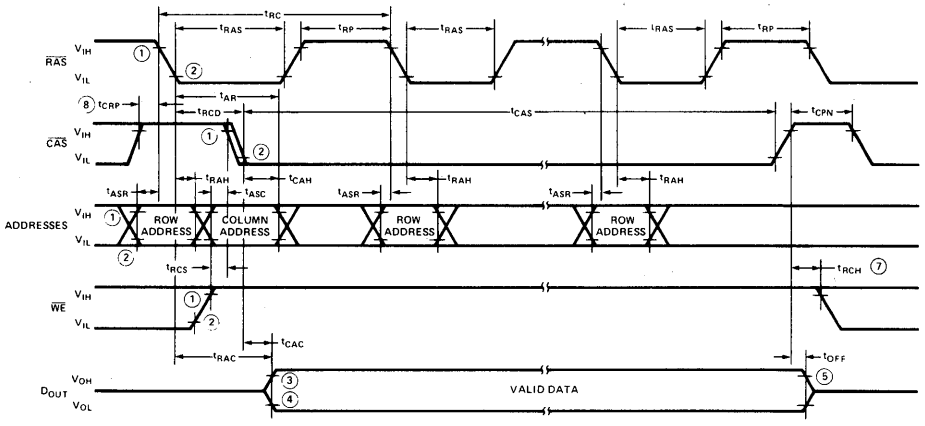


RAS-ONLY REFRESH CYCLE



HIDDEN REFRESH CYCLE

(For Hidden Refresh Operation order 2118-3 S6445, 2118-4 S6446 or 2118-7 S6447)



NOTES:
1,2. $V_{IH\ MIN}$ AND $V_{IL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF INPUT SIGNALS.
3,4. $V_{OH\ MIN}$ AND $V_{OL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF D_{OUT}.
5. t_{OFF} IS MEASURED TO $|I_{OUT}| \leq |I_{LO}|$.
6. t_{DS} AND t_{DH} ARE REFERENCED TO $\overline{CAS}$ OR $\overline{WE}$, WHICHEVER OCCURS LAST.
7. t_{RCH} IS REFERENCED TO THE TRAILING EDGE OF $\overline{CAS}$ OR $\overline{RAS}$, WHICHEVER OCCURS FIRST.
8. t_{CRP} REQUIREMENT IS ONLY APPLICABLE FOR $\overline{RAS}$/$\overline{CAS}$ CYCLES PRECEEDED BY A $\overline{CAS}$-ONLY CYCLE (i.e., FOR SYSTEMS WHERE $\overline{CAS}$ HAS NOT BEEN DECODED WITH $\overline{RAS}$).

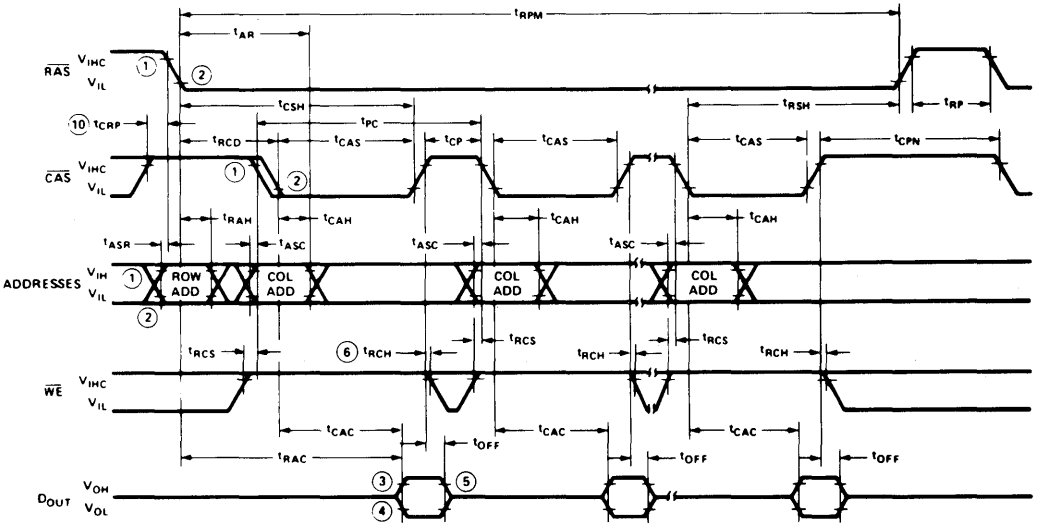
D.C. AND A.C. CHARACTERISTICS, PAGE MODE [7,8,11]

T_A = 0°C to 70°C, V_{DD} = 5V ±10%, V_{SS} = 0V, unless otherwise noted.
For Page Mode Operation order 2118-3 S6329, 2118-4 S6330, or 2118-7 S6331.

Symbol	Parameter	2118-3 S6329 Min.	2118-3 S6329 Max.	2118-4 S6330 Min.	2118-4 S6330 Max.	2118-7 S6331 Min.	2118-7 S6331 Max.	Unit	Notes
t_{PC}	Page Mode Read or Write Cycle	125		145		190		ns	
t_{PCM}	Page Mode Read Modify Write Cycle	175		200		280		ns	
t_{CP}	$\overline{CAS}$ Precharge Time, Page Cycle	60		70		85		ns	
t_{RPM}	$\overline{RAS}$ Pulse Width, Page Mode	115	10000	140	10000	175	10000	ns	
t_{CAS}	$\overline{CAS}$ Pulse Width	55	10000	65	10000	95	10000	ns	
I_{DD4}	V_{DD} Supply Current Page Mode, Minimum t_{PC}, Minimum t_{CAS}		20		17		15	mA	

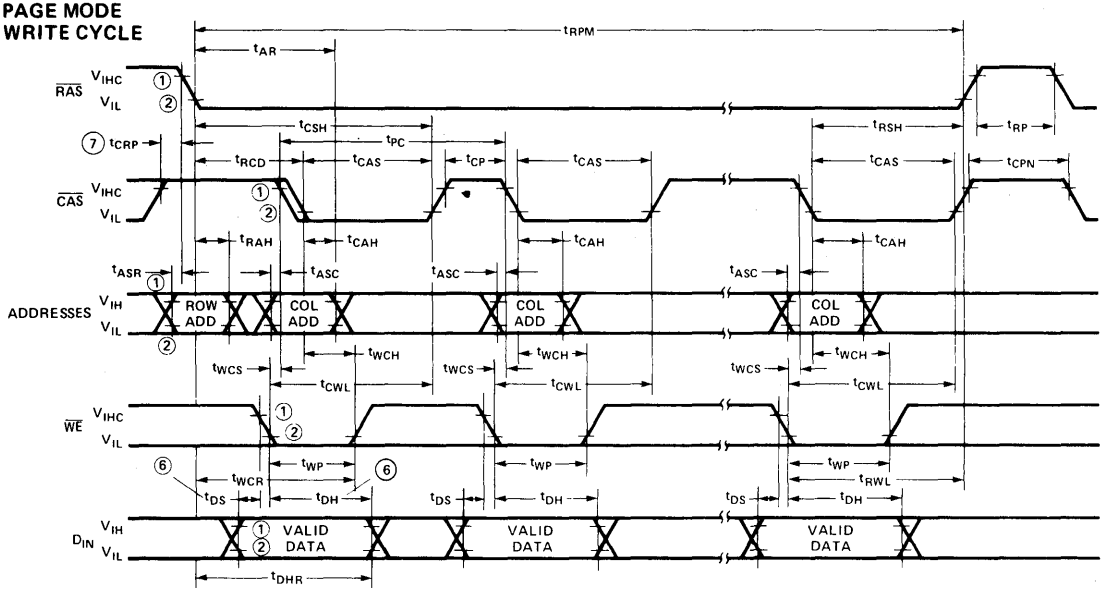
WAVEFORMS

PAGE MODE READ CYCLE

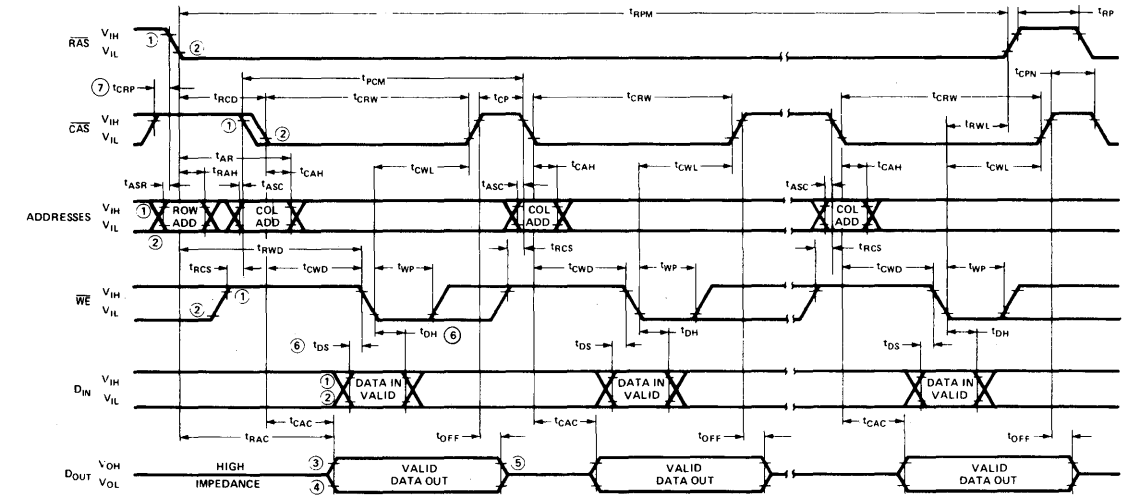


NOTES:
1,2. $V_{IH\ MIN}$ AND $V_{IL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF INPUT SIGNALS.
3,4. $V_{OH\ MIN}$ AND $V_{OL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF D_{OUT}.
5. t_{OFF} IS MEASURED TO $|I_{OUT}|$ = $|I_{LO}|$.
6. t_{RCH} IS REFERENCED TO THE TRAILING EDGE OF $\overline{CAS}$ OR $\overline{RAS}$, WHICHEVER OCCURS FIRST.
7. ALL VOLTAGES REFERENCED TO V_{SS}.
8. AC CHARACTERISTIC ASSUME t_T = 5ns.
9. SEE THE TYPICAL CHARACTERISTICS SECTION FOR VALUES OF THIS PARAMETER UNDER ALTERNATE CONDITIONS.
10. t_{CRP} REQUIREMENT IS ONLY APPLICABLE FOR $\overline{RAS}$/$\overline{CAS}$ CYCLES PRECEEDED BY A $\overline{CAS}$-ONLY CYCLE (i.e., FOR SYSTEMS WHERE $\overline{CAS}$ HAS NOT BEEN DECODED WITH $\overline{RAS}$).
11. ALL PREVIOUSLY SPECIFIED A.C. AND D.C. CHARACTERISTICS ARE APPLICABLE TO THEIR RESPECTIVE PAGE MODE DEVICE (i.e., 2118-3, S6329 WILL OPERATE AS A 2118-3).

PAGE MODE WRITE CYCLE

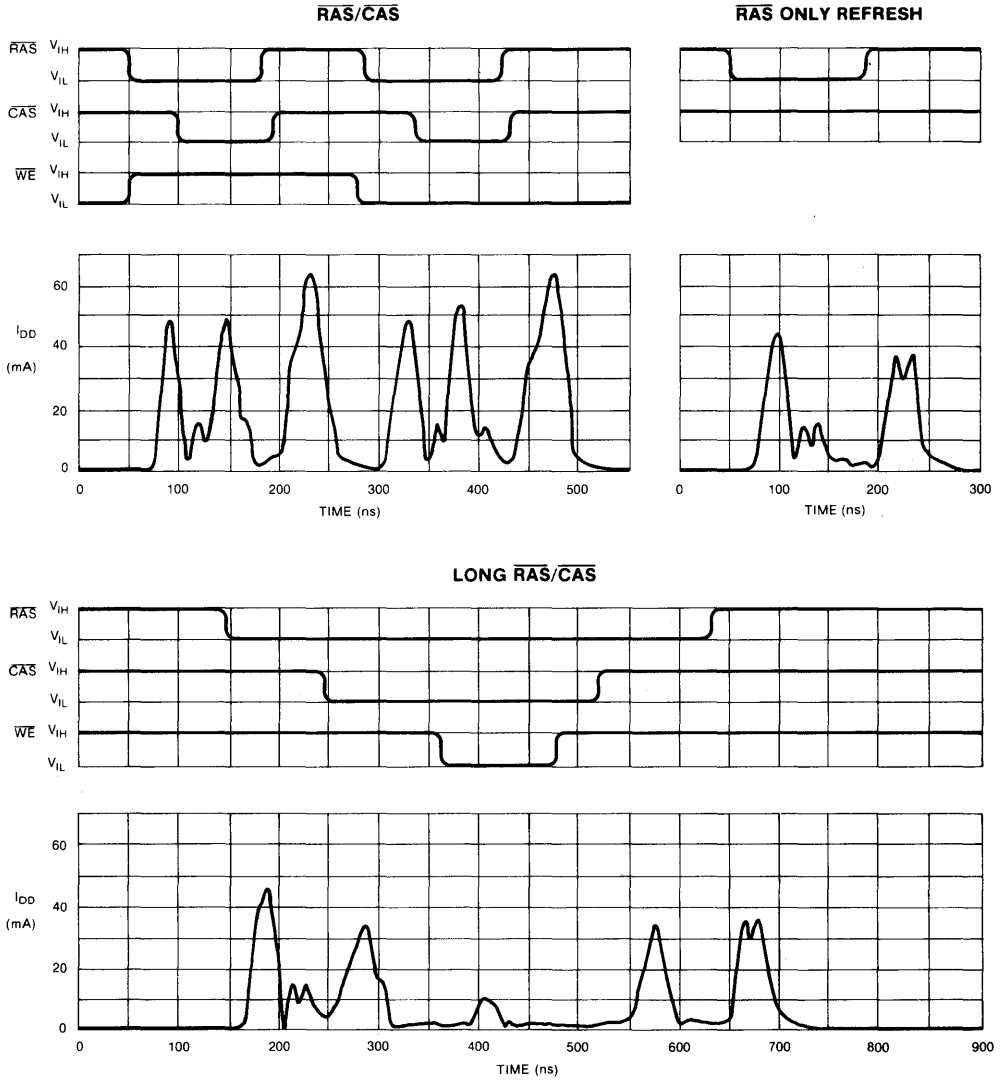


PAGE MODE READ-MODIFY-WRITE CYCLE



NOTES: 1,2. $V_{IH\ MIN}$ AND $V_{IL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF INPUT SIGNALS.
3,4. $V_{OH\ MIN}$ AND $V_{OL\ MAX}$ ARE REFERENCE LEVELS FOR MEASURING TIMING OF D_{OUT}.
5. t_{OFF} IS MEASURED TO $|I_{OUT}| = |I_{LO}|$.
6. t_{DS} AND t_{DH} ARE REFERENCED TO $\overline{CAS}$ OR $\overline{WE}$, WHICHEVER OCCURS LAST.
7. t_{RCH} IS REFERENCED TO THE TRAILING EDGE OF $\overline{CAS}$ OR $\overline{RAS}$, WHICHEVER OCCURS FIRST.
8. t_{CRP} REQUIREMENT IS ONLY APPLICABLE FOR $\overline{RAS}$/$\overline{CAS}$ CYCLES PRECEEDED BY A $\overline{CAS}$-ONLY CYCLE (i.e., FOR SYSTEMS WHERE $\overline{CAS}$ HAS NOT BEEN DECODED WITH $\overline{RAS}$).

TYPICAL SUPPLY CURRENT WAVEFORMS

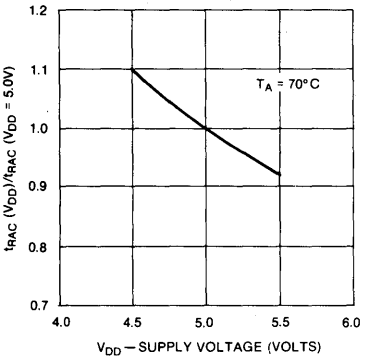


Typical power supply waveforms vs. time are shown for the $\overline{RAS}/\overline{CAS}$ timings of Read/Write, Read/Write (Long $\overline{RAS}/\overline{CAS}$), and $\overline{RAS}$-only refresh cycles. I_{DD} current transients at the $\overline{RAS}$ and $\overline{CAS}$ edges require adequate decoupling of these supplies.

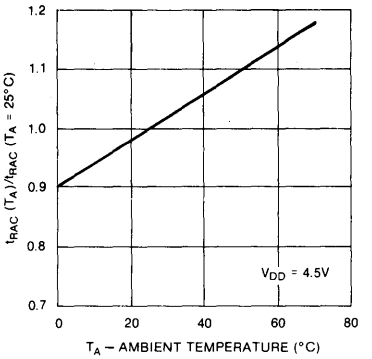
The effects of cycle time, V_{DD} supply voltage and ambient temperature on the I_{DD} current are shown in graphs included in the Typical Characteristics Section. Each family of curves for I_{DD1}, I_{DD2}, and I_{DD3} is related by a common point at $V_{DD} = 5.0V$ and $T_A = 25°C$ for two given t_{RAS} pulse widths. The typical I_{DD} current for a given condition of cycle time, V_{DD} and T_A can be determined by combining the effects of the appropriate family of curves.

TYPICAL CHARACTERISTICS

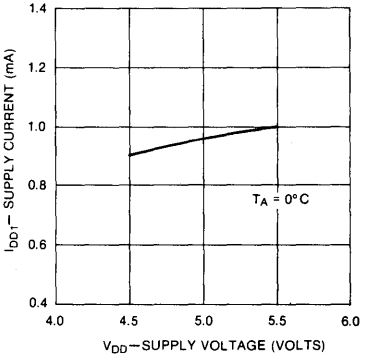
GRAPH 1
TYPICAL ACCESS TIME
t_{RAC} (NORMALIZED) VS. V_{DD}



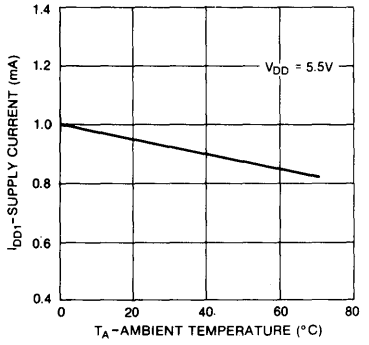
GRAPH 2
TYPICAL ACCESS TIME
t_{RAC} (NORMALIZED) VS.
AMBIENT TEMPERATURE



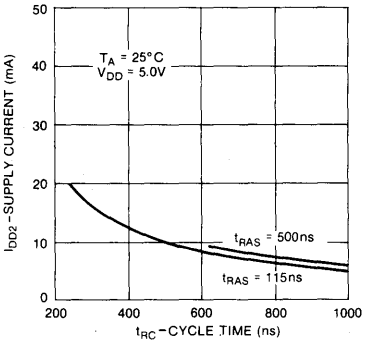
GRAPH 3
TYPICAL STANDBY CURRENT
I_{DD1} VS. V_{DD}



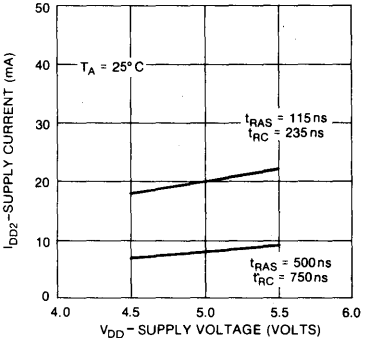
GRAPH 4
TYPICAL STANDBY CURRENT
I_{DD1} VS. AMBIENT TEMPERATURE



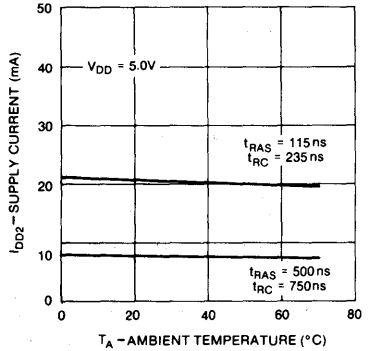
GRAPH 5
TYPICAL OPERATING CURRENT
I_{DD2} VS. t_{RC}



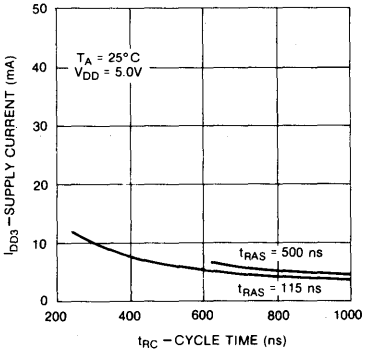
GRAPH 6
TYPICAL OPERATING CURRENT
I_{DD2} VS. V_{DD}



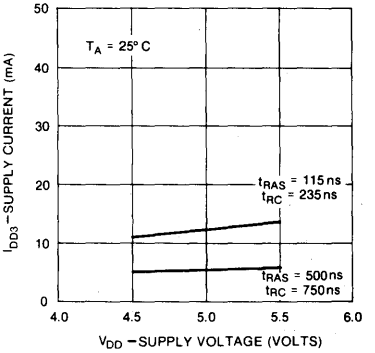
GRAPH 7
TYPICAL OPERATING CURRENT
I_{DD2} VS. AMBIENT TEMPERATURE



GRAPH 8
TYPICAL $\overline{RAS}$ ONLY
REFRESH CURRENT
I_{DD3} VS. t_{RC}

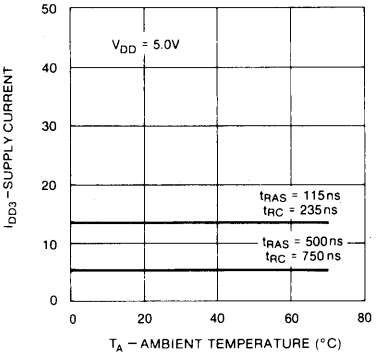


GRAPH 9
TYPICAL $\overline{RAS}$ ONLY
REFRESH CYCLE
I_{DD3} VS. V_{DD}

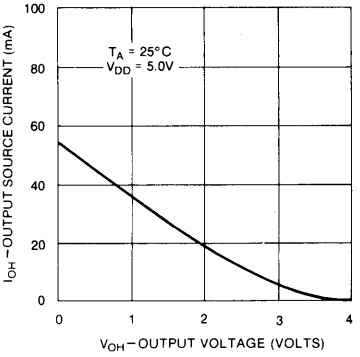


TYPICAL CHARACTERISTICS

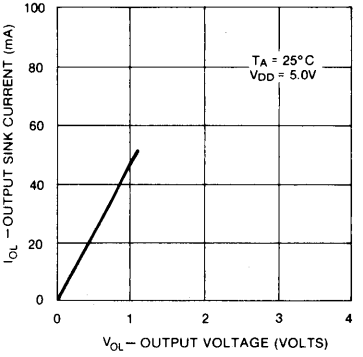
GRAPH 10
TYPICAL $\overline{RAS}$ ONLY
REFRESH CURRENT
I_{DD3} VS. AMBIENT TEMPERATURE



GRAPH 11
TYPICAL OUTPUT SOURCE CURRENT
I_{OH} VS. OUTPUT VOLTAGE V_{OH}



GRAPH 12
TYPICAL OUTPUT SINK CURRENT
I_{OL} VS. OUTPUT VOLTAGE V_{OL}



DEVICE DESCRIPTION

The Intel® 2118 is produced with HMOS, a high performance MOS technology which incorporates on chip substrate bias generation. This process, combined with new circuit design concepts, allows the 2118 to operate from a single +5V power supply, eliminating the +12V and -5V requirements. Pins 1 and 9 are not connected, which allows P.C.B. layout for future higher density memory generations.

The 2118 is functionally compatible with the industry standard 16-pin 16K dynamic RAMs, except for the power supply requirements. Replacing the +12V supply with a +5V supply and eliminating the -5V bias altogether, allows simple upgrade both in power and performance. To achieve total speed performance upgrade, however, the timing ciruitry must be modified to accommodate the higher performance.

READ CYCLE

A Read cycle is performed by maintaining Write Enable ($\overline{WE}$) high during a $\overline{RAS}/\overline{CAS}$ operation. The output pin of a selected device will remain in a high impedance state until valid data appears at the output at access time.

Device access time, t_{ACC}, is the longer of the two calculated intervals:

1. $t_{ACC} = t_{RAC}$ OR 2. $t_{ACC} = t_{RCD} + t_{CAC}$

Access time from $\overline{RAS}$, t_{RAC}, and access time from $\overline{CAS}$, t_{CAC}, are device parameters. Row to column address strobe delay time, t_{RCD}, are system dependent timing parameters. For example, substituting the device parameters of the 2118-3 yields:

3. $t_{ACC} = t_{RAC}$ = 100nsec for 25nsec $\leq t_{RCD} \leq$ 45nsec

OR

4. $t_{ACC} = t_{RCD} + t_{CAC} = t_{RCD}$ + 55nsec for $t_{RCD} >$ 45nsec

Note that if 25nsec $\leq t_{RCD} \leq$ 45nsec device access time is determined by equation 3 and is equal to t_{RAC}. If $t_{RCD} >$ 45nsec access time is determined by equation 4. This 20nsec interval (shown in the t_{RCD} inequality in equation 3) in which the falling edge of $\overline{CAS}$ can occur without affecting access time is provided to allow for system timing skew in the generation of $\overline{CAS}$.

REFRESH CYCLES

Each of the 128 rows of the 2118 must be refreshed every 2 milliseconds to maintain data. Any memory cycle:

1. Read Cycle
2. Write Cycle (Early Write, Delayed Write or Read-Modify-Write)
3. $\overline{RAS}$-only Cycle

refreshes the selected row as defined by the low order ($\overline{RAS}$) addresses. Any Write cycle, of course, may change the state of the selected cell. Using a Read, Write, or Read-Modify-Write cycle for refresh is not recommended for systems which utilize "wire-OR" outputs since output bus contention will occur.

A $\overline{RAS}$-only refresh cycle is the recommended technique for most applications to provide for data retention. A $\overline{RAS}$-only refresh cycle maintains the D_{OUT} in the high

impedance state with a typical power reduction of 30% over a Read or Write cycle.

$\overline{RAS}/\overline{CAS}$ TIMING

$\overline{RAS}$ and $\overline{CAS}$ have minimum pulse widths as defined by t_{RAS} and t_{CAS} respectively. These minimum pulse widths must be maintained for proper device operation and data integrity. A cycle, once begun by bringing $\overline{RAS}$ and/or $\overline{CAS}$ low must not be ended or aborted prior to fulfilling the minimum clock signal pulse width(s). A new cycle can not begin until the minimum precharge time, t_{RP}, has been met.

DATA OUTPUT OPERATION

The 2118 Data Output (D_{OUT}), which has three-state capability, is controlled by $\overline{CAS}$. During $\overline{CAS}$ high state ($\overline{CAS}$ at V_{IH}) the output is in the high impedance state. The following table summarizes the D_{OUT} state for various types of cycles.

Intel 2118 Data Output Operation for Various Types of Cycles

Type of Cycle	D_{OUT} State
Read Cycle	Data From Addressed Memory Cell
Early Write Cycle	HI-Z
$\overline{RAS}$-Only Refresh Cycle	HI-Z
$\overline{CAS}$-Only Cycle	HI-Z
Read/Modify/Write Cycle	Data From Addressed Memory Cell
Delayed Write Cycle	Indeterminate

HIDDEN REFRESH

An optional feature of the 2118 is that refresh cycles may be performed while maintaining valid data at the output pin. This feature is referred to as Hidden Refresh. Hidden Refresh is performed by holding $\overline{CAS}$ at V_{IL} and taking $\overline{RAS}$ high and after a specified precharge period (t_{RP}), executing a "$\overline{RAS}$-Only" refresh cycle, but with $\overline{CAS}$ held low (see Figure 1.)

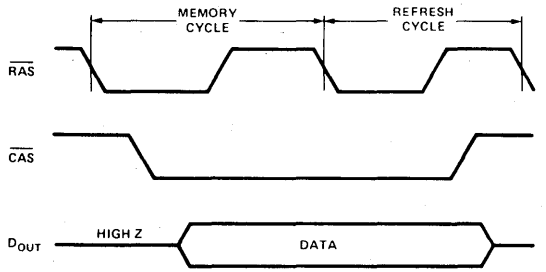


Figure 1. Hidden Refresh Cycle.

This feature allows a refresh cycle to be "hidden" among data cycles without affecting the data availability.

POWER ON

After the application of the V_{DD} supply, or after extended periods of bias (greater than 2ms) without clocks, the device must perform a minimum of eight (8) initialization cycles (any combination of cycles containing a $\overline{RAS}$ clock such as $\overline{RAS}$-only refresh) prior to normal operation.

The V_{DD} current (I_{DD}) requirement of the 2118 during power on is, however, dependent upon the input levels of $\overline{RAS}$ and $\overline{CAS}$. If the input levels of these clocks are at V_{IH} or V_{DD}, whichever is lower, the I_{DD} requirement per device is I_{DD1} (I_{DD} standby). If the input levels for these clocks are lower than V_{IH} or V_{DD} the I_{DD} requirement will be greater than I_{DD1}, as shown in Figure 2.

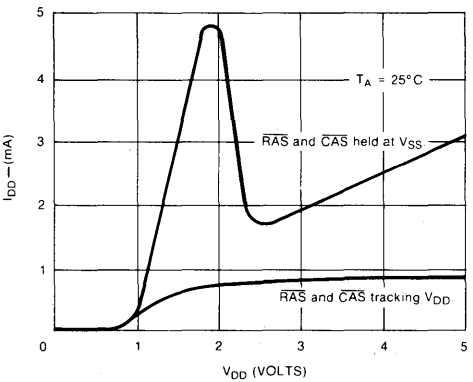


Figure 2. Typical I_{DD} VS V_{DD} during power up.

For large systems, this current requirement for I_{DD} could be substantially more than that for which the system has been designed. A system which has been designed, assuming the majority of devices to be operating in the refresh/standby mode, may produce sufficient I_{DD} loading such that the power supply may current limit. To assure that the system will not experience such loading during power on, a pullup resistor for each clock input to V_{DD} to maintain the non-selected current level (I_{DD1}) for the power supply is recommended.

2147H
HIGH SPEED 4096 x 1 BIT STATIC RAM

	2147H-1	2147H-2	2147H-3	2147HL-3	2147H	2147HL
Max. Access Time (ns)	35	45	55	55	70	70
Max. Active Current (mA)	180	180	180	125	160	140
Max. Standby Current (mA)	30	30	30	15	20	10

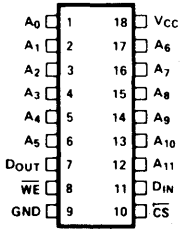
- **Pinout, Function, and Power Compatible to Industry Standard 2147**
- **HMOS II Technology**
- **Completely Static Memory—No Clock or Timing Strobe Required**
- **Equal Access and Cycle Times**
- **Single +5V Supply**
- **0.8-2.0V Output Timing Reference Levels**
- **Direct Performance Upgrade for 2147**
- **Automatic Power-Down**
- **High Density 18-Pin Package**
- **Directly TTL Compatible—All Inputs and Output**
- **Separate Data Input and Output**
- **Three-State Output**

The Intel® 2147H is a 4096-bit static Random Access Memory organized as 4096 words by 1-bit using HMOS-II, Intel's next generation high-performance MOS technology. It uses a uniquely innovative design approach which provides the ease-of-use features associated with non-clocked static memories and the reduced standby power dissipation associated with clocked static memories. To the user this means low standby power dissipation without the need for clocks, address setup and hold times, nor reduced data rates due to cycle times that are longer than access times.

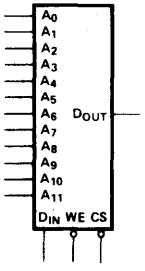
$\overline{CS}$ controls the power-down feature. In less than a cycle time after $\overline{CS}$ goes high—deselecting the 2147H—the part automatically reduces its power requirements and remains in this low power standby mode as long as $\overline{CS}$ remains high. This device feature results in system power savings as great as 85% in larger systems, where the majority of devices are deselected.

The 2147H is placed in an 18-pin package configured with the industry standard 2147 pinout. It is directly TTL compatible in all respects: inputs, output, and a single +5V supply. The data is read out nondestructively and has the same polarity as the input data. A data input and a separate three-state output are used.

PIN CONFIGURATION



LOGIC SYMBOL



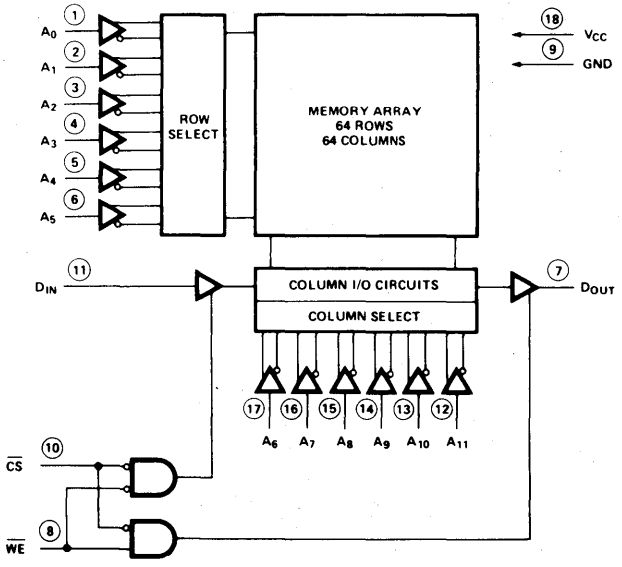
PIN NAMES

A_0–A_{11}	ADDRESS INPUTS	V_{CC}	POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND	GROUND
$\overline{CS}$	CHIP SELECT		
D_{IN}	DATA INPUT		
D_{OUT}	DATA OUTPUT		

TRUTH TABLE

$\overline{CS}$	$\overline{WE}$	MODE	OUTPUT	POWER
H	X	NOT SELECTED	HIGH Z	STANDBY
L	L	WRITE	HIGH Z	ACTIVE
L	H	READ	D_{OUT}	ACTIVE

BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to 85°C
Storage Temperature ..... −65°C to +150°C
Voltage on Any Pin
With Respect to Ground ..... −3.5V to +7V
Power Dissipation 1.2W
D.C. Output Current ..... 20 mA

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. AND OPERATING CHARACTERISTICS[1]

($T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 10\%$, unless otherwise noted.)

Symbol	Parameter	2147H-1, 2, 3 Min.	Typ.	Max.	2147HL-3 Min.	Typ.	Max.	Unit	2147H Min.	Typ.[2]	Max.	2147HL Min.	Typ.[2]	Max.
I_{LI}	Input Load Current (All Input Pins)		0.01	10		0.01	10	μA		0.01	10		0.01	10
I_{LO}	Outut Leakage Current		0.1	50		0.1	50	μA		0.1	50		0.1	50
I_{CC}	Operating Current		120	170			115	mA		100	150		100	135
				180			125	mA			160			140
I_{SB}	Standby Current		18	30		6	15	mA		12	20		7	10
I_{PO}[3]	Peak Power-On Current		35	70		25	50	mA		25	50		15	30
V_{IL}	Input Low Voltage	−3.0		0.8	−3.0		0.8	V	−3.0		0.8	−3.0		0.8
V_{IH}	Input High Voltage	2.0		6.0	2.0		6.0	V	2.0		6.0	2.0		6.0
V_{OL}	Output Low Voltage			0.4			0.4	V			0.4			0.4
V_{OH}	Output High Voltage	2.4			2.4			V	2.4			2.4		
I_{OS}	Output Short Circuit Current	−150		+150	−150		+150	mA	−150		+150	−150		+150

NOTES:
1. The operating ambient temperature range is guaranteed with transverse air flow exceeding 400 linear feet per minute.
2. Typical limits are at $V_{CC} = 5V$, $T_A = +25°C$, and specified loading.
3. A pull-up resistor to V_{CC} on the $\overline{CS}$ input is required to keep the device deselected; otherwise, power-on current approaches I_{CC} active.

A.C. TEST CONDITIONS

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	5 ns
Input Timing Reference Levels	1.5V
Output Timing Reference Level (2147H-1)	1.5V
Output Timing Reference Levels (2147H, H-2, H-3, HL, HL-3)	0.8–2.0V
Output Load	See Figure 1

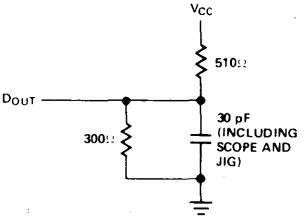


Figure 1. Output Load

CAPACITANCE[4] ($T_A = 25°C$, $f = 1.0$ MHz)

Symbol	Parameter	Max.	Unit	Conditions
C_{IN}	Input Capacitance	5	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	6	pF	$V_{OUT} = 0V$

NOTE:
4. This parameter is sampled and not 100% tested.

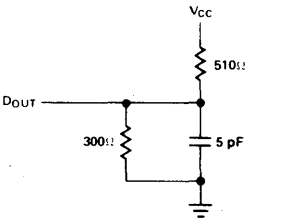


Figure 2. Output Load for t_{HZ}, t_{LZ}, t_{WZ}, t_{OW}

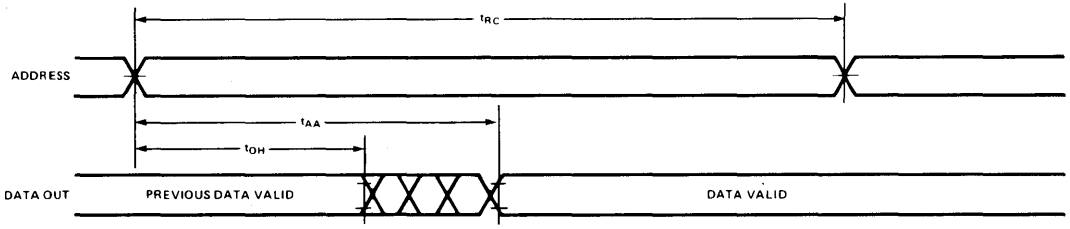
A.C. CHARACTERISTICS ($T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 10\%$, unless otherwise noted.)

Read Cycle

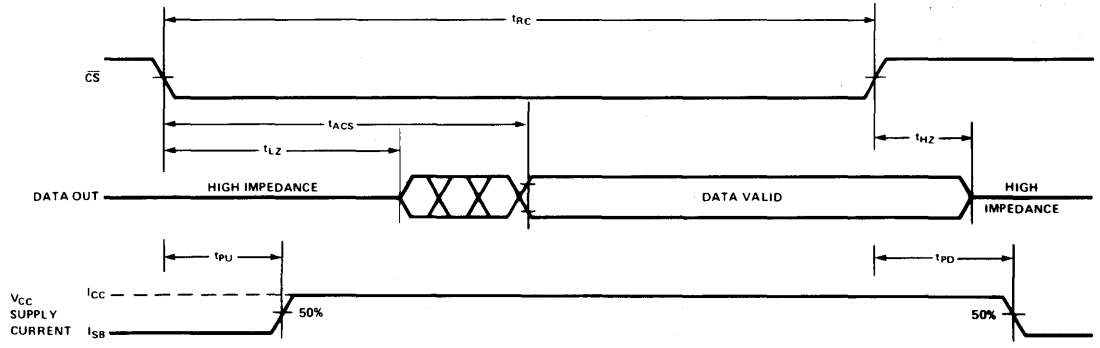
Symbol	Parameter	2147H-1 Min.	2147H-1 Max.	2147H-2 Min.	2147H-2 Max.	2147H-3, HL-3 Min.	2147H-3, HL-3 Max.	2147H, 2147HL Min.	2147H, 2147HL Max.	Unit
t_{RC}[1]	Read Cycle Time	35		45		55		70		ns
t_{AA}	Address Access Time		35		45		55		70	ns
t_{ACS1}[8]	Chip Select Access Time		35		45		55		70	ns
t_{ACS2}[9]	Chip Select Access Time		35		45		65		80	ns
t_{OH}	Output Hold from Address Change	5		5		5		5		ns
t_{LZ}[2,3,7]	Chip Selection to Output in Low Z	5		5		10		10		ns
t_{HZ}[2,3,7]	Chip Deselection to Output in High Z	0	30	0	30	0	30	0	40	ns
t_{PU}	Chip Selection to Power Up Time	0		0		0		0		ns
t_{PD}	Chip Deselection to Power Down Time		20		20		20		30	ns

WAVEFORMS

Read Cycle No. 1[4,5]



Read Cycle No. 2[4,6]



NOTES:

1. All Read Cycle timings are referenced from the last valid address to the first transitioning address.
2. At any given temperature and voltage condition, t_{HZ} max. is less than t_{LZ} min. both for a given device and from device to device.
3. Transition is measured ±500 mV from steady state voltage with specified loading in Figure 2.
4. $\overline{WE}$ is high for Read Cycles.
5. Device is continuously selected, $\overline{CS} = V_{IL}$.
6. Addresses valid prior to or coincident with $\overline{CS}$ transition low.
7. This parameter is sampled and not 100% tested.
8. Chip deselected for greater than 55 ns prior to selection.
9. Chip deselected for a finite time that is less than 55 ns prior to selection. If the deselect time is 0 ns, the chip is by definition selected and access occurs according to Read Cycle No. 1. Applies to 2147H, 2147HL, 2147H-3, and 2147HL-3.

A.C. CHARACTERISTICS (Continued)

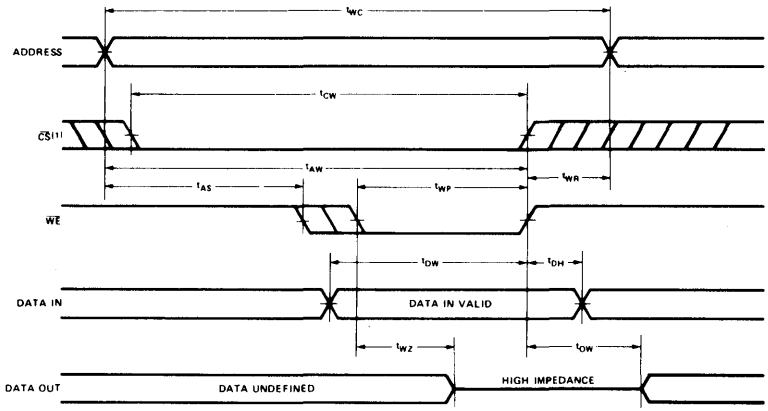
Write Cycle

Symbol	Parameter	2147H-1 Min.	2147H-1 Max.	2147H-2 Min.	2147H-2 Max.	2147H-3, HL-3 Min.	2147H-3, HL-3 Max.	2147H, 2147HL Min.	2147H, 2147HL Max.	Unit
t_{WC}[2]	Write Cycle Time	35		45		55		70		ns
t_{CW}	Chip Selection to End of Write	35		45		45		55		ns
t_{AW}	Address Valid to End of Write	35		45		45		55		ns
t_{AS}	Address Setup Time	0		0		0		0		ns
t_{WP}	Write Pulse Width	20		25		25		40		ns
t_{WR}	Write Recovery Time	0		0		10		15		ns
t_{DW}	Data Valid to End of Write	20		25		25		30		ns
t_{DH}	Data Hold Time	10		10		10		10		ns
t_{WZ}[3]	Write Enabled to Output in High Z	0	20	0	25	0	25	0	35	ns
t_{OW}[3]	Output Active from End of Write	0		0		0		0		ns

WAVEFORMS

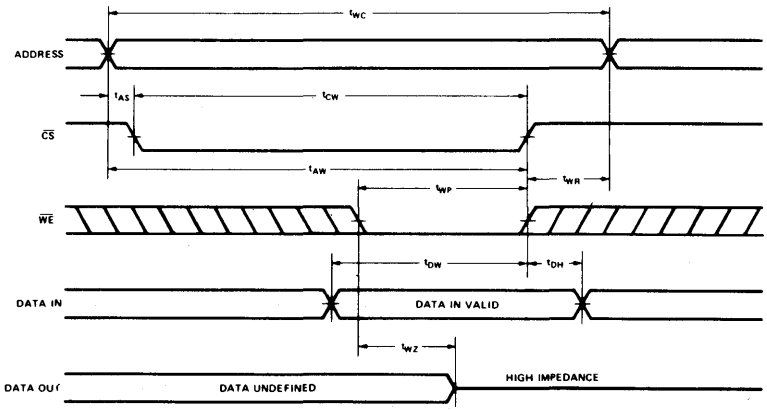
Write Cycle No. 1

($\overline{WE}$ CONTROLLED)[4]



Write Cycle No. 2

($\overline{CS}$ CONTROLLED)[4]



NOTES:

1. If $\overline{CS}$ goes high simultaneously with $\overline{WE}$ high, the output remains in a high impedance state.
2. All Write Cycle timings are referenced from the last valid address to the first transitioning address.
3. Transition is measured ±500 mV from steady state voltage with specified loading in Figure 2.
4. $\overline{CS}$ or $\overline{WE}$ must be high during address transitions.

2716
16K (2K x 8) UV ERASABLE PROM

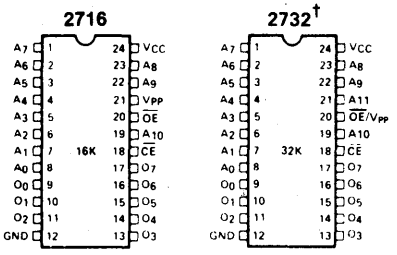
- **Fast Access Time**
  - **— 350 ns Max. 2716-1**
  - **— 390 ns Max. 2716-2**
  - **— 450 ns Max. 2716**
  - **— 490 ns Max. 2716-5**
  - **— 650 ns Max. 2716-6**
- **Single +5V Power Supply**
- **Low Power Dissipation**
  - **— 525 mW Max. Active Power**
  - **— 132 mW Max. Standby Power**
- **Pin Compatible to Intel® 2732 EPROM**
- **Simple Programming Requirements**
  - **— Single Location Programming**
  - **— Programs with One 50 ms Pulse**
- **Inputs and Outputs TTL Compatible during Read and Program**
- **Completely Static**

The Intel® 2716 is a 16,384-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2716 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical.

The 2716, with its single 5-volt supply and with an access time up to 350 ns, is ideal for use with the newer high performance +5V microprocessors such as Intel's 8085 and 8086. A selected 2716-5 and 2716-6 is available for slower speed applications. The 2716 is also the first EPROM with a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW while the maximum standby power dissipation is only 132 mW, a 75% savings.

The 2716 has the simplest and fastest method yet devised for programming EPROMs – single pulse TTL level programming. No need for high voltage pulsing because all programming controls are handled by TTL signals. Program any location at any time—either individually, sequentially or at random, with the 2716's single address location programming. Total programming time for all 16,384 bits is only 100 seconds.

PIN CONFIGURATION



†Refer to 2732 data sheet for specifications

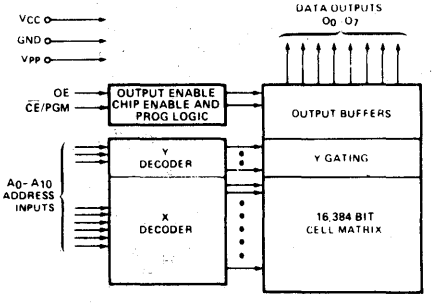
PIN NAMES

A_0–A_{10}	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS

MODE SELECTION

MODE \ PINS	$\overline{CE}$/PGM (18)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

BLOCK DIAGRAM



PROGRAMMING

The programming specifications are described in the Data Catalog PROM/ROM Programming Instructions Section.

Absolute Maximum Ratings*

Temperature Under Bias . . . . . -10°C to +80°C
Storage Temperature . . . . . -65°C to +125°C
All Input or Output Voltages with
Respect to Ground . . . . . +6V to -0.3V
V_{PP} Supply Voltage with Respect
to Ground During Program . . . . . +26.5V to -0.3V

**COMMENT:* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC and AC Operating Conditions During Read

	2716	2716-1	2716-2	2716-5	2716-6
Temperature Range	0°C – 70°C	0°C – 70°C	0°C – 70°C	0°C – 70°C	0°C – 70°C
V_{CC} Power Supply[1,2]	5V ±5%	5V ±10%	5V ±5%	5V ±5%	5V ±5%
V_{PP} Power Supply[2]	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}

READ OPERATION

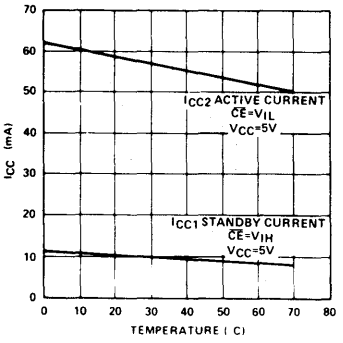
D.C. and Operating Characteristics

Symbol	Parameter	Limits			Unit	Conditions
		Min.	Typ.[3]	Max.		
I_{LI}	Input Load Current			10	μA	V_{IN} = 5.25V
I_{LO}	Output Leakage Current			10	μA	V_{OUT} = 5.25V
I_{PP1}[2]	V_{PP} Current			5	mA	V_{PP} = 5.25V
I_{CC1}[2]	V_{CC} Current (Standby)		10	25	mA	$\overline{CE}$ = V_{IH}, $\overline{OE}$ = V_{IL}
I_{CC2}[2]	V_{CC} Current (Active)		57	100	mA	$\overline{OE}$ = $\overline{CE}$ = V_{IL}
V_{IL}	Input Low Voltage	-0.1		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -400 μA

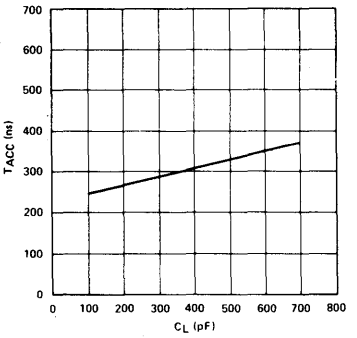
NOTES: 1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP1}.
3. Typical values are for T_A = 25°C and nominal supply voltages.
4. This parameter is only sampled and is not 100% tested.

Typical Characteristics

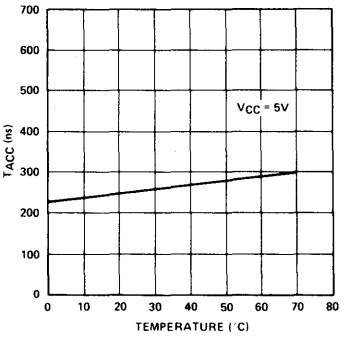
I_{CC} CURRENT vs. TEMPERATURE



ACCESS TIME vs. CAPACITANCE



ACCESS TIME vs. TEMPERATURE



A.C. Characteristics

Symbol	Parameter	Limits (ns)										Test Conditions
		2716		2716-1		2716-2		2716-5		2716-6		
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t_{ACC}	Address to Output Delay		450		350		390		450		450	$\overline{CE} = \overline{OE} = V_{IL}$
t_{CE}	$\overline{CE}$ to Output Delay		450		350		390		490		650	$\overline{OE} = V_{IL}$
t_{OE}	Output Enable to Output Delay		120		120		120		160		200	$\overline{CE} = V_{IL}$
t_{DF}	Output Enable High to Output Float	0	100	0	100	0	100	0	100	0	100	$\overline{CE} = V_{IL}$
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	0		0		0		0		0		$\overline{CE} = \overline{OE} = V_{IL}$

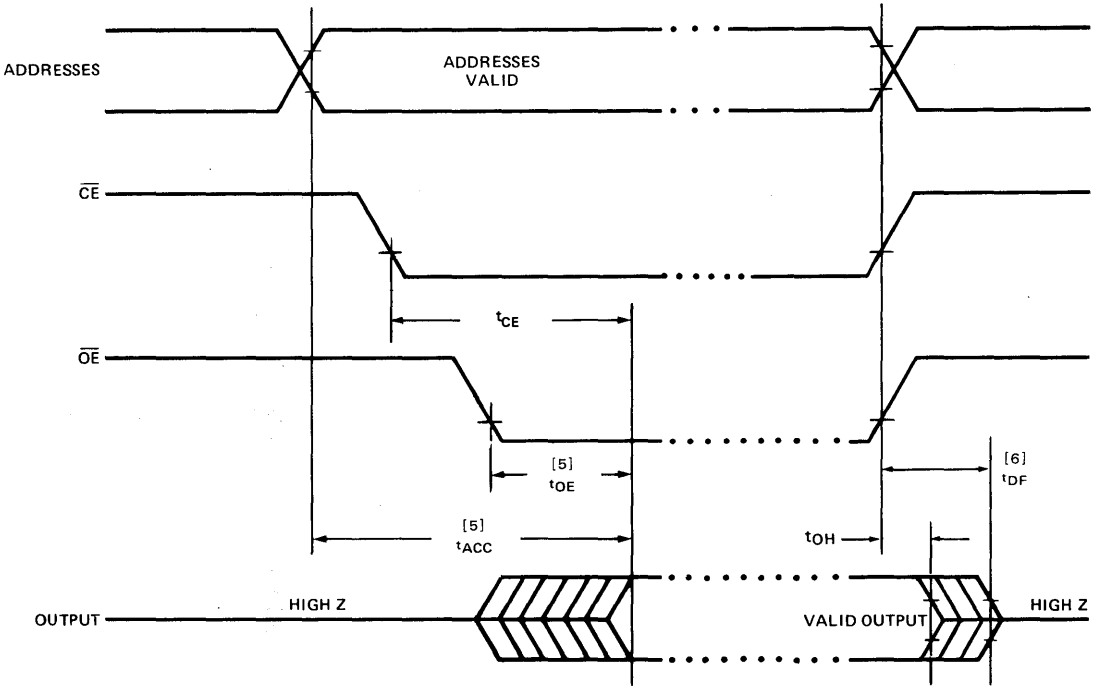
Capacitance [4] $T_A = 25°C$, f = 1 MHz

Symbol	Parameter	Typ.	Max.	Unit	Conditions
C_{IN}	Input Capacitance	4	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	8	12	pF	$V_{OUT} = 0V$

A.C. Test Conditions:

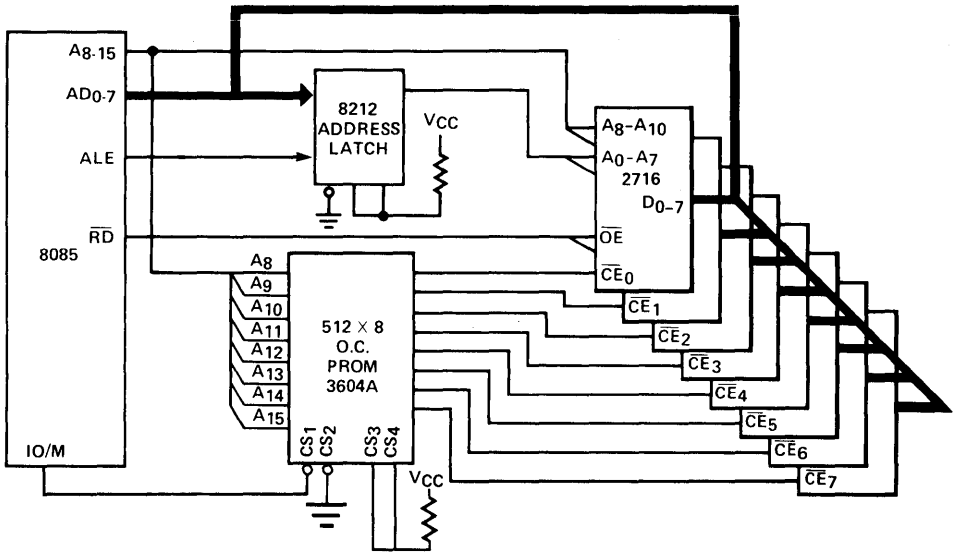
Output Load: 1 TTL gate and C_L = 100 pF
Input Rise and Fall Times: ≤20 ns
Input Pulse Levels: 0.8V to 2.2V
Timing Measurement Reference Level:
Inputs 1V and 2V
Outputs 0.8V and 2V

A. C. Waveforms [1]



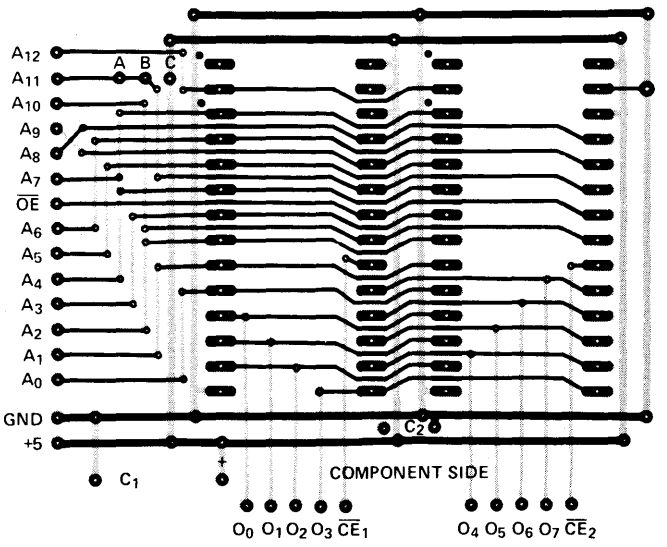
NOTE:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP1}.
3. Typical values are for $T_A = 25°C$ and nominal supply voltages.
4. This parameter is only sampled and is not 100% tested.
5. OE may be delayed up to $t_{ACC} - t_{OE}$ after the falling edge of CE without impact on t_{ACC}.
6. t_{DF} is specified from OE or CE, whichever occurs first.

TYPICAL 16K EPROM SYSTEM



- This scheme accomplished by using $\overline{CE}$ (PD) as the primary decode. $\overline{OE}$ (CS) is now controlled by previously unused signal. RD now controls data on and off the bus by way of $\overline{OE}$.
- A selected 2716 is available for systems which require $\overline{CE}$ access of less than 450 ns for decode network operation.
- The use of a PROM as a decoder allows for:
  a) Compatibility with upward (and downward) memory expansion.
  b) Easy assignment of ROM memory modules, compatible with PL/M modular software concepts.

8K, 16K, 32K, 64K 5V EPROM/ROM FAMILY PRINTED CIRCUIT BOARD LAYOUT



ERASURE CHARACTERISTICS

The erasure characteristics of the 2716 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 2716 in approximately 3 years, while it would take approximatley 1 week to cause erasure when exposed to direct sunlight. If the 2716 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 2716 window to prevent unintentional erasure.

The recommended erasure procedure (see Data Catalog PROM/ROM Programming Instruction Section) for the 2716 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15 W-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 μW/cm^2 power rating. The 2716 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

DEVICE OPERATION

The five modes of operation of the 2716 are listed in Table I. It should be noted that all inputs for the five modes are at TTL levels. The power supplies required are a +5V V_{CC} and a V_{PP}. The V_{PP} power supply must be at 25V during the three programming modes, and must be at 5V in the other two modes.

TABLE I. MODE SELECTION

MODE \ PINS	$\overline{CE}$/PGM (18)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

READ MODE

The 2716 has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data to the output pins, independent of device selection. Assuming that addresses are stable, address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs 120 ns (t_{OE}) after the falling edge of $\overline{OE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least $t_{ACC} - t_{OE}$.

STANDBY MODE

The 2716 has a standby mode which reduces the active power dissipation by 75%, from 525 mW to 132 mW. The 2716 is placed in the standby mode by applying a TTL high signal to the $\overline{CE}$ input. When in standby mode, the outputs are in a high impedence state, independent of the $\overline{OE}$ input.

OUTPUT OR-TIEING

Because 2716's are usually used in larger memory arrays, Intel has provided a 2 line control function that accomodates this use of multiple memory connections. The two line control function allows for:

a) the lowest possible memory power dissipation, and
b) complete assurance that output bus contention will not occur.

To most efficiently use these two control lines, it is recommended that $\overline{CE}$ (pin 18) be decoded and used as the primary device selecting function, while $\overline{OE}$ (pin 20) be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are only active when data is desired from a particular memory device.

PROGRAMMING

Initially, and after each erasure, all bits of the 2716 are in the "1" state. Data is introduced by selectively programming "0's" into the desired bit locations. Although only "0's" will be programmed, both "1's" and "0's" can be presented in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The 2716 is in the programming mode when the V_{PP} power supply is at 25V and $\overline{OE}$ is at V_{IH}. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

When the address and data are stable, a 50 msec, active high, TTL program pulse is applied to the $\overline{CE}$/PGM input. A program pulse must be applied at each address location to be programmed. You can program any location at any time — either individually, sequentially, or at random. The program pulse has a maximum width of 55 msec. The 2716 must not be programmed with a DC signal applied to the $\overline{CE}$/PGM input.

Programming of multiple 2716s in parallel with the same data can be easily accomplished due to the simplicity of the programming requirements. Like inputs of the paralleled 2716s may be connected together when they are programmed with the same data. A high level TTL pulse applied to the $\overline{CE}$/PGM input programs the paralleled 2716s.

PROGRAM INHIBIT

Programming of multiple 2716s in parallel with different data is also easily accomplished. Except for $\overline{CE}$/PGM, all like inputs (including $\overline{OE}$) of the parallel 2716s may be common. A TTL level program pulse applied to a 2716's $\overline{CE}$/PGM input with V_{PP} at 25V will program that 2716. A low level $\overline{CE}$/PGM input inhibits the other 2716 from being programmed.

PROGRAM VERIFY

A verify should be performed on the programmed bits to determine that they were correctly programmed. The verify may be performed wth V_{PP} at 25V. Except during programming and program verify, V_{PP} must be at 5V.

PRELIMINARY

2732
32K (4K x 8) UV ERASABLE PROM

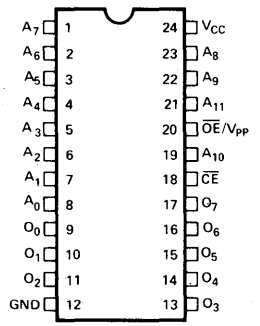
- **Fast Access Time:**
  - **— 450 ns Max. 2732**
  - **— 550 ns Max. 2732-6**
- **Single +5V ± 5% Power Supply**
- **Output Enable for MCS-85™ and MCS-86™ Compatibility**
- **Low Power Dissipation:**
  150mA Max. Active Current
  30mA Max. Standby Current
- **Pin Compatible to Intel® 2716 EPROM**
- **Completely Static**
- **Simple Programming Requirements**
  - **— Single Location Programming**
  - **— Programs with One 50ms Pulse**
- **Three-State Output for Direct Bus Interface**

The Intel® 2732 is a 32,768-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2732 operates from a single 5-volt power supply, has a standby mode, and features an output enable control. The total programming time for all bits is three and a half minutes. All these features make designing with the 2732 in microcomputer systems faster, easier, and more economical.

An important 2732 feature is the separate output control, Output Enable ($\overline{OE}$) from the Chip Enable control ($\overline{CE}$). The OE control eliminates bus contention in multiple bus microprocessor systems. Intel's Application Note AP-72 describes the microprocessor system implementation of the $\overline{OE}$ and $\overline{CE}$ controls on Intel's 2716 and 2732 EPROMs. AP-72 is available from Intel's Literature Department.

The 2732 has a standby mode which reduces the power dissipation without increasing access time. The maximum active current is 150mA, while the maximum standby current is only 30mA, an 80% savings. The standby mode is achieved by applying a TTL-high signal to the $\overline{CE}$ input.

PIN CONFIGURATION



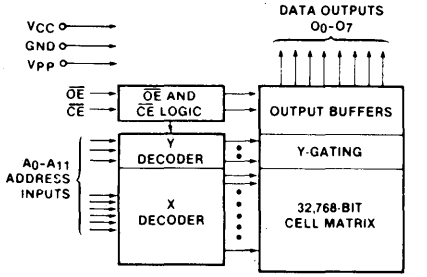
PIN NAMES

A_0-A_{11}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS

MODE SELECTION

MODE \ PINS	$\overline{CE}$ (18)	$\overline{OE}/V_{PP}$ (20)	V_{CC} (24)	OUTPUTS (9-11,13-17)
Read	V_{IL}	V_{IL}	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	High Z
Program	V_{IL}	V_{PP}	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+5	D_{OUT}
Program Inhibit	V_{IH}	V_{PP}	+5	High Z

BLOCK DIAGRAM



FEBRUARY 1980

PROGRAMMING

The programming specifications are described in the Data Catalog PROM/ROM Programming Instructions Section.

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... -10°C to +80°C
Storage Temperature ..... -65°C to +125°C
All Input or Output Voltages with
Respect to Ground ..... +6V to -0.3V

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 5%

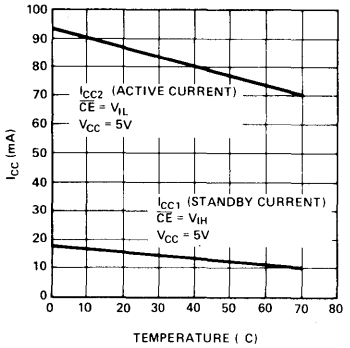
READ OPERATION

Symbol	Parameter	Limits Min.	Limits Typ.[1]	Limits Max.	Unit	Conditions
I_{LI1}	Input Load Current (except $\overline{OE}/V_{PP}$)			10	μA	V_{IN} = 5.25V
I_{LI2}	$\overline{OE}/V_{PP}$ Input Load Current			10	μA	V_{IN} = 5.25V
I_{LO}	Output Leakage Current			10	μA	V_{OUT} = 5.25V
I_{CC1}	V_{CC} Current (Standby)		15	30	mA	$\overline{CE}$ = V_{IH}, $\overline{OE}$ = V_{IL}
I_{CC2}	V_{CC} Current (Active)		85	150	mA	$\overline{OE}$ = $\overline{CE}$ = V_{IL}
V_{IL}	Input Low Voltage	-0.1		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 2.1mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -400μA

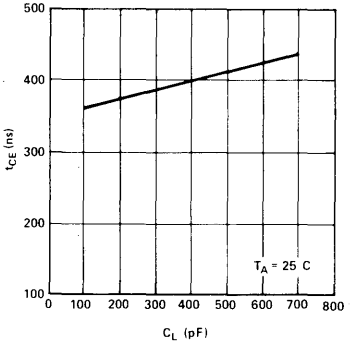
Note: 1. Typical values are for T_A = 25°C and nominal supply voltages.

TYPICAL CHARACTERISTICS

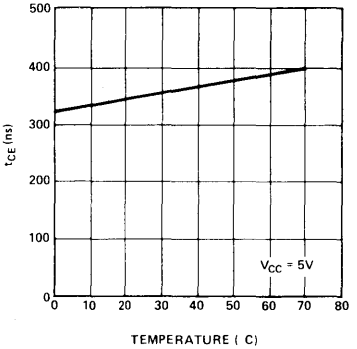
I_{CC} CURRENT VS. TEMPERATURE



$\overline{CE}$ TO OUTPUT DELAY (t_{CE}) VS. CAPACITANCE



$\overline{CE}$ TO OUTPUT DELAY (t_{CE}) VS. TEMPERATURE



A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 5%

Symbol	Parameter	2732 Limits		2732-6 Limits		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
t_{ACC}	Address to Output Delay		450		550	ns	$\overline{CE} = \overline{OE} = V_{IL}$
t_{CE}	$\overline{CE}$ to Output Delay		450		550	ns	$\overline{OE} = V_{IL}$
t_{OE}	Output Enable to Output Delay		120		120	ns	$\overline{CE} = V_{IL}$
t_{DF}	Output Enable High to Output Float	0	100	0	100	ns	$\overline{CE} = V_{IL}$
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$, Whichever Occurred First	0		0		ns	$\overline{CE} = \overline{OE} = V_{IL}$

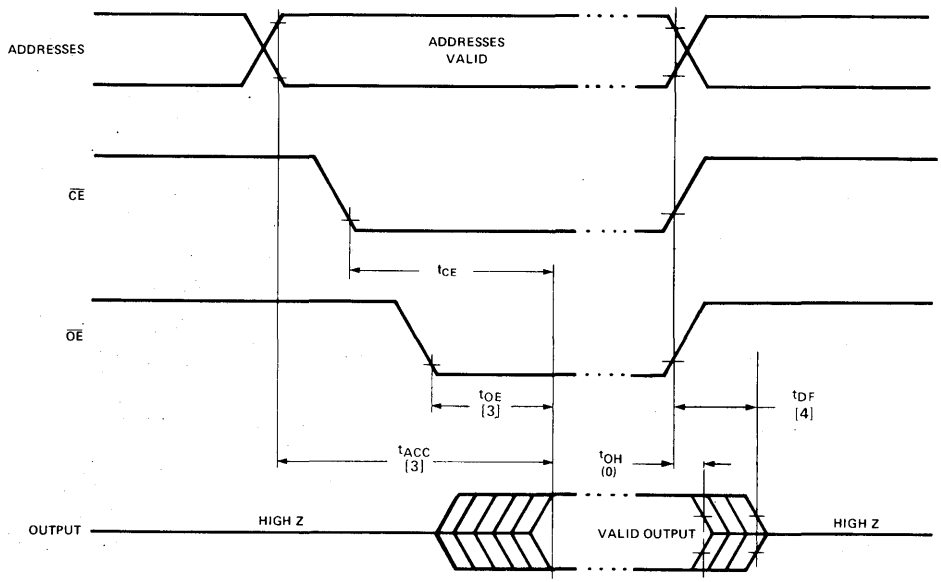
CAPACITANCE [1] T_A = 25°C, f = 1MHz

Symbol	Parameter	Typ.	Max.	Unit	Conditions
C_{IN1}	Input Capacitance Except $\overline{OE}/V_{PP}$	4	6	pF	V_{IN} = 0V
C_{IN2}	$\overline{OE}/V_{PP}$ Input Capacitance		20	pF	V_{IN} = 0V
C_{OUT}	Output Capacitance		12	pF	V_{OUT} = 0V

A.C. TEST CONDITIONS

Output Load: 1 TTL gate and C_L = 100pF
Input Rise and Fall Times: ≤ 20ns
Input Pulse Levels: 0.8V to 2.2V
Timing Measurement Reference Level:
Inputs 1V and 2V
Outputs 0.8V and 2V

A.C. WAVEFORMS [2]



NOTES:
1. THIS PARAMETER IS ONLY SAMPLED AND IS NOT 100% TESTED.
2. ALL TIMES SHOWN IN PARENTHESES ARE MINIMUM TIMES AND ARE NSEC UNLESS OTHERWISE SPECIFIED.
3. $\overline{OE}$ MAY BE DELAYED UP TO 330ns AFTER THE FALLING EDGE OF $\overline{CE}$ WITHOUT IMPACT ON t_{ACC}.
4. t_{DF} IS SPECIFIED FROM $\overline{OE}$ OR $\overline{CE}$, WHICHEVER OCCURS FIRST.

ERASURE CHARACTERISTICS

The erasure characteristics of the 2732 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 2732 in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 2732 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 2732 window to prevent unintentional erasure.

The recommended erasure procedure (see Data Catalog) for the 2732 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15 W-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 μW/cm^2 power rating. The 2732 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

DEVICE OPERATION

The five modes of operation of the 2732 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for $\overline{OE}/V_{PP}$ during programming. In the program mode the $\overline{OE}/V_{PP}$ input is pulsed from a TTL level to 25V.

TABLE 1. Mode Selection

MODE \ PINS	$\overline{CE}$ (18)	$\overline{OE}/V_{PP}$ (20)	V_{CC} (24)	OUTPUTS (9-11,13-17)
Read	V_{IL}	V_{IL}	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	High Z
Program	V_{IL}	V_{PP}	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+5	D_{OUT}
Program Inhibit	V_{IH}	V_{PP}	+5	High Z

Read Mode

The 2732 has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data to the output pins, independent of device selection. Assuming that addresses are stable, address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs 120ns (t_{OE}) after the falling edge of $\overline{OE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least $t_{ACC} - t_{OE}$.

Standby Mode

The 2732 has a standby mode which reduces the active power current by 80%, from 150mA to 30mA. The 2732 is placed in the standby mode by applying a TTL high signal to the $\overline{CE}$ input. When in standby mode, the outputs are in a high impedance state, independent of the $\overline{OE}$ input.

Output OR-Tieing

Because EPROMs are usually used in larger memory arrays, Intel has provided a 2 line control function that accommodates this use of multiple memory connections. The two line control function allows for:

a) the lowest possible memory power dissipation, and
b) complete assurance that output bus contention will not occur.

To most efficiently use these two control lines, it is recommended that $\overline{CE}$ (pin 18) be decoded and used as the primary device selecting function, while $\overline{OE}$ (pin 20) be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are only active when data is desired from a particular memory device.

Programming

Initially, and after each erasure, all bits of the 2732 are in the "1" state. Data is introduced by selectively programming "0's" into the desired bit locations. Although only "0's" will be programmed, both "1's" and "0's" can be presented in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The 2732 is in the programming mode when the $\overline{OE}/V_{PP}$ input is at 25V. It is required that a 0.1μF capacitor be placed across $\overline{OE}/V_{PP}$ and ground to suppress spurious voltage transients which may damage the device. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

When the address and data are stable, a 50msec, active low, TTL program pulse is applied to the $\overline{CE}$ input. A program pulse must be applied at each address location to be programmed. You can program any location at any time — either individually, sequentially, or at random. The program pulse has a maximum width of 55msec. The 2732 must not be programmed with a DC signal applied to the $\overline{CE}$ input.

Programming of multiple 2732s in parallel with the same data can be easily accomplished due to the simplicity of the programming requirements. Like inputs of the paralleled 2732s may be connected together when they are programmed with the same data. A low level TTL pulse applied to the $\overline{CE}$ input programs the paralleled 2732s.

Program Inhibit

Programming of multiple 2732s in parallel with different data is also easily accomplished. Except for $\overline{CE}$, all like inputs (including $\overline{OE}$) of the parallel 2732s may be common. A TTL level program pulse applied to a 2732's $\overline{CE}$ input with $\overline{OE}/V_{PP}$ at 25V will program that 2732. A high level $\overline{CE}$ input inhibits the other 2732s from being programmed.

Program Verify

A verify should be performed on the programmed bits to determine that they were correctly programmed. The verify is accomplished with $\overline{OE}/V_{PP}$ and $\overline{CE}$ at V_{IL}. Data should be verified t_{DV} after the falling edge of $\overline{CE}$.

2732A
32K (4K x 8) UV ERASABLE PROM

- **200ns (2732A-2) Maximum Access Time . . . HMOS*-E Technology**
- **Compatible to High Speed 8mHz 8086-2 MPU . . .Zero WAIT State**
- **Two Line Control**
- **Pin Compatible to 2764 EPROM**
- **Industry Standard Pinout . . . JEDEC Approved**
- **Low Standby Current . . . 35mA Max.**

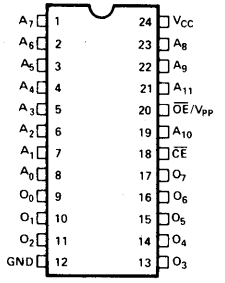
The Intel 2732A is a 5V only, 32,384 bit ultraviolet erasable and electrically programmable read-only memory (EPROM). It is pin compatible to Intel's 450ns 2732. The standard 2732A's access time is 250ns with speed selection (2732A-2) available at 200ns. The access time is compatible to high performance microprocessors, such as the 8mHz 8086-2. In these systems, the 2732A allows the microprocessor to operate without the addition of WAIT states.

An important 2732A feature is the separate output control, Output Enable ($\overline{OE}$), from the Chip Enable control ($\overline{CE}$). The $\overline{OE}$ control eliminates bus contention in multiple bus microprocessor systems. Intel's Application Note AP-72 describes the microprocessor system implementation of the $\overline{OE}$ and $\overline{CE}$ controls on Intel's EPROMs. AP-72 is available from Intel's Literature Department.

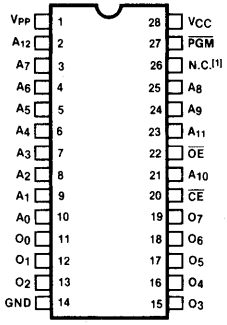
The 2732A has a standby mode which reduces the power dissipation without increasing access time. The maximum active current is 150mA, while the maximum standby current is only 35mA, a 75% saving. The standby mode is achieved by applying a TTL-high signal to the $\overline{CE}$ input.

The 2732A is fabricated with HMOS*-E technology, Intel's high speed N-channel MOS Silicon Gate Technology.

2732A PIN CONFIGURATION



2764 PIN CONFIGURATION



[1]For total compatibility from 2732A provide a trace to pin 26

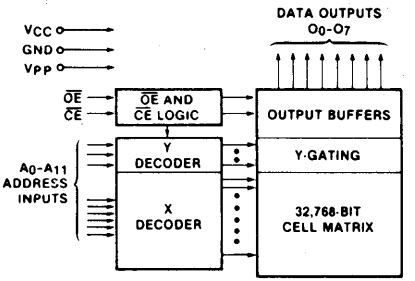
PIN NAMES

A_0-A_{11}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS

MODE SELECTION

MODE \ PINS	$\overline{CE}$ (18)	$\overline{OE}/V_{PP}$ (20)	V_{CC} (24)	OUTPUTS (9-11,13-17)
Read	V_{IL}	V_{IL}	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	High Z
Program	V_{IL}	V_{PP}	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+5	D_{OUT}
Program Inhibit	V_{IH}	V_{PP}	+5	High Z

BLOCK DIAGRAM



*HMOS is a patented process of Intel Corporation.

2758
8K (1K x 8) UV ERASABLE LOW POWER PROM

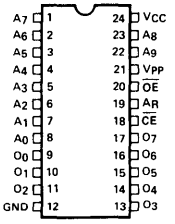
- **Single +5V Power Supply**
- **Simple Programming Requirements**
  - **Single Location Programming**
  - **Programs with One 50 ms Pulse**
- **Low Power Dissipation**
  525 mW Max. Active Power
  132 mW Max. Standby Power
- **Fast Access Time: 450 ns Max. in Active and Standby Power Modes**
- **Inputs and Outputs TTL Compatible during Read and Program**
- **Completely Static**
- **Three-State Outputs for OR-Ties**

The Intel® 2758 is a 8192-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2758 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical. The total programming time for all 8192 bits is 50 seconds.

The 2758 has a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525mW, while the maximum standby power dissipation is only 132mW, a 75% savings. Powerdown is achieved by applying a TTL-high signal to the $\overline{CE}$ input.

A 2758 system may be designed for total upwards compatibility with Intel's 16K 2716 EPROM (see Applications Note 72). The 2758 maintains the simplest and fastest method yet devised for programming EPROMs — single pulse TTL-level programming. There is no need for high voltage pulsing because all programming controls are handled by TTL signals. Program any location at any time — either individually, sequentially, or at random, with the single address location programming.

PIN CONFIGURATION



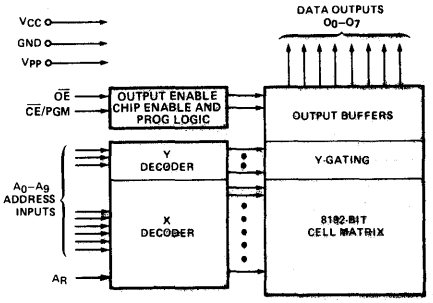
MODE SELECTION

MODE \ PINS	$\overline{CE}/PGM$ (18)	A_R (19)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	V_{IL}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IL}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IL}	V_{IH}	+25	+5	High Z

PIN NAMES

A_0–A_9	ADDRESSES
$\overline{CE}/PGM$	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS
A_R	SELECT REFERENCE INPUT LEVEL

BLOCK DIAGRAM



PROGRAMMING

The programming specifications are described in the Data Catalog PROM/ROM Programming Instructions section.

Absolute Maximum Ratings*

Temperature Under Bias . . . . . -10°C to +80°C
Storage Temperature . . . . . -65°C to +125°C
All Input or Output Voltages with
Respect to Ground . . . . . +6V to -0.3V
V_{PP} Supply Voltage with Respect
to Ground During Programming . . . . . +26.5V to -0.3V

**COMMENT:* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

READ OPERATION

D.C. and Operating Characteristics

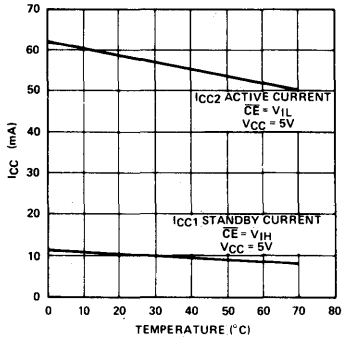
T_A = 0°C to 70°C, V_{CC}[1,2] = +5V ±5%, V_{PP}[2] = V_{CC}

Symbol	Parameter	Limits			Unit	Conditions
		Min.	Typ.[3]	Max.		
I_{LI}	Input Load Current			10	μA	V_{IN} = 5.25V
I_{LO}	Output Leakage Current			10	μA	V_{OUT} = 5.25V
I_{PP1}[2]	V_{PP} Current			5	mA	V_{PP} = 5.25V
I_{CC1}[2]	V_{CC} Current (Standby)		10	25	mA	$\overline{CE}$ = V_{IH}, $\overline{OE}$ = V_{IL}
I_{CC2}[2]	V_{CC} Current (Active)		57	100	mA	$\overline{OE}$ = $\overline{CE}$ = V_{IL}
A_R[4]	Select Reference Input Level	-0.1		0.8	V	I_{IN} = 10 μA
V_{IL}	Input Low Voltage	-0.1		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC} + 1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -400 μA

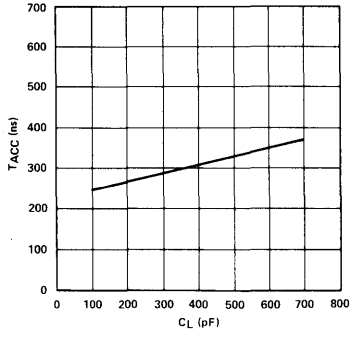
NOTES:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP1}.
3. Typical values are for T_A = 25°C and nominal supply voltages.
4. A_R is a reference voltage level which requires an input current of only 10 μA. The 2758 S1865 is also available which has a reference voltage level of V_{IH} instead of V_{IL}.

Typical Characteristics

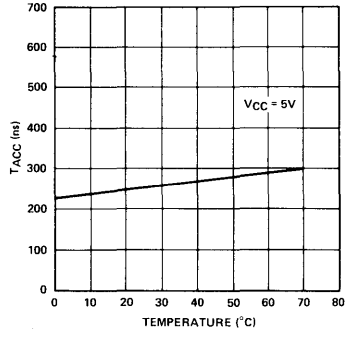
I_{CC} CURRENT vs. TEMPERATURE



ACCESS TIME vs. CAPACITANCE



ACCESS TIME vs. TEMPERATURE



A.C. Characteristics

$T_A = 0°C$ to $70°C$, V_{CC}[1] = +5V ±5%, V_{PP}[2] = V_{CC}

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.[3]	Max.		
t_{ACC}	Address to Output Delay		250	450	ns	$\overline{CE} = \overline{OE} = V_{IL}$
t_{CE}	$\overline{CE}$ to Output Delay		280	450	ns	$\overline{OE} = V_{IL}$
t_{OE}	Output Enable to Output Delay			120	ns	$\overline{CE} = V_{IL}$
t_{DF}	Output Enable High to Output Float	0		100	ns	$\overline{CE} = V_{IL}$
t_{OH}	Output Hold From Addresses, $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	0			ns	$\overline{CE} = \overline{OE} = V_{IL}$

Capacitance[4] $T_A = 25°C$, f = 1 MHz

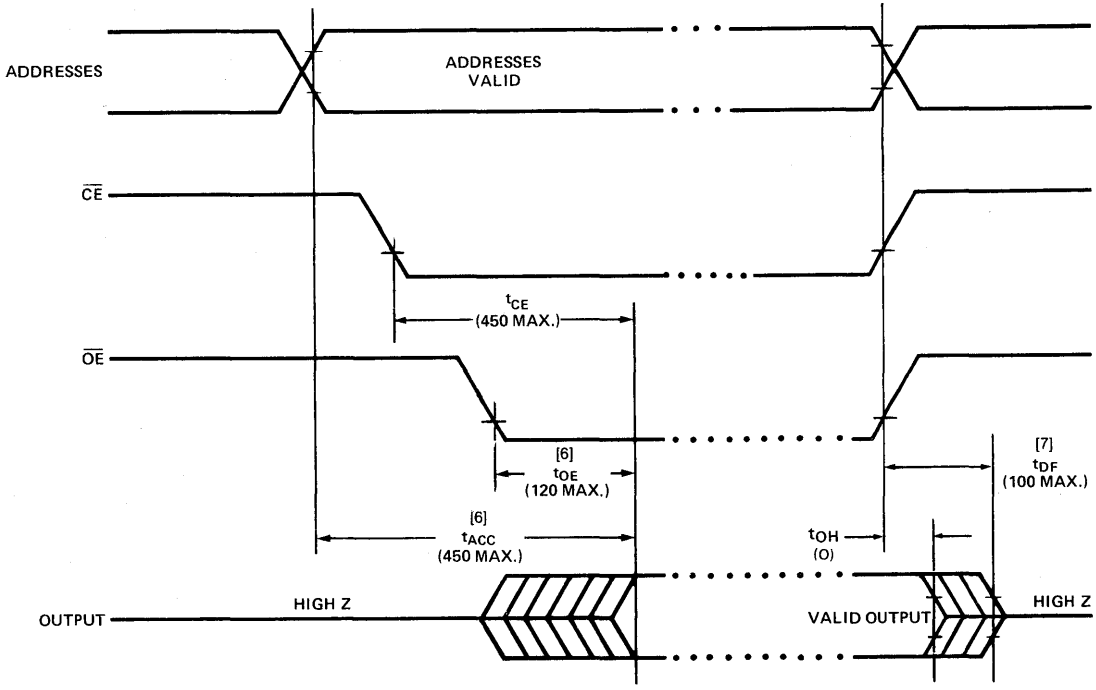
Symbol	Parameter	Typ.	Max.	Unit	Conditions
C_{IN}	Input Capacitance	4	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	8	12	pF	$V_{OUT} = 0V$

NOTE: Please refer to page 2 for notes.

A.C. Test Conditions:

Output Load: 1 TTL gate and C_L = 100 pF
Input Rise and Fall Times: ≤20 ns
Input Pulse Levels: 0.8V to 2.2V
Timing Measurement Reference Level:
Inputs 1V and 2V
Outputs 0.8V and 2V

A.C. Waveforms[5]



NOTES:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP1}.
3. Typical values are for $T_A = 25°C$ and nominal supply voltages.
4. This parameter is only sampled and is not 100% tested.
5. All times shown in parentheses are minimum times and are nsec unless otherwise specified.
6. $\overline{OE}$ may be delayed up to 330 ns after the falling edge of $\overline{CE}$ without impact on t_{ACC}.
7. t_{DF} is specified from $\overline{OE}$ or $\overline{CE}$, whichever occurs first.

ERASURE CHARACTERISTICS

The erasure characteristics of the 2758 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 2758 in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 2758 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 2758 window to prevent unintentional erasure.

The recommended erasure procedure (see Data Catalog Programming Section) for the 2758 is exposure to short-wave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated does (i.e., UV intensity X exposure time) for erasure should be a minimum of 15 W-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with 12,000 μW/cm^2 power rating. The 2758 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

DEVICE OPERATION

The five modes of operation of the 2758 are listed in Table 1. It should be noted that all inputs for the five modes are at TTL levels. The power supplied required are a +5V V_{CC} and a V_{PP}. The V_{PP} power supply must be at 25V during the two programming modes, and must be at 5V in the other three modes. In all operational modes, A_R must be at V_{IL} (except for the 2758 S1865 which has A_R at V_{IH}).

TABLE I. MODE SELECTION

PINS / MODE	$\overline{CE}$/PGM (18)	A_R (19)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	V_{IL}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IL}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IL}	V_{IH}	+25	+5	High Z

READ MODE

The 2758 has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data to the output pins, independent of device selection. Assuming that addresses are stable, address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs 120 ns (t_{OE}) after the falling edge of $\overline{OE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least $t_{ACC} - t_{OE}$.

STANDBY MODE

The 2758 has a standby mode which reduces the active power dissipation by 75%, from 525 mW to 132 mW. The 2758 is placed in the standby mode by applying a TTL high signal to $\overline{CE}$ input. When in standby mode, the outputs are in a high impedence state, independent of the OE input.

OUTPUT OR-TIEING

Because EPROMs are usually used in larger memory arrays, Intel has provided a 2 line control function that accommodates this use of multiple memory connections. The two line control function allows for:

a) the lowest possible memory Power dissipation, and
b) complete assurance that output bus contention will not occur.

To most efficiently use these two control lines, it is recommended that $\overline{CE}$ (pin 18) be decoded and used as the primary device selecting function, while $\overline{OE}$ (pin 20) be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are only active when data is desired from a particular memory device.

PROGRAMMING

Initially, and after each erasure, all bits of the 2758 are in the "1" state. Data is introduced by selectively programming "0's" into the desired bit locations. Although only "0's" will be programmed, both "1's" and "0's" can be presented in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The 2758 is in the programming mode when the V_{PP} power supply is at 25V and $\overline{OE}$ is at V_{IH}. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

When the address and data are stable, a 50 msec, active high, TTL program pulse is applied to the $\overline{CE}$/PGM input. A program pulse must be applied at each address location to be programmed. You can program any location at any time — either individually, sequentially, or at random. The program pulse has a maximum width of 55 msec.

The 2758 must be programmed with a DC signal applied to the $\overline{CE}$/PGM input.

Programming of multiple 2758s in parallel with the same data can be easily accomplished due to the simplicity of the programming requirements. Like inputs of the paralleled 2758s may be connected together when they are programmed with the same data. A high level TTL pulse applied to the $\overline{CE}$/PGM input programs the paralleled 2758s.

PROGRAM INHIBIT

Programming of multiple 2758s in parallel with different data is also easily accomplished. Except for $\overline{CE}$/PGM, all like inputs(including $\overline{OE}$) of the parallel 2758s may be common. A TTL level program pulse applied to a 2758's $\overline{CE}$/PGM input with V_{PP} at 25V will program that 2758. A low level $\overline{CE}$/PGM input inhibits the other 2758 from being programmed.

PROGRAM VERIFY

A verify should be performed on the programmed bits to determine that they were correctly programmed. The verify may be performed with V_{PP} at 25V. Except during programming and program verify, V_{PP} must be at 5V.

Development Support Tools 8

CHAPTER 8
DEVELOPMENT SUPPORT TOOLS

INTELLEC SERIES II/85 MODEL 235

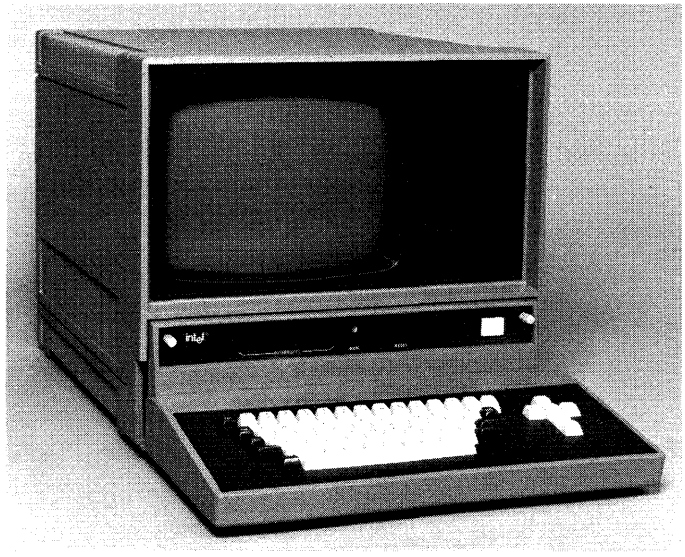
Complete 8051 application development support is based on the Intellec Model 235 development system. A Model 235 system is configured from a Model 225 system and Model 720 Flexible Disk Subsystem, described in the following pages. The Model 235 provides the ISIS-II operating system, supported by integral CRT and keyboard, 1.25M bytes of flexible disk storage and 64K bytes of RAM.

The Model 225 system without the additional disk drives can support limited 8051 development tasks which require only a single drive and 250K bytes of storage.

MODEL 225
INTELLEC® SERIES II/85
MICROCOMPUTER DEVELOPMENT SYSTEM

- **Complete microcomputer development system for MCS®-86, MCS®-85, MCS®-80, MCS®-48, and MCS®-51 microprocessor families**
- **High performance 8085A-2 CPU, 64K bytes RAM memory, and 4K bytes ROM memory**
- **Self-test diagnostic capability**
- **Built-in interfaces for high speed paper tape reader/punch, printer, and universal PROM programmer**
- **Integral 250K byte floppy disk drive with total storage capacity expandable to over 2M bytes of floppy disk storage and 7.3M bytes of hard disk storage**
- **Powerful ISIS-II Disk Operating System with relocating macroassembler, linker, locater, and CRT based editor CREDIT**
- **Supports PL/M, FORTRAN, BASIC, PASCAL and COBOL high level languages**
- **Software compatible with previous Intellec® systems**

The Intellec Series II/85 Model 225 Microcomputer Development System is a performance enhanced, complete microcomputer development system integrated into one compact package. The Model 225 includes a CPU with 64K bytes of RAM, 4K bytes of ROM, a 2000-character CRT, detachable full ASCII keyboard with cursor controls and upper/lower case capability, and a 250K-byte floppy disk drive. Powerful ISIS-II Disk Operating System software allows the Model 225 to be used quickly and efficiently for assembling and debugging programs for Intel's MCS-86, MCS-85, MCS-80, MCS-48, or MCS-51 microprocessor families. ISIS-II performs all file handling operations for the user, leaving him free to concentrate on the details of his own application. When used with an optional in-circuit emulator (ICE™) module, the Model 225 provides all of the hardware and software development tools necessary for the rapid development of a microcomputer-based product. Optional storage peripherals provide over 2 million bytes of floppy disk, and 7.3 million of hard disk storage capacity.



121599-001 Rev. A

FUNCTIONAL DESCRIPTION

Hardware Components

The Intellec Series II/85 Model 225 is a highly-integrated microcomputer development system consisting of a CRT chassis with a 6-slot cardcage, power supply, fans, cables, single floppy disk drive, and two printed circuit cards. A separate, full ASCII keyboard is connected with a cable. A block diagram of the Model 225 is shown in Figure 1.

CPU Cards — The master CPU card contains its own microprocessor, memory, I/O, interrupt and bus interface circuitry implemented with Intel's high technology LSI components. Known as the integrated processor card (IPC), it occupies the first slot in the cardcage. A second slave CPU card is responsible for all remaining I/O control including the CRT and keyboard interface. This card, mounted on the rear panel, also contains its own microprocessor, RAM and ROM memory, and I/O interface logic, thus, in effect, creating a dual processor environment. Known as the I/O controller (IOC), the slave CPU card communicates with the IPC over an 8-bit bidirectional data bus.

Expansion — Five remaining slots in the cardcage are available for system expansion. Additional expansion of 4 slots can be achieved through the addition of an Intellec Series II expansion chassis.

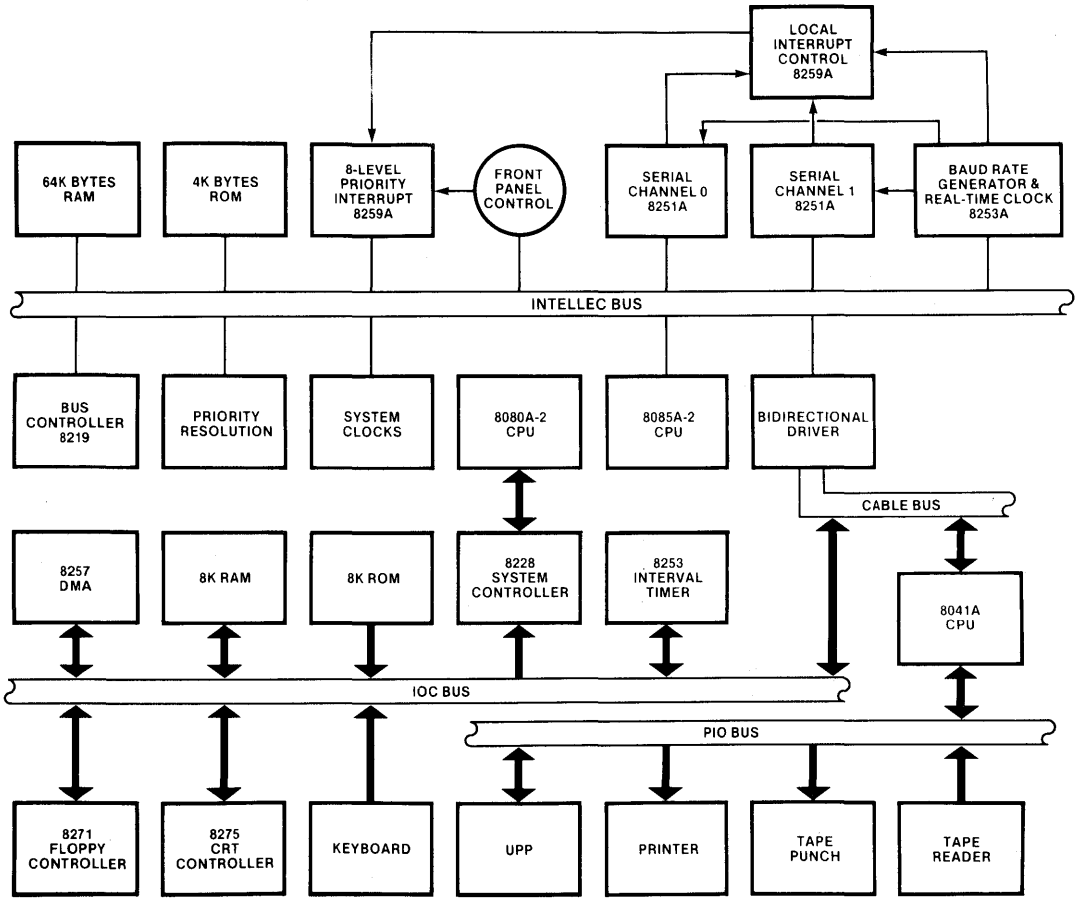


Figure 1. Intellec Series II/85 Model 225 Microcomputer Development System Block Diagram

System Components

The heart of the IPC is an Intel NMOS 8-bit microprocessor, the 8085A-2, running at 4.0 MHz. 64K bytes of RAM memory are provided on the board using 16K RAMs. 4K of ROM is provided, preprogrammed with system bootstrap "self-test" diagnostics and the Intellec Series II/85 System Monitor. The eight-level vectored priority interrupt system allows interrupts to be individually masked. Using Intel's versatile 8259A interrupt controller, the interrupt system may be user programmed to respond to individual needs.

Input/Output

IPC Serial Channels — The I/O subsystem in the Model 225 consists of two parts: the IOC card and two serial channels on the IPC itself. Each serial channel is RS232 compatible and is capable of running asynchronously from 110 to 9600 baud or synchronously from 150 to 56K baud. Both may be connected to a user defined data set or terminal. One channel contains current loop adapters. Both channels are implemented using Intel's 8251A USART. They can be programmed to perform a variety of I/O functions. Baud rate selection is accomplished through an Intel 8253 interval timer. The 8253 also serves as a real-time clock for the entire system. I/O activity through both serial channels is signaled to the system through a second 8259A interrupt controller, operating in a polled mode nested to the primary 8259A.

IOC Interface — The remainder of system I/O activity takes place in the IOC. The IOC provides interface for the CRT, keyboard, and standard Intellec peripherals including printer, high speed paper tape reader/punch, and universal PROM programmer. The IOC contains its own independent microprocessor, an 8080A-2. The CPU controls all I/O operations as well as supervising communications with the IPC. 8K bytes of ROM contain all I/O control firmware. 8K bytes of RAM are used for CRT screen refresh storage. These do not occupy space in Intellec Series II main memory since the IOC is a totally independent microcomputer subsystem.

Integral CRT

Display — The CRT is a 12-inch raster scan type monitor with a 50/60 Hz vertical scan rate and 15.5kHz horizontal scan rate. Controls are provided for brightness and contrast adjustments. The interface to the CRT is provided through an Intel 8275 single-chip programmable CRT controller. The master processor on the IPC transfers a character for display to the IOC, where it is stored in RAM. The CRT controller reads a line at a time into its line buffer through an Intel 8257 DMA controller and then feeds one character at a time to the character generator to produce the video signal. Timing for the CRT control is provided by an Intel 8253 interval timer. The screen display is formatted as 25 rows of 80 characters. The full set of ASCII characters is displayed, including lower case alphas.

Keyboard — The keyboard interfaces directly to the IOC processor via an 8-bit data bus. The keyboard contains an Intel UPI-41™ Universal Peripheral Interface, which scans the keyboard, encodes the characters, and buffers the characters to provide N-key rollover. The keyboard itself is a high quality typewriter style keyboard containing the full ASCII character set. An upper/lower case switch allows the system to be used for document preparation. Cursor control keys are also provided.

Peripheral Interface

A UPI-41 Universal Peripheral Interface on the IOC board provides interface for other standard Intellec peripherals including a printer, high speed paper tape reader, high speed paper tape punch, and universal PROM programmer. Communication between the IPC and IOC is maintained over a separate 8-bit bidirectional data bus. Connectors for the four devices named above, as well as the two serial channels, are mounted directly on the IOC itself.

Control

User control is maintained through a front panel, consisting of a power switch and indicator, reset/boot switch, run/halt light, and eight interrupt switches and indicators. The front panel circuit board is attached directly to the IPC, allowing the eight interrupt switches to connect to the primary 8259A, as well as to the Intellec Series II bus.

Integral Floppy Disk Drive

The integral floppy disk is controlled by an Intel 8271 single chip, programmable floppy disk controller. It transfers data via an Intel 8257 DMA controller between an IOC RAM buffer and the diskette. The 8271 handles reading and writing of data, formatting diskettes, and reading status, all upon appropriate commands from the IOC microprocessor.

MULTIBUS™ Interface Capability

All Intellec Series II/85 models implement the industry standard MULTIBUS protocol. The MULTIBUS protocol enables several bus masters, such as CPU and DMA devices, to share the bus and memory by operating at different priority levels. Resolution of bus exchanges is synchronized by a bus clock signal derived independently from processor clocks. Read/write transfers may take place at rates up to 5 MHz. The bus structure is suitable for use with any Intel microcomputer family.

SPECIFICATIONS

Host Processor (IPC)

8085A-2 based, operating at 4.0 MHz.
RAM — 64K on the CPU card
ROM — 4K (2K in monitor, 2K in boot/diagnostic)
Bus — MULTIBUS™ bus, maximum transfer rate of 5 MHz
Clocks — Host processor, crystal controlled at 4.0 MHz, bus clock, crystal controlled at 9.8304 MHz

I/O Interfaces

Two Serial I/O Channels, RS232C, at 110-9600 baud (asynchronous) or 150-56K baud (synchronous). Baud rates and serial format fully programmable using Intel 8251A USARTs. Serial Channel 1 additionally provided with 20 mA current loop. Parallel I/O interfaces provided for paper tape punch, paper tape reader, printer, and UPP-103 Universal PROM Programmer.

Interrupts

8-level, maskable, nested priority interrupt network initiated from front panel or user selected devices.

Direct Memory Access (DMA)

Standard capability on MULTIBUS interface; implemented for user selected DMA devices through optional DMA module—maximum transfer rate of 5 MHz.

Memory Access Time

RAM — 470 ns max
PROM — 540 ns max

Integral Floppy Disk Drive

Floppy Disk System Capacity —
250K bytes (formatted)
Floppy Disk System Transfer Rate —
160K bits/sec
Floppy Disk System Access Time —
Track to Track: 10 ms max
Average Random Positioning: 260 ms
Rotational Speed: 360 rpm
Average Rotational Latency: 83 ms
Recording Mode: FM

Physical Characteristics

CHASSIS

Width — 17.37 in. (44.12 cm)
Height — 15.81 in. (40.16 cm)
Depth — 19.13 in. (48.59 cm)
Weight — 73 lb. (33 kg)

KEYBOARD

Width — 17.37 in. (44.12 cm)
Height — 3.0 in. (7.62 cm)
Depth — 9.0 in. (22.86 cm)
Weight — 6 lb. (3 kg)

Electrical Characteristics

DC POWER SUPPLY

Volts Supplied	Amps Supplied	Typical System Requirements
+ 5 ± 5%	30.0	17.0
+12 ± 5%	2.5	1.1
−12 ± 5%	0.3	0.1
−10 ± 5%	1.0	0.08
+15 ± 5%*	1.5	1.5
+24 ± 5%*	1.7	1.7

*Not available on bus.

AC REQUIREMENTS FOR MAINFRAME

110V, 60 Hz — 5.9 Amp
220V, 50 Hz — 3.0 Amp

Environmental Characteristics

Operating Temperature — 16°C to 32°C (61°F to 90°F)
Humidity — 20% to 80%

Equipment Supplied

Model 225 Chassis including:
- Integrated Processor Card (IPC)
- I/O Controller Board (IOC)
- CRT
- ROM-Resident System Monitor

Detachable keyboard
ISIS-II System Diskette with MCS-80/MCS-85 Macroassembler
ISIS-II CREDIT Diskette CRT-Based Text Editor

Documentation Supplied

A Guide to Microcomputer Development Systems, 9800558

Intellec® Series II Model 22X/23X Installation Manual, 9800559

ISIS-II System User's Guide, 9800306

Intellec® Series II Hardware Reference Manual, 9800556

8080/8085 Assembly Language Programming Manual, 9800301

ISIS-II 8080/8085 Assembler Operator's Manual, 9800292

Intellec® Series II Systems Monitor Source Listing, 9800605

Intellec® Series II Schematic Drawings, 9800554

ISIS-II CREDIT (CRT-Based Text Editor) User's Guide, 9800902

Additional manuals may be ordered from any Intel sales representative or distributor office, or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

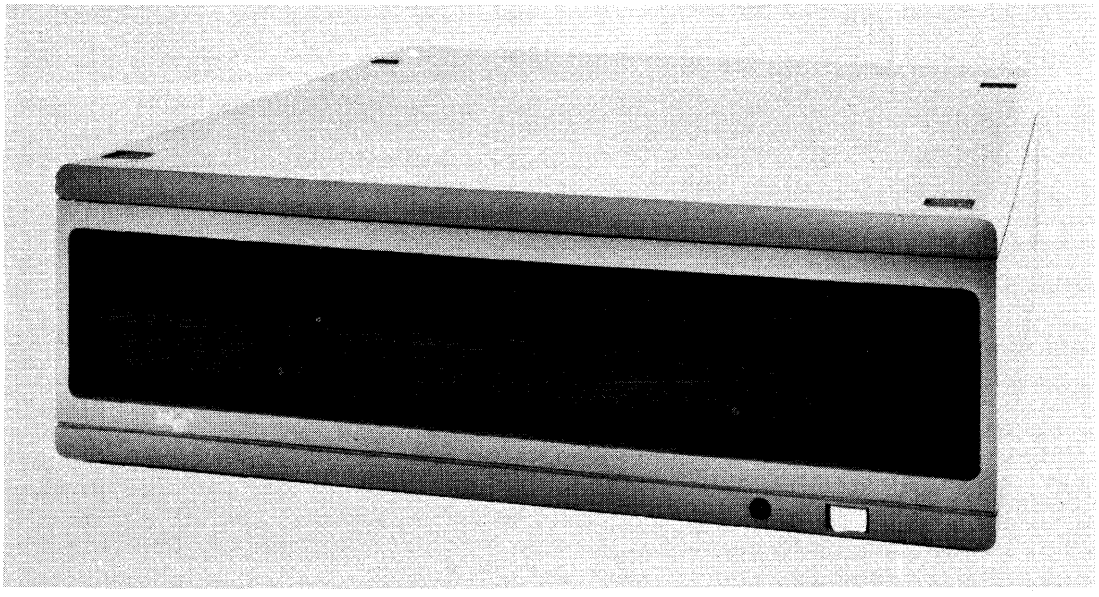
Part Number	Description
MDS-225*	Intellec® Series II/85 Model 225 Microcomputer Development System (110V/60 Hz)
MDS-226*	Intellec® Series II/85 Model 226 Microcomputer Development System (220V/50 Hz)

**"MDS"* is an ordering code only, and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corp.

INTELLEC® SINGLE/DOUBLE DENSITY FLEXIBLE DISK SYSTEM

- **Flexible Disk System Providing High Speed Input/Output and Data Storage for Intellec® Microcomputer Development Systems**
- **Available in Both Single Density and Double Density Systems**
- **Data Recorded on Single Density Flexible Disk Is in IBM Soft-Sectored Format Which Allows ¼ Million Byte Data Capacity with Up to 200 Files Per Flexible Disk**
- **Data Recorded on Double Density Flexible Disk is in Soft-Sectored Format Which Allows ½ Million Byte Data Capacity with Up to 200 Files Per Flexible Disk**
- **Associated Software Supports Up to Four Double Density Drives and Two Single Density Drives, Providing Up to 2.5 Megabytes of Storage in One System**
- **Dynamic Allocation and Deallocation of Flexible Disk Sectors for Variable Length Files**

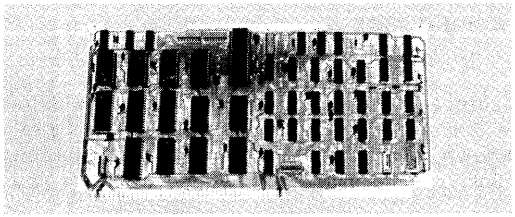
The Intellec Flexible Disk System is a sophisticated, general purpose, bulk storage peripheral for use with the Intellec Microcomputer Development System. The use of a flexible disk operating system significantly reduces program development time. The software system known as ISIS-11 (Intel System Implementation Supervisor), provides the ability to edit, assemble, compile, link, relocate, execute and debug programs, and performs all file management tasks for the user.



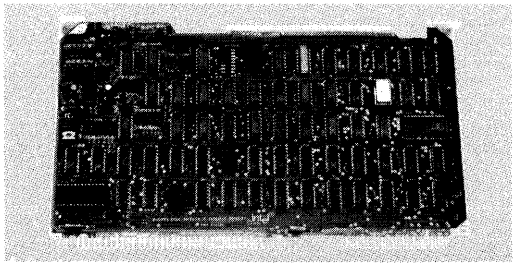
HARDWARE

The Intellec® flexible disk system provides direct access bulk storage, intelligent controller, and two flexible disk drives. Each single density drive provides ¼ million bytes of storage with a data transfer rate of 250,000 bits/second. The double density drive provides ½ million bytes of storage with a data transfer rate of 500,000 bits/second. The controllers are implemented with Intel's powerful Series 3000 Bipolar Microcomputer Set. The controllers provide interface to the Intellec System bus. Each single density controller will support two drives. Each double density controller will support up to four drives. The flexible disk system records all data in soft sector format.

The single/double density flexible disk controllers each consists of two boards, the Channel Board and the Interface Board. These two printed circuit boards reside in the Intellec System chassis. The boards are shown in the photograph, and are described in more detail in the following paragraphs.



SINGLE/DOUBLE DENSITY CHANNEL BOARD



DOUBLE DENSITY INTERFACE BOARD
(SINGLE DENSITY INTERFACE BOARD
IS SIMILAR TO THE ONE SHOWN ABOVE)

CHANNEL BOARD

The *Channel Board* is the primary control module within the flexible disk system. The Channel Board receives, decodes, and responds to channel commands from the Central Processor Unit (CPU) in the Intellec system. The Channel Board can access a block of Intellec system memory to determine the particular flexible disk operations to be performed and fetch the parameters required for the successful completion of the specified operation.

The control functions of the Channel Board have been achieved with an 8-bit microprogrammed processor, designed with Intel's Series 3000 Bipolar Microcomputer Set. This 8-bit processor includes four 3002 Central Processing Elements (2-bit slice per CPE), a 3001 Microprogram Control Unit, and 512 x 32 bits of 3604 programmable-read-only-memory (PROM) which stores the microprogram. It is the execution of the microprogram by the microcomputer set which actually effects the control capability of the Channel Board.

This board is the same for either single or double density drives, except that the Series 3000 microcode is different.

INTERFACE BOARD

The *Interface Board* provides the flexible disk controller with a means of communication with the flexible disk drives, as well as with the Intellec system bus. Under control of the microprogram being executed on the Channel Board, the Interface Board generates those signals which cause the read/write head on the selected drive to be loaded (i.e., to come in contact with the flexible disk platter), cause the head to move to the proper track and verify successful operation. The Interface Board accepts the data being read off the flexible disk, interprets synchronizing bit patterns, checks the validity of the data using a cyclic redundancy check (CRC) polynomial, and then transfers the data to the Channel Board.

During write operations, the Interface Board outputs the data and clock bits to the selected drive at the proper times, and generates the CRC characters which are then appended to the data.

When the flexible disk controller requires access to Intellec system memory, the Interface Board requests the DMA master control of the system bus, and generates the appropriate memory command. The Interface Board also acknowledges I/O commands as required by the Intellec bus.

The Flexible Disk System is capable of performing seven different operations: recalibrate, seek, format track, write data, write deleted data, read data, and verify CRC.

The channel board is different for single and double density drives, due to the different recording techniques used. The single density controller boards support one set of dual single density drives. The double density controller boards support up to two sets of dual double density drives (four drives total).

The double density controller may co-reside with the Intel single density controller to allow conversion of single density flexible disk to double density format, and provide up to 2.5M bytes of storage.

FLEXIBLE DISK DRIVE MODULES

Each flexible disk drive consists of read/write and control electronics, drive mechanisms, read/write head, track positioning mechanism, and the removable flexible disk platter. These components interact to perform the following functions:

- Interpret and generate control signals
- Move read/write head to selected track
- Read and write data

ASSOCIATED SOFTWARE — INTEL SYSTEMS IMPLEMENTATION SUPERVISOR (ISIS-II)

The Flexible Disk Drive System is to be used in conjunction with the ISIS-II Operating System. ISIS-II provides total file management capabilities, file editing, library management, run-time supports, and utility management.

ISIS-II provides automatic implementation of random access disk files. Up to 200 files may be stored on each ¼ million byte flexible disk for single density system or on each ½ million byte flexible disk for double density system. For more information, see the ISIS-II data specification sheet.

ISIS-II OPERATIONAL ENVIRONMENTAL

ISIS-II

32K bytes RAM memory
48K bytes when using Assembler Macro feature
64K bytes when using PL/M or Fortran
System Console
Single or Double density Flexible Disk Drive

HARDWARE SPECIFICATIONS

MEDIA

Single Density	Double Density
Flexible Disk	Double Density Specified Flexible Disk
One Recording Surface	One Recording Surface
IBM Soft Sector Format	Soft Sector Format
77 Tracks/Diskette	77 Tracks/Diskette
26 Sectors/Track	52 Sectors/Track
128 Bytes/Sector	128 Bytes/Sector

PHYSICAL CHARACTERISTICS

CHASSIS AND DRIVES

Mounting:	Table-Top
Height:	5.7 in. (14.5 cm)
Width:	17.6 in. (44.7 cm)
Depth:	19.4 in. (49.3 cm)
Weight:	43.0 lb. (19.5 kg)

ELECTRICAL CHARACTERISTICS

CHASSIS

DC Power Supplies
Supplied Internal to the Cabinet

AC Power Requirements
3-wire input with center conductor (earth ground) tied to chassis

Single-phase, 115 VAC; 60 Hz; 1.2 Amp Maximum (For a Typical Unit)
230 VAC; 50 Hz; 0.7 Amp Maximum (For a Typical Unit)

FLEXIBLE DISK OPERATING SYSTEM CONTROLLER

DC Power Requirements (All power supplied by Intellec Development System)

CHANNEL BOARD

Single Density	Double Density
5V @ 3.75A (typ), 5A (max)	5V @ 3.75A (typ), 5A (max)

INTERFACE BOARD

Single Density	Double Density
5V @ 1.5A (typ), 2.5A (max)	5V @ 1.5A (typ), 2.5A (max) – 10V @ 0.1A (typ), 0.2A (max)

FLEXIBLE DISK DRIVE PERFORMANCE SPECIFICATION

	Single Density	Double Density
Capacity (Unformatted):		
Per Disk	3.1 megabits	6.2 megabits
Per Track	41 kilobits	82 kilobits
Capacity (Formatted):		
Per Disk	2.05M bits	4.10 megabits
Per Track	26.6K bits	53.2 kilobits
Data Transfer Rate	250 kilobits/sec	500 kilobits/sec
Access Time:		
Track-to-Track	10 ms	10 ms
Head Settling Time	10 ms	10 ms
Average Random Positioning Time	260 ms	260 ms
Rotational Speed	360 rpm	360 rpm
Average Latency	83 ms	83 ms
Recording Mode	Frequency Modulation	M^2FM

ENVIRONMENTAL CHARACTERISTICS

MEDIA

Temperature:
Operating: 15.6°C to 51.7°C
Non-Operating: 5°C to 55°C
Humidity:
Operating: 8 to 80% (Wet bulb 29.4°C)
Non-Operating: 8 to 90%

DRIVES AND CHASSIS

Temperature:
Operating: 10°C to 38°C
Non-Operating: – 35°C to 65°C
Humidity:
Operating: 20% to 80% (Wet bulb 26.7°C)
Non-Operating: 5% to 95%

CONTROLLER BOARDS

Temperature:
Operating: 0 to 55°C
Non-Operating: – 55°C to 85°C
Humidity:
Operating: Up to 95% relative humidity without condensation
Non-Operating: All conditions without condensation of water or frost

EQUIPMENT SUPPLIED

SINGLE DENSITY

Cabinet, Power Supplies, Line Cord, Two Drives
Single Density FDC Channel Board
Single Density FDC Interface Board
Dual Auxiliary Board Connector
Flexible Disk Controller Cable
Flexible Disk Peripheral Cable
Hardware Reference Manual
Reference Schematics
ISIS-II Single Density System Disk
ISIS-II System User's Guide

DOUBLE DENSITY

Cabinet, Power Supplies, Line Cord, Two Drives
Double Density FDC Channel Board
Double Density FDC Interface Board
Dual Auxiliary Board Connector
Flexible Disk Controller Cable
Flexible Disk Peripheral Cable
Hardware Reference Manual
Reference Schematics
ISIS-II Double Density System Disk
ISIS-II System User's Guide

OPTIONAL EQUIPMENT

MDS-BLD*	10 Blank Flexible Disks
MDS-730/731*	Second Drive Cabinet with two additional drives

ORDERING INFORMATION

Part Number	Description
MDS-710/110V* 711/220V	Flexible Disk drive unit with two drives, single density drive controller, software, and cables.
MDS-720-110V* 721/220V	Flexible Disk drive unit with two drives, double density drive controller, software, and cables.
MDS-730/110V* 731/220V	Add-on drive unit with two drives and double density cable, without controller and software. Can be used with double density controller.

*MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

8051 SOFTWARE DEVELOPMENT PACKAGE

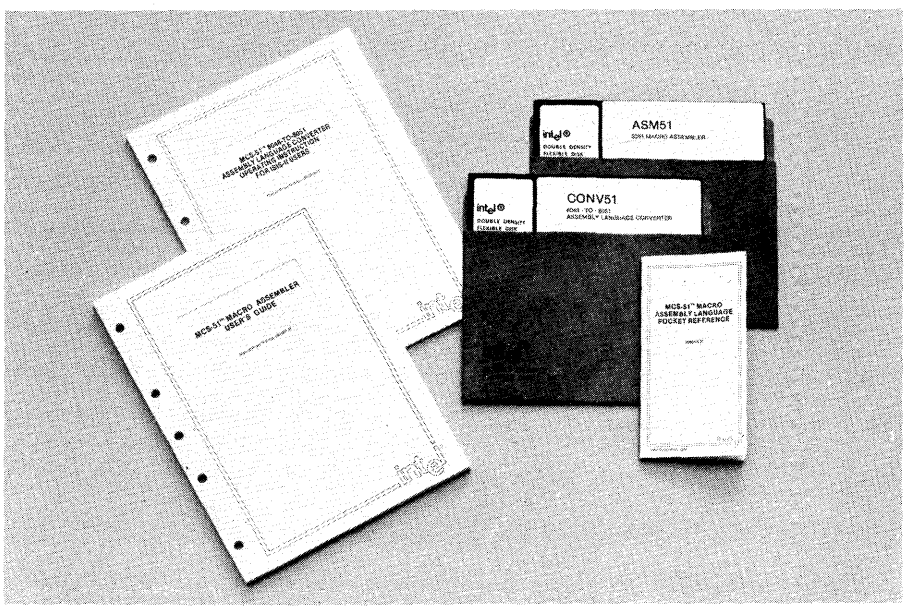
- **Symbolic assembly language programming for 8051 microcontrollers**
- **Extends Intellec® Microcomputer Development System to support 8051 program development**
- **Provides assembler output in standard Intel hex format**
- **Macro Assembler features conditional assembly and macro capabilities**
- **CONV51 Converter for translation of 8048 assembly language source code to 8051 assembly language source code**
- **Provides upward compatibility from the MCS-48™ family of single-chip microcontrollers**
- **Supports conversion of ASM48 source code macro definitions**

The 8051 software development package provides development system support for the powerful 8051 family of single chip microcomputers. The package contains a symbolic macro assembler and MCS-48 source code converter.

The assembler produces absolute machine code from 8051 macro assembly language instructions. This object code may be used to program the 8751 EPROM version of the chip. The assembler output may also be debugged using the ICE-51™ in-circuit emulator.

The converter translates 8048 assembly language instructions into 8051 source instructions to provide software compatibility between the two families of microcontrollers.

This diskette-based software package runs under ISIS-II on an Intellec Microcomputer Development System with 64K bytes of memory.



8051 MACRO ASSEMBLER

- **Supports 8051 family program development on Intellec® Microcomputer Development Systems**
- **Gives symbolic access to powerful 8051 hardware features**
- **Produces object file, listing file and error diagnostics**
- **Provides software support for many addressing and data allocation capabilities**
- **Symbolic Assembler supports symbol table, cross-reference, macro capabilities, and conditional assembly**

The 8051 Macro Assembler (ASM51) translates symbolic 8051 macro assembly language instructions into machine executable object code. These assembly language mnemonics are easier to program and are more readable than binary or hexadecimal machine instructions. Also, by allowing the programmer to give symbolic names to memory locations rather than absolute addresses, software design and debug are performed more quickly and reliably.

The assembler supports macro definitions and calls. This is a convenient way to program a frequently used code sequence only once. The assembler also provides conditional assembly capabilities.

Cross referencing is provided in the symbol table listing, showing the user the lines in which each symbol was defined and referenced.

ASM51 provides symbolic access to the many useful addressing features of the 8051 architecture. These features include referencing for bit and byte locations, and for providing 4-bit operations for BCD arithmetic. The assembler also provides symbolic access to hardware registers, I/O ports, control bits, and RAM addresses.

Math routines are enhanced by the MULtiply and DIVide instructions.

If an 8051 program contains errors, the assembler provides a comprehensive set of error diagnostics, which are included in the assembly listing or on another file. Program testing may be performed by using the Universal PROM Programmer and 8751 personality card to program the 8751 EPROM version of the chip, or by using the ICE-51 in-circuit emulator.

```
MCS-51 MACRO ASSEMBLER                                                    PAGE    1

ISIS-II MCS-51 MACRO ASSEMBLER
OBJECT MODULE PLACED IN :F1:CONVRT.HEX
ASSEMBLER INVOKED BY:  :F1:ASM51 :F1:CONVRT.51 SYMBOLS XREF

LOC  OBJ             LINE     SOURCE

                        1
                        2
                        3
                        4     ; This routine converts BCD to binary and binary to BCD.
                        5     ;
                        6     ;
                        7     ;
                        8
0100                    9             ORG 100H
0100 E7                10     CONVPT: MOV A,@R1
0101 75F00A            11             MOV B,#10       ; Load B for division or multiplication by 10
0104 4004              12             JC BINBCD
0106 54F0              13     BCDBIN: ANL A,#0F0H     ; Mask out low order digit
0108 C4                14             SWAP A          ; Move high digit into low order nibble of accumulator
                       15     ; Multiply digit by 10
0109 A4                16             MUL AB          ; Multiply digit by 10
010A C7                17             XCH A,@R1       ; Store intermediate result and get new copy of BCD
010B 540F              18             ANL A,#0FH      ; Mask out high order digit
010D 27                19             ADD A,@R1       ; Add high order digit and low order digit
010E F7                20             MOV @R1,A       ; Store result
010F 22                21             RET
0110 84                22     BINBCD: DIV AB          ; Divide by 10
0111 C4                23             SWAP A          ; Put quotient in high order nibble
0112 45F0              24             ORL A,B         ; Place remainder in low order nibble
0114 F7                25             MOV @R1,A       ; Store result
0115 22                26             RET
                       27             END

MCS-51 MACRO ASSEMBLER                                                    PAGE    2

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

NAME    TYPE     VALUE AND REFERENCES

B       N DSEG   00F0H  11 24
BCDBIN  L CSEG   0106H  13#
BINBCD  L CSEG   0110H  12 22#
CONVRT  L CSEG   0100H  10#

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

Sample ASM51 Listing

CONV51
8048 TO 8051 ASSEMBLY LANGUAGE CONVERTER UTILITY PROGRAM

- **Enables software written for the MCS-48™ family to be upgraded to run on the 8051**
- **Maps each 8048 instruction to a corresponding 8051 instruction**
- **Preserves comments; translates 8048 macro definitions and calls**
- **Provides diagnostic information and warning messages embedded in the output listing**

The 8048 to 8051 Assembly Language Converter is a utility to help users of the MCS-48 family of microcomputers upgrade their designs with the high performance 8051 architecture. By converting 8048 source code to 8051 source code, the software investment developed for the 8048 is maintained when the system is upgraded.

The goal of the converter (CONV51) is to attain functional equivalence with the 8048 code by mapping each 8048 instruction to a corresponding 8051 instruction. In some cases a different instruction is produced because of the enhanced instruction set (e.g., bit CLR instead of ANL).

Although CONV51 tries to attain functional equivalence with each instruction, certain 8048 code sequences cannot be automatically converted. For example, a delay routine which depends on 8048 execution speed would require manual adjustment. A few instructions, in fact, have no 8051 equivalent (such as those involving P4-P7). Finally, there are a few areas of possible intervention such as PSW manipulation and interrupt processing, which at least require the user to confirm proper translation. The converter always warns the user when it cannot guarantee complete conversion.

CONV51 produces two files. The output file contains the ASM51 source program produced from the 8048 instructions. The listing file produces correlated listings of the input and output files, with warning messages in the output file to point out areas that may require users' intervention in the conversion.

SPECIFICATIONS

OPERATING ENVIRONMENT

Required Hardware:

Intellec Microcomputer Development System with

64K Bytes of RAM

Flexible Disk Drive(s)

System Console

—CRT or hard copy device

Optional Hardware:

Universal PROM Programmer
Line Printer
ICE-51 In-Circuit Emulator

Required Software:

ISIS-II Diskette Operating System (V3.4 or later)

Documentation Package:

MCS-51 Macro Assembler User's Guide

MCS-51 Macro Assembly Language Pocket Reference

MCS-51 8048-to-8051 Assembly Language Converter Operating Instructions for ISIS-II Users

ORDERING INFORMATION

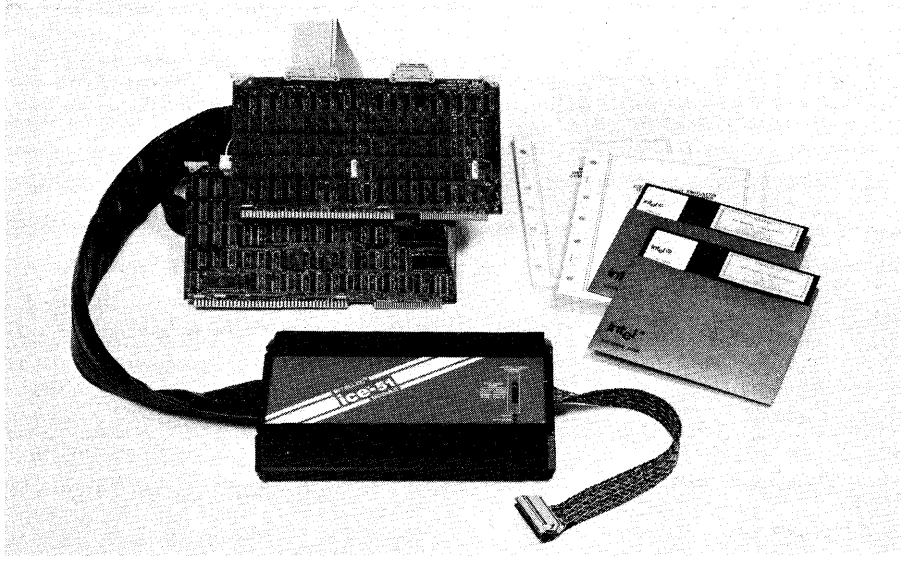
Part Number	Description
MCI-51-ASM	8051 Software Development Package

PRELIMINARY

ICE-51™
8051 IN-CIRCUIT EMULATOR

- **Precise, full-speed, real-time emulation**
  - **— Load, drive, timing characteristics**
  - **— Full-speed program RAM**
  - **— Serial and parallel ports**
- **User-specified breakpoints**
- **Execution trace**
  - **— User-specified qualifier registers**
  - **— Conditional trigger**
  - **— Symbolic groupings and display**
  - **— Instruction and frame modes**
- **Emulation timer**
- **Full symbolic debugging**
- **Single-line assembly and disassembly for program instruction changes**
- **Macro commands and conditional block constructs for automated debugging sessions**
- **HELP facility: ICE-51 command syntax reference at the console**
- **User confidence test of ICE-51 hardware**

The ICE-51 module resides in the Intellec® Microcomputer Development System and interfaces to any user-designed 8051 system through a cable terminating in an 8051 emulator microprocessor and a pin-compatible plug. The emulator processor, together with 8K bytes of user program RAM located in the ICE-51 buffer box, replaces the 8051 device in the user system while maintaining the 8051 electrical and timing characteristics. Powerful Intellec debugging functions are thus extended into the user system. Using the ICE-51 module, the designer can emulate the system's 8051 in real-time or single-step mode. Breakpoints allow the user to stop emulation on user-specified conditions, and a trace qualifier feature allows the conditional collection of 1000 frames of trace data. Using the single-line 8051 assembler the user may alter program memory using ASM51 mnemonics and symbolic references, without leaving the emulator environment. Frequently used command sequences can be combined into compound commands and identified as macros with user-defined names.



FUNCTIONAL DESCRIPTION

Integrated Hardware and Software Development

The ICE-51 emulator allows hardware and software development to proceed interactively. This approach is more effective than the traditional method of independent hardware and software development followed by system integration. With the ICE-51 module, prototype hardware can be added to the system as it is designed. Software and hardware integration occurs while the product is being developed.

The ICE-51 emulator assists four stages of development:

SOFTWARE DEBUGGING

It can be operated without being connected to the user's system before any of the user's hardware is available. In this stage ICE-51 debugging capabilities can be used in conjunction with the Intellec text editor and 8051 macroassembler to facilitate program development.

HARDWARE DEVELOPMENT

The ICE-51 module's precise emulation characteristics and full-speed program RAM make it a valuable tool for debugging hardware, including time-critical serial port, parallel port, and timer interfaces.

SYSTEM INTEGRATION

Integration of software and hardware can begin when any functional element of the user system hardware is connected to the 8051 socket. As each section of the user's hardware is completed, it is added to the prototype. Thus, each section of the hardware and software is "system" tested in real-time operation as it becomes available.

SYSTEM TEST

When the user's prototype is complete, it is tested with the final version of the user system software. The ICE-51 module is then used for real-time emulation of the 8051 to debug the system as a completed unit.

The final product verification test may be performed using the 8751 EPROM version of the 8051 microcomputer. Thus, the ICE-51 module provides the user with the ability to debug a prototype or production system at any stage in its development without introducing extraneous hardware or software test tools.

Symbolic Debugging

The ICE-51 emulator permits the user to define and use symbolic, rather than absolute, references to program and data memory addresses; additional symbols are predefined by the ICE-51 software for referencing registers, flags, and input/output ports. Thus, the user need not recall or look up the addresses of key locations in his program as they change with each assembly, or become involved with machine code.

When a symbol is used for memory reference in an ICE-51 emulator command, the emulator supplies the corresponding location as stored in the ICE-51 emulator symbol table. This table can be loaded with the symbol table produced by the assembler during application program assembly. The user can obtain the symbol table during software preparation simply by using the "DEBUG" switch in the ASM51 macroassembler. Furthermore, the user can interactively modify the emulator symbol table by adding new symbols or changing or deleting old ones. This feature provides great flexibility in debugging and minimizes the need to work with hexadecimal values.

Through symbolic references in combination with other features of the emulator, the user can easily:

- Interpret the results of emulation activity collected during trace.
- Disassemble program memory to mnemonics, or assemble mnemonic instructions to executable code.
- Examine or modify 8051 internal registers, data memory, or port contents.
- Reference labels or addresses defined in a user program.

Automated Debugging and Testing

MACRO COMMAND

A macro is a set of commands which is given a name. A group of commands which is executed frequently can be defined as a macro. The user can execute the group of commands by typing a colon followed by the macro name. Up to ten parameters may be passed to the macro.

Macro commands can be defined at the beginning of a debug session and then used throughout the whole session. The user can save one or more macro definitions on diskette for later use. The Intellec text editor may be used to edit the macro file. The macro definitions are easy to include in any later emulation session.

The power of the development system can be applied to manufacturing testing as well as development by writing test sequences as macros. The macros are stored on diskettes for use during system test.

COMPOUND COMMAND

Compound commands provide conditional execution of commands (IF command) and execution of commands repeatedly until certain conditions are met (COUNT, REPEAT commands).

Compound commands may be nested any number of times, and may be used in macro commands.

Example:

```
*DEFINE .I = 0          ; Define symbol .I to 0
*COUNT 100H            ; Repeat the following
                        commands 100H times.
.*IF .I AND 1 THEN     ; Check if .I is odd
..*BYTE .I = .I        ; Fill the memory at location .I
                        to value .I
..*END
.*.I = .I + 1          ; Increment .I by 1.
.*END                  ; Command executes upon
                        carriage-return after END
```

(The characters *, .*, and ..* shown in this example are system prompts which include an indication of the nesting level of compound commands.)

Operating Modes

The ICE-51 software is an Intellec RAM-based program that provides the user with easy-to-use commands for initiating emulation, defining breakpoints, controlling trace data collection, and displaying and controlling system parameters. ICE-51 commands are configured with a broad range of modifiers which provide the user with maximum flexibility in describing the operation to be performed.

EMULATION

The ICE-51 module can emulate the operation of a prototype 8051 system, at real-time speed (1.2 to 12 MHz) or in single steps. Emulation commands to the ICE-51 module control the process of setting up, running, and halting an emulation of the user's 8051-based system. Breakpoints and tracepoints enable the ICE-51 emulator to halt emulation and provide a detailed trace of execution in any part of the user's program. A summary of the emulation commands is shown in Table 1.

Breakpoints

The ICE-51 hardware includes two breakpoint registers that allow the user to halt emulation when specified conditions are met. The emulator continuously compares the values stored in the breakpoint registers with the status of specified address, opcode, operand, or port values, and halts emulation when this comparison is satisfied. When an instruction initiates a break, that instruction is executed completely before the break takes place. The ICE-51 emulator then regains control of the console and enters the Interrogation Mode. With the breakpoint feature, the user can request an emulation break when his program:

- Executes an instruction at a specific address or within a range of addresses.
- Executes a particular opcode.
- Receives a specific signal on a port pin.
- Fetches a particular operand from the user program memory.
- Fetches an operand from a specific address in program memory.

Table 1. Major Emulation Commands

Command	Description
GO	Begins real-time emulation and optionally specifies break conditions.
BR0, BR1, BR	Sets or displays either or both Breakpoint Registers used for stopping real-time emulation.
STEP	Performs single-step emulation.
QR0, QR1	Specifies match conditions for qualified trace.
TR	Specifies or displays trace-data collection conditions and optionally sets Qualifier Register (QR0, QR1).
Synchronization Line Commands	Set and display status of synchronization line outputs or latched inputs. Used to allow real-time emulation or trace to start and stop synchronously with external events.

Trace and Tracepoints

Tracing is used with both real-time and single-step emulation to record diagnostic information in the trace buffer as a program is executed. The information collected includes opcodes executed, port values, and memory addresses. The ICE—51 emulator collects 1000 frames of trace data.

This information can be displayed as assembler instruction mnemonics, if desired, for analysis during interrogation or single-step mode. The trace-collection facility may be set to run conditionally or unconditionally. Two unique trace qualifier registers, specified in the same way as break-

point registers, govern conditional trace activity. The qualifiers can be used to condition trace data collection to take place as follows:

- Under all conditions (forever).
- Only while the trace qualifier is satisfied.
- For the frames or instructions preceding the time when a trace qualifier is first satisfied (pre-trigger trace).
- For the frames or instructions after a trace qualifier is first satisfied (post-triggered trace).

Table 2 shows an example of a trace display.

Table 2. Trace Display (Instruction Mode)

```
*T ALL
FRAME  LOC    OBJ     INSTRUCTION                    P1   P2  P3   IOVF
0000:  0000H  E0      MOVX  @DPTR,A                  00H  00H FFH  0
0007:  0001H  F8      MOV   R0,A                     00H  00H FFH  0
0011:  0002H  F4      CPL   A                        00H  00H FFH  0
0015:  0003H  32      RLC   A                        00H  00H FFH  0
0019:  0004H  75E0AA  MOV   ACC,#0AAH                00H  00H FFH  0
0027:  0007H  C3      CLR   C                        00H  00H FFH  0
0031:  0008H  98      SUBB  A,R0                     AAH  00H FFH  0
0035:  0009H  0000    NOP   00H                      AAH  00H FFH  0
0043:  000AH  75E0AA  MOV   ACC,#0AAH                AAH  00H FFH  0
0051:  000DH  0150    AJMP  0050H                    AAH  00H FFH  0
```

INTERROGATION AND UTILITY

Interrogation and utility commands give the user convenient access to detailed information about the user program and the state of the 8051 that is useful in debugging hardware and software. Changes can be made in memory and in the 8051 registers, flags, and port values. Commands are also provided for various utility operations such as loading and saving program files, defining symbols, displaying trace data, controlling system synchronization and returning control to ISIS-II. A summary of the basic interrogation and utility commands is shown in Table 3. Two time-saving emulator features are discussed below.

SINGLE-LINE ASSEMBLER/DISASSEMBLER — The single-line assembler/disassembler (ASM and DASM commands) permits the designer to examine and alter program memory using assembly language mnemonics, without leaving the emulator environment or requiring time-consuming program reassembly. When assembling new mnemonic instructions into program memory, previously defined symbolic references (from the original program assembly, or subsequently defined during the emulation session) may be used in the instruction operand field. The emulator will supply the absolute address or data values as stored in the emulator symbol table. These features eliminate user time spent translating to and from machine code and searching for absolute addresses, with a corresponding reduction in transcription errors.

Table 3. Major Interrogation and Utility Commands

Command	Description
HELP	Displays help messages for ICE-51 emulator command-entry assistance.
LOAD	Loads user object program (8051 code) into user program memory, and user symbols into ICE-51 emulator symbol table.
SAVE	Saves ICE-51 emulator symbol table and/or user object program in ISIS-II hexadecimal file.
LIST	Copies all emulator console input and output to ISIS-II file.
EXIT	Terminates ICE-51 emulator operation.
DEFINE	Defines ICE-51 emulator symbol or macro.
REMOVE	Removes ICE-51 emulator symbol or macro.
ASM	Assembles mnemonic instructions into user program memory.
DASM	Disassembles and displays user program memory contents.
Change/Display Commands	Change or display value of symbolic reference in ICE-51 emulator symbol table, contents of key-word references (including registers, I/O ports, and status flags), or memory references.
EVALUATE	Evaluates expression and displays resulting value.
MACRO	Displays ICE-51 macro or macros.
INTERRUPT	Displays serial, external, or timer interrupt register settings.
SECONDS	Displays contents of emulation timer, in microseconds.
Trace Commands	Position trace buffer pointer and select format for trace display.
PRINT	Displays trace data pointed to by trace buffer pointer.

HELP — The HELP file allows the user to display ICE-51 command syntax information at the Intellec console. By typing "HELP", a listing of all items for which help messages are available is displayed; typing "HELP <Item>" then displays relevant information about the item requested, including typical usage examples. Table 4 shows some sample HELP messages.

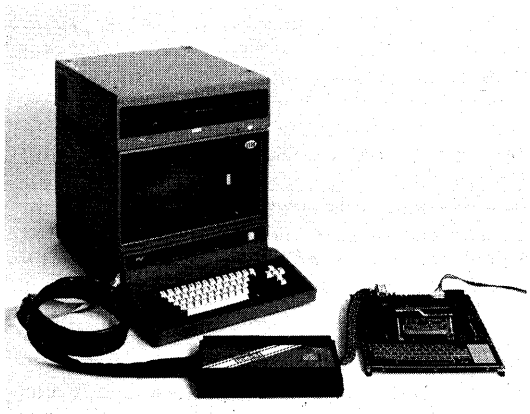


Figure 1. A Typical 8051 Development Configuration. The host system is an Intellec Model 225, plus 1 megabyte dual double-density flexible disk storage. The ICE-51 module is connected to an SDK-51 system design kit.

Emulation Accuracy

The speed and interface demands of a high-performance single-chip microcomputer require extremely accurate emulation, including full-speed, real-time operation with the full function of the microcomputer. The ICE-51 emulator achieves accurate emulation with an 8051 bond-out chip, a special configuration of the 8051 microcomputer family, as its emulation processor.

Each of the 40 pins on the user plug is connected directly to the corresponding 8051 pin on the bond-out chip. Thus the user system sees the emulator as an 8051 microcomputer at the 8051 socket. The resulting characteristics provide extremely accurate emulation of the 8051, including speed, timing characteristics, load and drive values, and crystal operation. The emulator may draw more power from the user system than a standard 8051 family device.

Additional bond-out pins provide signals such as internal address, data, clock, and control lines to the emulator buffer box. These signals let static RAM in the buffer box substitute for on-chip program ROM or EPROM or external program memory. The 8K bytes of full-speed RAM in the buffer box can be mapped in 4K blocks to anywhere within the 64K program memory space of the 8051. The bond-out chip also gives the emulator "back-door" access to internal chip operation, so that the emulator can break and trace execution without interfering with the values on the user-system pins.

Table 4. HELP Command

```
*HELP
Help is available for the following items. Type HELP followed by
the item name. The help items cannot be abbreviated. (For more
information, type HELP HELP or HELP INFO.)
Emulation:    Trace Collection:  Misc:       <address>
CO CR  SYP    TR CP CPP CPI SYJ  BASE        <CPUSkeyword>
BR BRP BRI                       DISABLE     <expr>
STEP          Trace Display:     ENABLE      <ICESISkeyword>
              TRACE MOVE PRINT   ERROR       <identifier>
              OLDEST NEWEST      EVALUATE    <instruction>
                                 HELP        <maskedSconstant>
Change/Display/Define/Remove:    INFO        <matchScond>
<CHANCE>  REMOVE CPYTE PBIT      <LIGHTS>    <numericSconstant>
<DISPLAY> SYMBOL PBYTE PASM      LIST        <partition>
RECISTER  RESET  PBYTE ASM       LOAD        <string>
SECONDS   WRITE  RPYTE MAP       SAVE        <stringSconstant>
DEFINE    STACK  XBYTE SY        SUFFIX      <symbolicSref>
                                 SYMBOLIC    <systemSsymbols>

Macro:                Compound               <traceSreference>
DEFINE  DIR           Commands:              <unlimitedSmatchScond>
DISABLE ENABLE        COUNT                  <usersSsymbols>
INCLUDE PUT           IF
<MACROSDISPLAY>       REPEAT
<MACROSINVOCATION>
*
```

```
*HELP IF
IF - The conditional command allows conditional execution of one
or more commands based on the values of boolean conditions.
     IF <expr> [THEN] <cr>       <trueSlist>::=[<command> <cr>]P
      <trueSlist>                <falseSlist>::=[<command> <cr>]P
      [OBIF <expr> <cr>          <command>::=An ICE-51 command.
      <trueSlist>]P
      [ELSE <cr>
      <falseSlist>]
      END
The <expr>s are evaluated in order as 16-bit unsigned integers.
If one is reached whose value has low-order bit 1 (TRUE), all
commands in the <trueSlist> following that <expr> are then
executed and all commands in the other <trueSlist>s and in the
<falseSlist> are skipped.  If all <expr>s have value with low-
order bit 0 (FALSE), then all commands in all <trueSlist>s are
skipped and, if ELSE is present, all commands in the <falseSlist>
are executed.
     (EX:  IF .LOOP=5 THEN
             STEP
             ELSE
             GO
             END)
```

SPECIFICATIONS

ICE-51 Operating Requirements

Intellec® Microcomputer Development System (64K RAM required)

System console

Intellec® Diskette Operating System (single or double density) ISIS-II v. 3.4 or later

Equipment Supplied

- Printed circuit boards (2)
- Emulation buffer box, Intellec interface cables, and user-interface cable with 8051 emulation processor
- Crystal power accessory
- Operating instructions manual
- Diskette-based ICE-51 software (single and double density)

Emulation Clock

User's system clock (1.2 to 12 MHz) or ICE-51 crystal power accessory (12 MHz)

Environmental Characteristics

Operating Temperature: 0° to 40°C
Operating Humidity: Up to 95% relative humidity without condensation.

Physical Characteristics

Printed Circuit Boards
Width: 12.00 in. (30.48 cm)
Height: 6.75 in. (17.15 cm)
Depth: 0.50 in. (1.27 cm)

Buffer Box
Width: 8.00 in. (20.32 cm)
Length: 12.00 in. (30.48 cm)
Depth: 1.75 in. (4.44 cm)
Weight: 4.0 lb (1.81 kg)

Electrical Characteristics

DC Power Requirements (from Intellec system)

$V_{CC} = +5V, +5\%, -1\%$
$I_{CC} = 13.2A$ max; 11.0A typical
$V_{DD} = +12V, \pm 5\%$
$I_{DD} = 0.1A$ max; 0.05A typical
$V_{BB} = -10V, \pm 5\%$
$I_{BB} = 0.05A$ max; 0.01A typical

User plug characteristics at 8051 socket

Same as 8031, 8051, or 8751, except that the user system will see an added load of 25 pF capacitance and 50 μA leakage from the ICE-51 emulator user plug at ports 0, 1, and 2.

ORDERING INFORMATION

Part Number	Description
MCI-51-ICE	8051 Microcontroller In-Circuit Emulator, cable assembly and interactive diskette software

UPP-103*
UNIVERSAL PROM PROGRAMMER

**Replaces UPP-101, UPP-102 Universal PROM Programmers*

Intellec development system peripheral for PROM programming and verification

Provides personality cards for programming all Intel PROM families

Provides zero insertion force sockets for both 16-pin and 24-pin PROMs

Universal PROM mapper software provides powerful data manipulation and programming commands

Provides flexible power source for system logic and programming pulse generation

Holds two personality cards to facilitate programming operations using several PROM types

The UPP-103 Universal PROM Programmer is an Intellec system peripheral capable of programming and verifying all of the Intel programmable ROMs (PROMs). In addition, the UPP-103 programs the PROM memory portions of the 8748 microcomputer, 8741 UPI, the 8755 PROM and I/O chip and the 2920 signal processor. Programming and verification operations are initiated from the Intellec development system console and are controlled by the universal PROM mapper (UPM) program.

FUNCTIONAL DESCRIPTION

Universal PROM Programmer

The basic Universal PROM Programmer (UPP) consists of a controller module, two personality card sockets, a front panel, power supplies, a chassis, and an Intellec development system interconnection cable. An Intel 4040-based intelligent controller monitors the commands from the Intellec System and controls the data transfer interface between the selected PROM personality card and the Intellec memory. A unique personality card contains the appropriate pulse generation functions for each Intel PROM family. Programming and verifying any Intel PROM may be accomplished by selecting and plugging in the appropriate personality card. The front panel contains a power-on switch and indicator, a reset switch, and two zero-force insertion sockets (one 16-pin and one 24-pin or two 24-pin). A central power supply provides power for system logic and for PROM programming pulse generation. The Universal PROM Programmer may be used as a table top unit or mounted in a standard 19-inch RETMA cabinet.

Universal PROM Mapper

The Universal PROM Mapper (UPM) is the software program used to control data transfer between paper tape or diskette files and a PROM plugged into the Universal PROM Programmer. It uses Intellec system memory for intermediate storage. The UPM transfers data in 8-bit HEX, BNPF, or binary object format between paper tape or diskette files and the Intellec system memory. While the data is in Intellec system memory, it can be displayed and changed. In addition, word length, bit position, and data sense can be adjusted as required for the PROM to be programmed. PROMs may also be duplicated or altered by copying the PROM contents into the Intellec system memory. Easy to use program and compare commands give the user complete control over programming and verification operations. The UPM eliminates the need for a variety of personalized PROM programming routines because it contains the programming algorithms for all Intel PROM families. The UPM (diskette based version) is included with the Universal PROM Programmer.

SPECIFICATIONS

Hardware Interface

Data — Two 8-bit unidirectional buses

Commands — 3 write commands, 2 read commands, one initiate command

Physical Characteristics

Width — 6 in. (14.7 cm)

Height — 7 in. (17.2 cm)

Depth — 17 in. (41.7 cm)

Weight — 18 lb (8.2 kg)

Electrical Characteristics

AC Power Requirements — 50-60 Hz; 115/230V AC: 80W

Environmental Characteristics

Operating Temperature — 0°C to 55°C

Optional Equipment

Personality Cards

UPP-816: 2716 personality card

UPP-832: 2732 personality card

UPP-848: 8748, 8741 personality card with 40-pin adaptor socket

UPP-865: 3602, 3622, 3602A, 3622A, 3621, 3604, 3624, 3604A, 3624A, 3604AL, 36046-6, 3605, 3605A, 3625, 3625A, 3608, 3628, 3636

UPP-872: 8702A/1702A personality card

UPP-878: 8708/8704/2708/2704 personality card

UPP-955: 8755A personality card with 40-pin adaptor socket

PROM Programming Sockets

UPP-501: 16-pin/24-pin socket pair

UPP-502: 24-pin/24-pin socket pair

UPP-562: Socket adaptor for 3621, 3602, 3622, 3602A, 3622A

UPP-555: Socket adaptor for 3604AL, 36046-6, 3608, 3628, 3636

UPP-566: Socket adaptor for 3605, 3625, 3605A, 3625A

Equipment Supplied

Cabinet

Power supplies

4040 intelligent controller module

Specified zero insertion force socket pair

Intellec development system interface cable

Universal PROM Mapper program (diskette-based version)

Reference Manuals

9800819 — Universal PROM Programmer User's Manual (SUPPLIED)

ORDERING INFORMATION

Part Number	Description
UPP-103	Universal PROM programmer with 16-pin/24-pin socket pair and 24-pin/24-pin socket pair.

PRELIMINARY

SDK-51
MCS-51 SYSTEM DESIGN KIT

- **Complete single-board microcomputer kit:**
  - **— Intel 8031 CPU**
  - **— ASCII keyboard and 24-character alpha-numeric display**
  - **— Wire-wrap area for custom circuitry**
  - **—User-configurable RAM**
  - **— Serial and parallel interfaces**
- **Extensive system software in ROM:**
  - **— Single-line assembler and disassembler**
  - **— System debugging commands**
    - **Go**
    - **Step**
    - **Breakpoints**
- **Interface software:**
  - **— Serial port**
  - **— Audio cassette**
  - **— Intellec® system**
- **User's guide, assembly manual, and MCS-51 design manuals**

The SDK-51 MCS-51 System Design Kit contains all of the components required to assemble a complete single-board microcomputer based on Intel's high-performance 8051 single-chip microcomputer. SDK-51 uses the external ROM version of the 8051 (8031). Once you have assembled the kit and supplied +5V power, you can enter programs in MCS-51 assembly language mnemonics, translate them into MCS-51 object code, and run them under control of the system monitor. The kit supports optional memory and interface configurations, including a serial terminal link, audio cassette storage, EPROM program memory, and Intellec® development system upload and download capability.

FUNCTIONAL DESCRIPTION

The SDK-51 is a kit which includes hardware and software components to assemble a complete MCS-51 family single-board microcomputer. Only common laboratory tools and test equipment are required to assemble the kit. Assembly generally requires 5 to 10 hours, depending on the experience of the user.

The MCS-51 Microcomputer Series

MCS-51 is a series of high-performance single-chip microcomputers for use in sophisticated real-time applications such as instrumentation, industrial control and intelligent computer peripherals. The 8031, 8051, and 8751 microcomputers belong to the 8051 family, which is the first family in the MCS-51 series.

In addition to their advanced features for control applications, MCS-51 family devices have a microprocessor bus and arithmetic capability such as hardware multipy and divide instructions, which make the SDK-51 a versatile stand-alone microcomputer board.

The 8031, 8051, and 8751 CPUs

The 8031, 8051, and 8751 CPUs each combine, on a single chip, a 128 x 8 data RAM; 32 input/output lines; two 16-bit timer/event counters; a five-source, two-level nested interrupt structure; a serial I/O port; and on-chip oscillator and clock circuits. An 8051 block diagram is shown in Figure 1.

The 8031, the SDK-51's CPU, is a CPU without on-chip program memory. The 8031 can address 64K bytes of external program memory in addition to 64K bytes of external data memory. For systems requiring extra capability, each member of the 8051 family can be expanded using standard memories and the byte-oriented MCS-80 and MCS-85 peripherals. The 8051 is an 8031 with the lower 4K bytes of program memory filled with on-chip mask-programmable ROM while the 8751 has 4K bytes of ultraviolet light-erasable, electrically programmable ROM (EPROM).

The 8031 CPU operates at a 12 MHz clock rate, resulting in 4 μs multiply and divide and other instructions of 1 μs and 2 μs.

For additional information on the 8051 family, see the 8051 User's Manual or MCS-51 Macroassembler User's Guide.

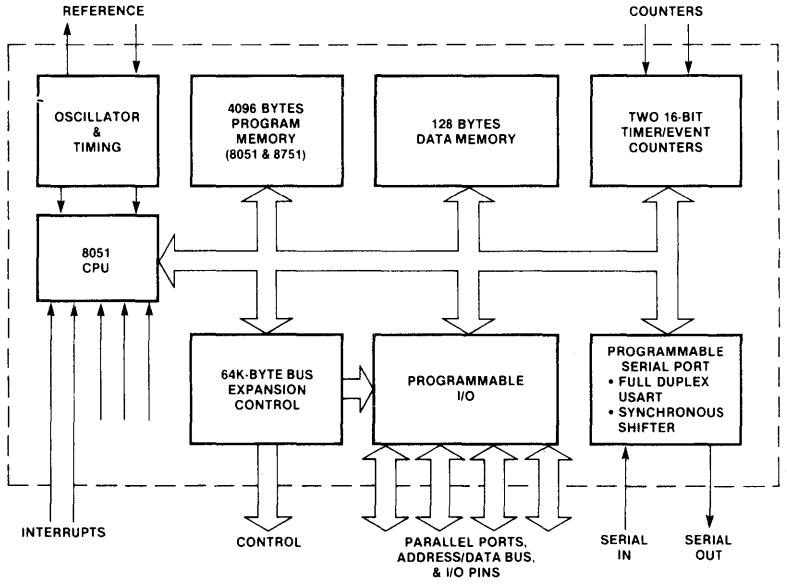


Figure 1. 8051 Block Diagram

System Software

A compact but powerful system monitor is contained in 8K bytes of pre-programmed ROM. The monitor includes system utilities such as command interpretation, user program debugging, and interface controls. Table 1 summarizes the SDK-51 monitor commands.

The ROM devices also include a single-line assembler and disassembler. The assembler lets you enter programs in MCS-51 assembly language mnemonics directly from the ASCII keyboard. The disassembler supports debugging by letting you look at MCS-51 instructions in mnemonic form during system interrogation.

Memory

The two 64K external memory spaces are combined into a single memory space which you can configure between program memory and data memory. The kit includes 1K-byte of static RAM. The board has space and printed circuitry for an additional 15K bytes of RAM and 8K bytes of ROM.

User Interface

The kit includes a typewriter-format, ASCII-subset keyboard and a 24-character, alpha-numeric LED

Table 1. SDK-51 Commands

Command	Operation
Set breakpoint	Define addresses for breaking execution.
Display cause	Ask the system why execution stopped.
Upload, download	Transfer files to and from Intellec® development system.
Save, load	Transfer files to and from optional cassette interface.
Set top of program memory	Define partition between program memory and data memory.
Set baud	Define baud rate value of serial port.
Display memory	Examine and change program memory or data locations.
Assemble	Translate an MCS-51 assembly mnemonic into object code.
Disassemble	Translate program memory into MCS-51 assembly language mnemonics.
Go	Start execution between a selected pair of addresses.
Step	Execute a specified number of instructions.

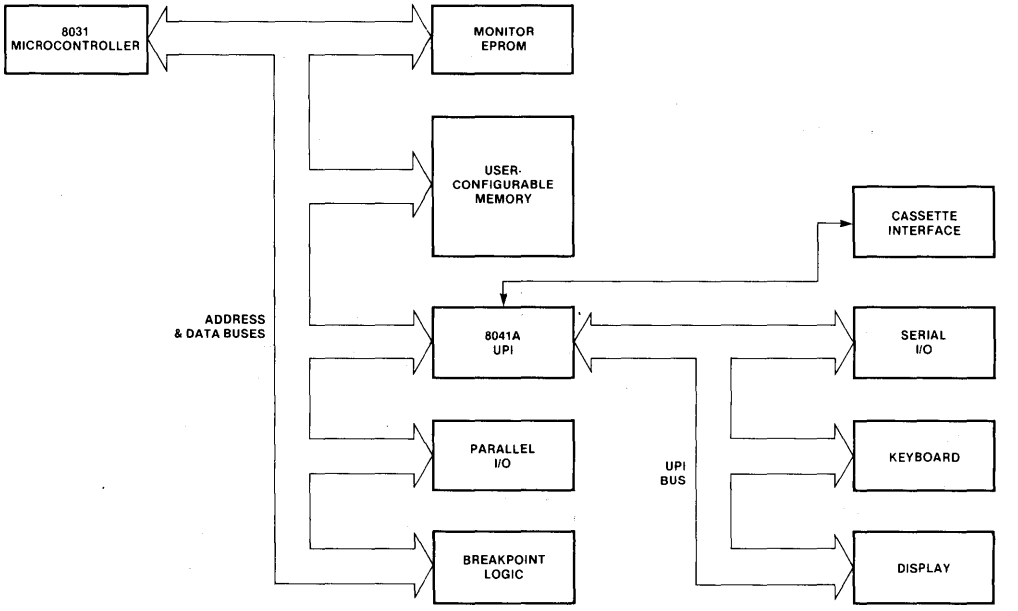


Figure 2. Block Diagram of SDK-51 System Design Kit

AFN-01792A

display. The standard keyboard and display provide full access to all of the SDK-51's capabilities. All of the SDK-51 interfaces are controlled by a pre-programmed Intel 8041 Universal Peripheral Interface.

A 3 x 4 matrix keyboard can be jumpered to port 1 of the 8031.

Optional Interfaces

TERMINAL

An RS-232-compatible CRT or printing terminal or a current-loop-interface terminal may be used as a listing device by connecting it to the board's serial interface connector and supplying + 12 and − 12 volts to the board.

AUDIO CASSETTE

The kit includes hardware, software, and user's guide instructions to connect and operate an audio cassette tape recorder for low-cost program and data storage.

INTELLEC SYTSTEM

An SDK-51 and an Intellec Model 800 or Series II development system with ISIS-II can upload and download files through the serial interface without adding any software to the Intellec system.

Parallel I/O

The kit includes an Intel 8155 parallel I/O device which expands the 8031 I/O capability by providing 22 dedicated parallel lines. Three 40-pin headers between the 8031 and 8155 devices and the wire-wrap area facilitate interconnections with the user's custom circuitry.

Debugging

Hardware breakpoint logic in the SDK-51 checks the address of a program or external data-memory access against values defined by the user and stops execution when it sees a "break" condition. After a breakpoint, you can examine and modify registers, memory locations, and other points in the system. A step command lets you execute instructions in a single-step mode.

Assembly and Test

The SDK-51 assembly manual describes hardware assembly in a step-by-step process that includes checking each hardware subsystem as it is installed. Building the system requires only a few common tools and standard laboratory instruments.

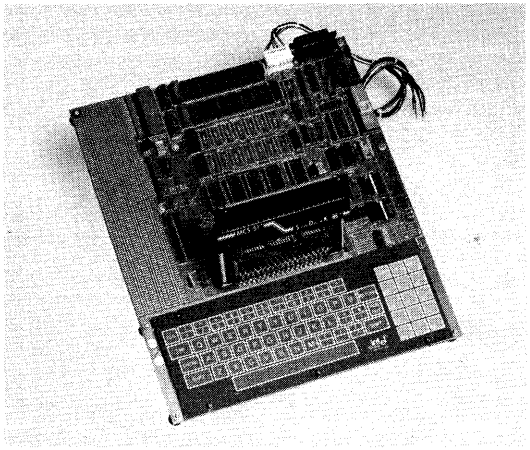


Figure 3. SDK-51 Assembled with Additional RAM and ROM Devices Installed

SPECIFICATIONS

Control Processor

Intel 8031 microcomputer
12 MHz clock rate

Memory

RAM — 1K-byte static, expandable in 1K segments to 16K-byte with 2114 RAM devices; user-configurable as program or data memory.

ROM — Printed circuitry for 8K bytes of program memory in 4K segments using 2732A EPROM devices.

Interfaces

Keyboard — 51-key, ASCII subset, typewriter format, 12-key (3 x 4) matrix

Display — 24-character, alpha-numeric

Serial — RS-232 with user-selectable baud rate. Printed circuitry for 110 baud 20 mA current loop interface. 8031 serial port.

Parallel — 22 lines, TTL compatible

Cassette — Audio cassette tape storage interface

AFN-01792A

Software

System monitor preprogrammed in on-board ROM

MCS-51 assembler and disassembler preprogrammed in on-board ROM

Interface control software preprogrammed in 8041's on-chip ROM

Assembly and Test Equipment Required

Needle-nose pliers

Small Phillips screwdriver

Small diagonal wire cutters

Soldering pencil, ≤30 watts, 1/16″ diameter tip

Rosin-core, 60-40 solder, 0.05″ diameter

Volt-Ohm-Milliammeter, 1 meg-ohm input impedance

Oscilloscope, 1 volt/division vertical sensitivity, 200 μs/division sweep rate, single trace, internal and external triggering

Physical Characteristics

Length — 13.5 in. (34.29 cm)
Width — 12 in. (30.48 cm)
Height — 4 in. (10.16 cm)
Weight — 3 lb (1.36 kg)

Electrical Characteristics

DC Power Requirement (supplied by user, cable included with kit)

Voltage	Current
+ 5V ±5%	3A
+12V ±5%*	100 mA
−12V ±5%*	100 mA

* ± 12 volts required only for operation with serial interface.

Environmental Characteristics

Operating Temperature — 0 to 40°C

Relative Humidity — 10% to 90%, non-condensing

Reference Manuals

SDK-51 User's Manual

SDK-51 Assembly Manual

SDK-51 Monitor Listing

MCS-51 Macro Assembler User's Guide

MCS-51 Macro Assembly Language Pocket Reference

ORDERING INFORMATION

Part Number	Description
MCI-51-SDK	MCS-51 System Design Kit

Appendix A
PL/M Description of 8051 Instruction Set

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SIM51
OBJECT MODULE PLACED IN SIM51.OBJ
COMPILER INVOKED BY:  :F1:PLM80 SIM51.PLM PRINT(:F1:SIM51.LST) XREF DATE(302)


            $TITLE('8051 INSTRUCTION SET SIMULATOR')


            /*  THE FOLLOWING IS A PLM-80 PROGRAM TO SIMULATE THE
                OPERATION OF THE INTEL 8051 INSTRUCTION SET.
                NO ATTEMPT HAS BEEN MADE TO SIMULATE THE I/O PORTS OR
                SPECIAL FUNCTION REGISTERS, THOUGH THERE ARE 'HOOKS' TO
                ACCESS EXTERNALLY-DEFINED PROCEDURES WHENEVER P0-P3 OR
                SBUF ARE READ OR WRITTEN.

            RELEVANT ENTRY POINTS:

            'INITIALIZE' - SIMULATE HARDWARE RESET;
                SUBROUTINE WITH NO INPUT PARAMETERS.

            'STEP' - SIMULATE EXECUTION OF ONE INSTRUCTION
                INPUT PARAMETER = STARTING ADDRESS;
                VALUE RETURNED = UPDATED PC.

            'FETCH$SIM' - FETCH DATA FROM VARIOUS ADDRESS SPACES
                (SEE ROUTINE DEFINITION FOR PARAMETER DEFINITION).

            'STORE$SIM' - STORE DATA INTO VARIOUS ADDRESS SPACES
                (SEE ROUTINE DEFINITION FOR PARAMETER DEFINITION).

            ALL PARAMETERS PASSED TO AND FROM ALL ROUTINES ARE SIXTEEN-BIT.
```

```
                $EJECT

  1             SIM51:
                DO;

                /*  DEFINITIONS OF GLOBAL VARIABLES USED BY INDIVIDUAL
                    INSTRUCTION SIMULATION PROCEDURES:                    */

  2    1        DECLARE ROM$SIZE LITERALLY '4096';  /*  4K ROM SUPPORTED  */
  3    1        DECLARE INT$RAM(128) BYTE;
  4    1        DECLARE HARD$REG(128) BYTE;
  5    1        DECLARE USER$CODE(ROM$SIZE) BYTE PUBLIC;
  6    1        DECLARE EXTERN$RAM(256) BYTE;

  7    1        DECLARE REG$ADDR BYTE;
  8    1        DECLARE DIR$ADDR BYTE;
  9    1        DECLARE SOURCE$ADDR BYTE;
 10    1        DECLARE DEST$ADDR BYTE;
 11    1        DECLARE BIT$ADDR BYTE;
 12    1        DECLARE CODE$ADDR ADDRESS;
 13    1        DECLARE PAGED$EXTERNAL$ADDR BYTE;

 14    1        DECLARE REG$DATA BYTE;
 15    1        DECLARE DIR$DATA BYTE;
 16    1        DECLARE IND$DATA BYTE;
 17    1        DECLARE IMM$DATA BYTE;
 18    1        DECLARE BIT$DATA BYTE;
 19    1        DECLARE STACK$DATA BYTE;

 20    1        DECLARE PAGE$CODE BYTE;
 21    1        DECLARE PAGE$OFFSET BYTE;
 22    1        DECLARE DISPLACEMENT BYTE;

 23    1        DECLARE LINK$BIT BYTE;
 24    1        DECLARE LOW$NIB BYTE;
 25    1        DECLARE HIGH$NIB BYTE;
 26    1        DECLARE LOW$SOURCE$NIB BYTE;

 27    1        DECLARE ADD$TEMP ADDRESS;
 28    1        DECLARE SUB$TEMP BYTE;
 29    1        DECLARE MUL$TEMP ADDRESS;
 30    1        DECLARE DIV$TEMP BYTE;

 31    1        DECLARE PC ADDRESS;
 32    1        DECLARE OPCODE BYTE;
 33    1        DECLARE MACH$CYC ADDRESS PUBLIC;

 34    1        DECLARE BIT$REG$ADDR BYTE;
 35    1        DECLARE BIT$PATTERN BYTE;
 36    1        DECLARE BIT$MASK BYTE;
```

```
                $EJECT

                /*  PREDEFINED SPECIAL SYMBOLS FOR HARDWARE REGISTERS
                    (NOTE:  VALUES OFFSET BY 80H TO CORRESPOND TO INDEX INTO
                            HARD$REG ARRAY(0-127).          */

 37    1        DECLARE P0   LITERALLY 'HARD$REG(00H)';
 38    1        DECLARE SP   LITERALLY 'HARD$REG(01H)';
 39    1        DECLARE DPL  LITERALLY 'HARD$REG(02H)';
 40    1        DECLARE DPH  LITERALLY 'HARD$REG(03H)';
 41    1        DECLARE TCON LITERALLY 'HARD$REG(08H)';
 42    1        DECLARE TMOD LITERALLY 'HARD$REG(09H)';
 43    1        DECLARE TL0  LITERALLY 'HARD$REG(0AH)';
 44    1        DECLARE TL1  LITERALLY 'HARD$REG(0BH)';
 45    1        DECLARE TH0  LITERALLY 'HARD$REG(0CH)';
 46    1        DECLARE TH1  LITERALLY 'HARD$REG(0DH)';
 47    1        DECLARE P1   LITERALLY 'HARD$REG(10H)';
 48    1        DECLARE SCON LITERALLY 'HARD$REG(18H)';
 49    1        DECLARE SBUF LITERALLY 'HARD$REG(19H)';
 50    1        DECLARE P2   LITERALLY 'HARD$REG(20H)';
 51    1        DECLARE IE   LITERALLY 'HARD$REG(28H)';
 52    1        DECLARE P3   LITERALLY 'HARD$REG(30H)';
 53    1        DECLARE IP   LITERALLY 'HARD$REG(38H)';
 54    1        DECLARE PSW  LITERALLY 'HARD$REG(50H)';
 55    1        DECLARE ACC  LITERALLY 'HARD$REG(60H)';
 56    1        DECLARE B    LITERALLY 'HARD$REG(70H)';



                /*  HARDWARE REGISTER TYPE CODES ARE ASSIGNED AS FOLLOWS:
                        0 - REGISTER UNDEFINED OR BEYOND SCOPE OF SIMULATOR;
                        1 - REGISTER WRITTEN OR READ SIMPLY BY DIRECT ADDRESSING;
                        2 - I/O PORT;
                        3 - . . .       (RESERVED FOR EXPANSION)      */

 57    1        DECLARE HARD$REG$ATTRIB(128) BYTE DATA
                       (2,1,1,1,  0,0,0,0,  1,1,1,1,  1,1,0,0,
                        2,0,0,0,  0,0,0,0,  1,2,0,0,  0,0,0,0,
                        2,0,0,0,  0,0,0,0,  1,0,0,0,  0,0,0,0,
                        2,0,0,0,  0,0,0,0,  1,0,0,0,  0,0,0,0,
                        0,0,0,0,  0,0,0,0,  0,0,0,0,  0,0,0,0,
                        1,0,0,0,  0,0,0,0,  0,0,0,0,  0,0,0,0,
                        1,0,0,0,  0,0,0,0,  0,0,0,0,  0,0,0,0,
                        1,0,0,0,  0,0,0,0,  0,0,0,0,  0,0,0,0);

 58    1        DECLARE BIT$REG$MAP(32) BYTE DATA
                       ( 20H, 21H, 22H, 23H,   24H, 25H, 26H, 27H,
                         28H, 29H, 2AH, 2BH,   2CH, 2DH, 2EH, 2FH,
                         80H, 88H, 90H, 98H,  0A0H,0A8H,0B0H,0B8H,
                        0C0H,0C8H,0D0H,0D8H,  0E0H,0E8H,0F0H,0F8H);

 59    1        DECLARE MASK$TABLE(8) BYTE DATA
                       (00000001B,  00000010B,  00000100B,  00001000B,
                        00010000B,  00100000B,  01000000B,  10000000B);
```

```
                $EJECT

                /*  HOOKS FOR EXTERNAL I/O PORT AND ERROR
                    HANDLING ROUTINES:                          */

 60    1        PORT$OUTPUT:  PROCEDURE(PORT$NO,PORT$DATA) EXTERNAL;
 61    2            DECLARE (PORT$NO,PORT$DATA) BYTE;
 62    2        END PORT$OUTPUT;

 63    1        PORT$INPUT:  PROCEDURE(PORT$NO) BYTE EXTERNAL;
 64    2            DECLARE PORT$NO BYTE;
 65    2        END PORT$INPUT;


 66    1        DIR$ADDR$ERR:  PROCEDURE(HARD$REG$CODE) EXTERNAL;
 67    2            DECLARE HARD$REG$CODE BYTE;
 68    2        END DIR$ADDR$ERR;


 69    1        IND$ADDR$ERR:  PROCEDURE(ILLEGAL$IND$ADDR) EXTERNAL;
 70    2            DECLARE ILLEGAL$IND$ADDR BYTE;
 71    2        END IND$ADDR$ERR;


 72    1        STACK$ERR:  PROCEDURE(ILLEGAL$STACK$ADDR) EXTERNAL;
 73    2            DECLARE ILLEGAL$STACK$ADDR BYTE;
 74    2        END STACK$ERR;


 75    1        FETCH$PROG$ERR:  PROCEDURE(ILLEGAL$CODE$ADDR) EXTERNAL;
 76    2            DECLARE ILLEGAL$CODE$ADDR ADDRESS;
 77    2        END FETCH$PROG$ERR;
```

```
                $EJECT

                /*  VARIOUS MEMORY SPACE ACCESS ROUTINES:       */

 78    1        FETCH$PROGRAM:  PROCEDURE (CODE$ADDR) BYTE;
 79    2            DECLARE (CODE$ADDR) ADDRESS;
 80    2            IF CODE$ADDR < ROM$SIZE
                        THEN RETURN USER$CODE(CODE$ADDR);
 82    2                  ELSE DO;
 83    3                      CALL FETCH$PROG$ERR(CODE$ADDR);
 84    3                      RETURN 00H;
 85    3                      END;
 86    2        END FETCH$PROGRAM;



 87    1        FETCH$REG:  PROCEDURE (REG$NO) BYTE;
 88    2            DECLARE (REG$NO,REG$ADDR) BYTE;
 89    2            REG$ADDR=(PSW AND 00011000B) + REG$NO;
 90    2            RETURN INT$RAM(REG$ADDR);
 91    2        END FETCH$REG;


 92    1        STORE$REG:  PROCEDURE (REG$NO,DATA$VALUE);
 93    2            DECLARE (REG$NO,REG$ADDR,DATA$VALUE) BYTE;
 94    2            REG$ADDR=(PSW AND 00011000B) + REG$NO;
 95    2            INT$RAM(REG$ADDR)=DATA$VALUE;
 96    2        END STORE$REG;



 97    1        FETCH$DIR:  PROCEDURE(DIR$ADDR$NO) BYTE;
 98    2        DECLARE(DIR$ADDR$NO,HARD$REG$INDEX,HARD$REG$TYPE) BYTE;
 99    2            IF DIR$ADDR$NO <= 7FH THEN
100    2                RETURN INT$RAM(DIR$ADDR$NO);
101    2            ELSE DO;
102    3                HARD$REG$INDEX=DIR$ADDR$NO - 80H;
103    3                HARD$REG$TYPE=HARD$REG$ATTRIB(HARD$REG$INDEX);
104    3                DO CASE HARD$REG$TYPE;
105    4                    DO;
106    5                        CALL DIR$ADDR$ERR(DIR$ADDR$NO);
107    5                        RETURN 00H;
108    5                        END;
109    4                    RETURN HARD$REG(HARD$REG$INDEX);
110    4                    RETURN PORT$INPUT(DIR$ADDR$NO);
111    4                    END;
112    3                END;
113    2        END FETCH$DIR;


114    1        FETCH$DIR$INT:  PROCEDURE(DIR$ADDR$NO) BYTE;
115    2        DECLARE(DIR$ADDR$NO,HARD$REG$INDEX,HARD$REG$TYPE) BYTE;
116    2            IF DIR$ADDR$NO <= 7FH THEN
117    2                RETURN INT$RAM(DIR$ADDR$NO);
118    2            ELSE DO;
```

```
119   3             HARD$REG$INDEX=DIR$ADDR$NO - 80H;
120   3             HARD$REG$TYPE=HARD$REG$ATTRIB(HARD$REG$INDEX);
121   3             DO CASE HARD$REG$TYPE;
122   4                 DO;
123   5                     CALL DIR$ADDR$ERR(DIR$ADDR$NO);
124   5                     RETURN 00H;
125   5                     END;
126   4                 RETURN HARD$REG(HARD$REG$INDEX);
127   4                 RETURN HARD$REG(HARD$REG$INDEX);
128   4                 END;
129   3             END;
130   2       END FETCH$DIR$INT;


131   1       STORE$DIR:  PROCEDURE(DIR$ADDR$NO, DATA$VALUE);
132   2           DECLARE(DIR$ADDR$NO, HARD$REG$INDEX, HARD$REG$TYPE, DATA$VALUE)
133   2           IF DIR$ADDR$NO <= 7FH THEN
134   2               INT$RAM(DIR$ADDR$NO)=DATA$VALUE;
135   2           ELSE DO;
136   3               HARD$REG$INDEX=DIR$ADDR$NO - 80H;
137   3               HARD$REG$TYPE=HARD$REG$ATTRIB(HARD$REG$INDEX);
138   3               DO CASE HARD$REG$TYPE;
139   4                   CALL DIR$ADDR$ERR(DIR$ADDR$NO);
140   4                   HARD$REG(HARD$REG$INDEX)=DATA$VALUE;
141   4                   DO;
142   5                       HARD$REG(HARD$REG$INDEX)=DATA$VALUE;
143   5                       CALL PORT$OUTPUT(DIR$ADDR$NO, DATA$VALUE);
144   5                       END;
145   4                   END;
146   3               END;
147   2       END STORE$DIR;



148   1       FETCH$IND:  PROCEDURE(REG$NO) BYTE;
149   2           DECLARE (REG$NO, REG$ADDR, RAM$ADDR) BYTE;
150   2           REG$ADDR=(PSW AND 00011000B) + REG$NO;
151   2           RAM$ADDR=INT$RAM(REG$ADDR);
152   2           IF RAM$ADDR <= 7FH
                     THEN RETURN INT$RAM(RAM$ADDR);
154   2               ELSE DO;
155   3                   CALL IND$ADDR$ERR(RAM$ADDR);
156   3                   RETURN 00H;
157   3                   END;
158   2       END FETCH$IND;


159   1       STORE$IND:  PROCEDURE(REG$NO, DATA$VALUE);
160   2           DECLARE (REG$NO, REG$ADDR, RAM$ADDR, DATA$VALUE) BYTE;
161   2           REG$ADDR=(PSW AND 00011000B) + REG$NO;
162   2           RAM$ADDR=INT$RAM(REG$ADDR);
163   2           IF RAM$ADDR <= 7FH
                     THEN INT$RAM(RAM$ADDR)=DATA$VALUE;
165   2               ELSE CALL IND$ADDR$ERR(RAM$ADDR);
166   2       END STORE$IND;
```

```
167   1       FETCH$BIT:  PROCEDURE(BIT$ADDR) BYTE;
168   2           DECLARE BIT$ADDR BYTE;
169   2           BIT$REG$ADDR=BIT$REG$MAP(BIT$ADDR / 8);
170   2           BIT$PATTERN=FETCH$DIR(BIT$REG$ADDR);
171   2           BIT$MASK=MASK$TABLE(BIT$ADDR AND 00000111B);
172   2           IF (BIT$PATTERN AND BIT$MASK) = 0
                      THEN RETURN 00H;
174   2                 ELSE RETURN 1;
175   2       END FETCH$BIT;


176   1       FETCH$BIT$INT:  PROCEDURE(BIT$ADDR) BYTE;
177   2           DECLARE BIT$ADDR BYTE;
178   2           BIT$REG$ADDR=BIT$REG$MAP(BIT$ADDR / 8);
179   2           BIT$PATTERN=FETCH$DIR$INT(BIT$REG$ADDR);
180   2           BIT$MASK=MASK$TABLE(BIT$ADDR AND 00000111B);
181   2           IF (BIT$PATTERN AND BIT$MASK) = 0
                      THEN RETURN 00H;
183   2                 ELSE RETURN 1;
184   2       END FETCH$BIT$INT;


185   1       STORE$BIT:  PROCEDURE(BIT$ADDR,BIT$DATA);
186   2           DECLARE (BIT$ADDR,BIT$DATA) BYTE;
187   2           BIT$REG$ADDR=BIT$REG$MAP(BIT$ADDR / 8);
188   2           BIT$PATTERN=FETCH$DIR$INT(BIT$REG$ADDR);
189   2           BIT$MASK=MASK$TABLE(BIT$ADDR AND 00000111B);
190   2           IF BIT$DATA = 0
                      THEN BIT$PATTERN=BIT$PATTERN AND (NOT BIT$MASK);
192   2                 ELSE BIT$PATTERN=(BIT$PATTERN OR BIT$MASK);
193   2           CALL STORE$DIR(BIT$REG$ADDR,BIT$PATTERN);
194   2       END STORE$BIT;


195   1       PUSH$STACK:  PROCEDURE (DATA$VALUE);
196   2           DECLARE (DATA$VALUE) BYTE;
197   2           SP=SP+1;
198   2           IF SP <= 7FH
                      THEN INT$RAM(SP)=DATA$VALUE;
200   2                 ELSE CALL STACK$ERR(SP);
201   2       END PUSH$STACK;


202   1       POP$STACK:  PROCEDURE BYTE;
203   2           DECLARE (DATA$VALUE) BYTE;
204   2           IF SP <= 7FH
                      THEN DATA$VALUE=INT$RAM(SP);
206   2                 ELSE DO;
207   3                      CALL  STACK$ERR(SP);
208   3                      DATA$VALUE=00H;
209   3                      END;
210   2           SP=SP-1;
211   2           RETURN DATA$VALUE;
212   2       END POP$STACK;
```

```
213   1       FETCH$PAGED$EXTERNAL:  PROCEDURE (PAGED$EXTERN$ADDR) BYTE;
214   2           DECLARE (PAGED$EXTERN$ADDR) BYTE;
215   2           RETURN EXTERN$RAM(PAGED$EXTERN$ADDR);
216   2       END FETCH$PAGED$EXTERNAL;

217   1       STORE$PAGED$EXTERNAL:  PROCEDURE (PAGED$EXTERN$ADDR,DATA$BYTE);
218   2           DECLARE (PAGED$EXTERN$ADDR) BYTE;
219   2           DECLARE DATA$BYTE BYTE;
220   2           EXTERN$RAM(PAGED$EXTERN$ADDR) = DATA$BYTE;
221   2       END STORE$PAGED$EXTERNAL;

222   1       FETCH$LONG$EXTERNAL:  PROCEDURE (LONG$EXTERN$ADDR) BYTE;
223   2           DECLARE (LONG$EXTERN$ADDR) ADDRESS;
224   2           RETURN EXTERN$RAM(LONG$EXTERN$ADDR);
225   2       END FETCH$LONG$EXTERNAL;

226   1       STORE$LONG$EXTERNAL:  PROCEDURE (LONG$EXTERN$ADDR,DATA$BYTE);
227   2           DECLARE (LONG$EXTERN$ADDR) ADDRESS;
228   2           DECLARE DATA$BYTE BYTE;
229   2           EXTERN$RAM(LONG$EXTERN$ADDR) = DATA$BYTE;
230   2       END STORE$LONG$EXTERNAL;

231   1       SIGN$EXTENDED:  PROCEDURE (SIGNED$BYTE) ADDRESS;
232   2           DECLARE SIGNED$BYTE BYTE;
233   2           IF (SIGNED$BYTE AND 10000000B) = 0
                    THEN RETURN SIGNED$BYTE;
235   2               ELSE RETURN (SIGNED$BYTE + 0FF00H);
236   2       END SIGN$EXTENDED;

237   1       PARITY$STATE:  PROCEDURE (DATA$BYTE) BYTE;
238   2           DECLARE (DATA$BYTE,PARITY$BIT) BYTE;
239   2           PARITY$BIT=0;
240   2           IF (DATA$BYTE AND 00000001B) <> 0
                    THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
242   2           IF (DATA$BYTE AND 00000010B) <> 0
                    THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
244   2           IF (DATA$BYTE AND 00000100B) <> 0
                    THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
246   2           IF (DATA$BYTE AND 00001000B) <> 0
                    THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
248   2           IF (DATA$BYTE AND 00010000B) <> 0
                    THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
250   2           IF (DATA$BYTE AND 00100000B) <> 0
                    THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
```

```
252   2           IF (DATA$BYTE AND 01000000B) <> 0
                      THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
254   2           IF (DATA$BYTE AND 10000000B) <> 0
                      THEN PARITY$BIT=PARITY$BIT XOR 00000001B;
256   2           RETURN PARITY$BIT;
257   2       END PARITY$STATE;
```

```
                $EJECT
                /*  THE FOLLOWING CODE PROVIDES A SINGLE ENTRY POINT FOR
                    AN EXTERNAL ROUTINE TO READ DATA FROM ALL SIMULATOR ADDRESS SP
                    THE FIRST CALLING PARAMETER GIVES UP TO 16 BITS OF ADDRESS;
                    THE SECOND SPECIFIES WHICH LOGICAL ADDRESS SPACE TO
                    READ, USING THE SCHEME:

                    0 = PROGRAM MEMORY
                    1 = WORKING REGISTER
                    2 = DIRECT ADDRESS (INPUTS FOR PORTS)
                    3 = INDIRECT THROUGH REGISTER SPECIFIED
                    4 = DIRECT BIT ADDRESS (DATA RIGHT-JUSTIFIED)
                    5 = PAGED EXTERNAL MEMORY
                    6 = 64K EXTERNAL MEMORY (ALL WRITES CURRENTLY TO PAGE 0)
                    7 = TOP-OF-STACK (SP UPDATED)

                    THE FUNCTION CALL RETURNS THE BYTE SO ADDRESSED.

 258    1       FETCH$SIM:  PROCEDURE (DATA$ADDR, DATA$TYPE) ADDRESS PUBLIC;
 259    2           DECLARE (RETURN$DATA, DATA$ADDR, DATA$TYPE) ADDRESS;
 260    2           DECLARE DATA$ADDR$BYTE BYTE;
 261    2           DATA$ADDR$BYTE=DATA$ADDR;
 262    2           DO CASE DATA$TYPE;
 263    3               RETURN$DATA=FETCH$PROGRAM(DATA$ADDR);
 264    3               RETURN$DATA=FETCH$REG(DATA$ADDR$BYTE);
 265    3               RETURN$DATA=FETCH$DIR$INT(DATA$ADDR$BYTE);
 266    3               RETURN$DATA=FETCH$IND(DATA$ADDR$BYTE);
 267    3               RETURN$DATA=FETCH$BIT$INT(DATA$ADDR$BYTE);
 268    3               RETURN$DATA=FETCH$PAGED$EXTERNAL(DATA$ADDR$BYTE);
 269    3               RETURN$DATA=FETCH$LONG$EXTERNAL(DATA$ADDR);
 270    3               RETURN$DATA=POP$STACK;
 271    3               END;
 272    2           RETURN RETURN$DATA;
 273    2       END FETCH$SIM;



                /*  THE FOLLOWING CODE PROVIDES A SINGLE ENTRY POINT FOR
                    AN EXTERNAL ROUTINE TO WRITE DATA INTO ALL SIMULATOR ADDRESS S
                    THE FIRST CALLING PARAMETER GIVES UP TO 16 BITS OF ADDRESS;
                    THE SECOND SPECIFIES WHICH LOGICAL ADDRESS SPACE TO
                    READ, USING THE SCHEME:

                    0 = PROGRAM MEMORY
                    1 = WORKING REGISTER
                    2 = DIRECT ADDRESS (OUTPUT LATCHES FOR PORTS)
                    3 = INDIRECT THROUGH REGISTER SPECIFIED
                    4 = DIRECT BIT ADDRESS (DATA RIGHT-JUSTIFIED)
                    5 = PAGED EXTERNAL MEMORY
                    6 = 64K EXTERNAL MEMORY (ALL WRITTEN CURRENTLY TO PAGE 0)
                    7 = TOP-OF-STACK (SP UPDATED)

                    THE THIRD PARAMETER HOLDS THE BYTE VALUE TO BE WRITTEN.     */

 274    1       STORE$SIM:  PROCEDURE (DATA$ADDR, DATA$TYPE, DATA$VALUE) PUBLIC;
 275    2           DECLARE (DATA$ADDR, DATA$TYPE, DATA$VALUE) ADDRESS;
 276    2           DECLARE (DATA$ADDR$BYTE, DATA$VALUE$BYTE) BYTE;
```

```
 277   2           DATA$ADDR$BYTE=DATA$ADDR;
 278   2           DATA$VALUE$BYTE=DATA$VALUE;
 279   2           DO CASE DATA$TYPE;
 280   3               USER$CODE(DATA$ADDR MOD ROM$SIZE)=DATA$VALUE$BYTE;
 281   3               CALL STORE$REG(DATA$ADDR$BYTE,DATA$VALUE$BYTE);
 282   3               CALL STORE$DIR(DATA$ADDR$BYTE,DATA$VALUE$BYTE);
 283   3               CALL STORE$IND(DATA$ADDR$BYTE,DATA$VALUE$BYTE);
 284   3               CALL STORE$BIT(DATA$ADDR$BYTE,DATA$VALUE$BYTE);
 285   3               CALL STORE$PAGED$EXTERNAL(DATA$ADDR$BYTE,DATA$VALUE$BYTE);
 286   3               CALL STORE$LONG$EXTERNAL(DATA$ADDR,DATA$VALUE$BYTE);
 287   3               CALL PUSH$STACK(DATA$VALUE$BYTE);
 288   3               END;
 289   2       END STORE$SIM;
```

```
                $EJECT
                $INCLUDE (ISET51.PLM)
            =   /*  INDIVIDUAL INSTRUCTION PROCEDURES:  */
            =
            =
            =   /*  "ACALL   addr16"  INSTRUCTION:  */
            =
290     1   =   ACALL$ADDR11:  PROCEDURE;
291     2   =       PAGE$CODE=(OPCODE AND 11100000B) / 32;
292     2   =       PAGE$OFFSET=FETCH$PROGRAM(PC+1);
293     2   =       PC=PC+2;
294     2   =       CALL PUSH$STACK(LOW(PC));
295     2   =       CALL PUSH$STACK(HIGH(PC));
296     2   =       PC=(PC AND 0F800H) + (PAGE$CODE * 100H) + PAGE$OFFSET;
297     2   =   END ACALL$ADDR11;
            =
            =
            =   /*  "ADD     A,<src-byte>"  FUNCTION:  */
            =
298     1   =   ADD$A:  PROCEDURE(DATA$BYTE);
299     2   =       DECLARE DATA$BYTE BYTE;
300     2   =       IF ((ACC AND 0FH)+(DATA$BYTE AND 0FH)) > 0FH
            =           THEN PSW=PSW OR  01000000B;
302     2   =           ELSE PSW=PSW AND 10111111B;
303     2   =       IF ((ACC AND 7FH)+(DATA$BYTE AND 7FH)) > 7FH
            =           THEN PSW=PSW OR  00000100B;
305     2   =           ELSE PSW=PSW AND 11111011B;
306     2   =       ADD$TEMP = (ACC);
307     2   =       ADD$TEMP = (ADD$TEMP+DATA$BYTE);
308     2   =       IF ADD$TEMP > 0FFH
            =           THEN PSW=(PSW OR  10000000B) XOR 00000100B;
310     2   =           ELSE PSW=(PSW AND 01111111B);
311     2   =       ACC=LOW(ADD$TEMP);
312     2   =   END ADD$A;
            =
            =
            =   /*  "ADD     A,Rn"  INSTRUCTION:  */
            =
313     1   =   ADD$A$REG:  PROCEDURE;
314     2   =       REG$ADDR=OPCODE AND 00000111B;
315     2   =       REG$DATA=FETCH$REG(REG$ADDR);
316     2   =       CALL ADD$A(REG$DATA);
317     2   =       PC=PC+1;
318     2   =   END ADD$A$REG;
            =
            =
            =   /*  "ADD     A,direct"  INSTRUCTION:  */
            =
319     1   =   ADD$A$DIR:  PROCEDURE;
320     2   =       DIR$ADDR=FETCH$PROGRAM(PC+1);
321     2   =       DIR$DATA=FETCH$DIR(DIR$ADDR);
322     2   =       CALL ADD$A(DIR$DATA);
323     2   =       PC=PC+2;
324     2   =   END ADD$A$DIR;
            =
            =
            =   /*  "ADD     A,@Ri"  INSTRUCTION:  */
```

```
                =
325     1       =   ADD$A$IND:  PROCEDURE;
326     2       =       REG$ADDR=OPCODE AND 00000001B;
327     2       =       IND$DATA=FETCH$IND(REG$ADDR);
328     2       =       CALL ADD$A(IND$DATA);
329     2       =       PC=PC+1;
330     2       =   END ADD$A$IND;
                =
                =
                =   /*  "ADD     A,#data"  INSTRUCTION:  */
                =
331     1       =   ADD$A$IMM:  PROCEDURE;
332     2       =       IMM$DATA=FETCH$PROGRAM(PC+1);
333     2       =       CALL ADD$A(IMM$DATA);
334     2       =       PC=PC+2;
335     2       =   END ADD$A$IMM;
                =
                =
                =   /*  "ADDC    A,<src-byte>"  FUNCTION:  */
                =
336     1       =   ADDC$A:  PROCEDURE(DATA$BYTE);
337     2       =       DECLARE DATA$BYTE BYTE;
338     2       =       LINK$BIT = (PSW AND 10000000B) / 128;
339     2       =       IF ((ACC AND 0FH)+(DATA$BYTE AND 0FH)+LINK$BIT) > 0FH
                =           THEN PSW=PSW OR  01000000B;
341     2       =           ELSE PSW=PSW AND 10111111B;
342     2       =       IF ((ACC AND 7FH)+(DATA$BYTE AND 7FH)+LINK$BIT) > 7FH
                =           THEN PSW=PSW OR  00000100B;
344     2       =           ELSE PSW=PSW AND 11111011B;
345     2       =       ADD$TEMP = (ACC);
346     2       =       ADD$TEMP = (ADD$TEMP+DATA$BYTE);
347     2       =       ADD$TEMP = (ADD$TEMP+LINK$BIT);
348     2       =       IF ADD$TEMP > 0FFH
                =           THEN PSW=(PSW OR  10000000B) XOR 00000100B;
350     2       =           ELSE PSW=(PSW AND 01111111B);
351     2       =       ACC=LOW(ADD$TEMP);
352     2       =   END ADDC$A;
                =
                =
                =   /*  "ADDC    A,Rn"  INSTRUCTION:  */
                =
353     1       =   ADDC$A$REG:  PROCEDURE;
354     2       =       REG$ADDR=OPCODE AND 00000111B;
355     2       =       REG$DATA=FETCH$REG(REG$ADDR);
356     2       =       CALL ADDC$A(REG$DATA);
357     2       =       PC=PC+1;
358     2       =   END ADDC$A$REG;
                =
                =
                =   /*  "ADDC    A,direct"  INSTRUCTION:  */
                =
359     1       =   ADDC$A$DIR:  PROCEDURE;
360     2       =       DIR$ADDR=FETCH$PROGRAM(PC+1);
361     2       =       DIR$DATA=FETCH$DIR(DIR$ADDR);
362     2       =       CALL ADDC$A(DIR$DATA);
363     2       =       PC=PC+2;
364     2       =   END ADDC$A$DIR;
```

```
                =
                =
                =   /*  "ADDC    A,@Ri"  INSTRUCTION:  */
                =
 365    1       =   ADDC$A$IND:  PROCEDURE;
 366    2       =       REG$ADDR=OPCODE AND 00000001B;
 367    2       =       IND$DATA=FETCH$IND(REG$ADDR);
 368    2       =       CALL ADDC$A(IND$DATA);
 369    2       =       PC=PC+1;
 370    2       =   END ADDC$A$IND;
                =
                =
                =   /*  "ADDC    A,#data"  INSTRUCTION:  */
                =
 371    1       =   ADDC$A$IMM:  PROCEDURE;
 372    2       =       IMM$DATA=FETCH$PROGRAM(PC+1);
 373    2       =       CALL ADDC$A(IMM$DATA);
 374    2       =       PC=PC+2;
 375    2       =   END ADDC$A$IMM;
                =
                =
                =   /*  "AJMP    addr11"  INSTRUCTION:  */
                =
 376    1       =   AJMP$ADDR11:  PROCEDURE;
 377    2       =       PAGE$CODE=(OPCODE AND 11100000B) / 32;
 378    2       =       PAGE$OFFSET=FETCH$PROGRAM(PC+1);
 379    2       =       PC=PC+2;
 380    2       =       PC=(PC AND 0F800H) + (PAGE$CODE * 100H) + PAGE$OFFSET;
 381    2       =   END AJMP$ADDR11;
                =
                =
                =   /*  "ANL     A,Rn"  INSTRUCTION:  */
                =
 382    1       =   ANL$A$REG:  PROCEDURE;
 383    2       =       REG$ADDR=OPCODE AND 00000111B;
 384    2       =       REG$DATA=FETCH$REG(REG$ADDR);
 385    2       =       ACC=ACC AND REG$DATA;
 386    2       =       PC=PC+1;
 387    2       =   END ANL$A$REG;
                =
                =
                =   /*  "ANL     A,direct"  INSTRUCTION:  */
                =
 388    1       =   ANL$A$DIR:  PROCEDURE;
 389    2       =       DIR$ADDR=FETCH$PROGRAM(PC+1);
 390    2       =       DIR$DATA=FETCH$DIR(DIR$ADDR);
 391    2       =       ACC=ACC AND DIR$DATA;
 392    2       =       PC=PC+2;
 393    2       =   END ANL$A$DIR;
                =
                =
                =   /*  "ANL     A,@Ri"  INSTRUCTION:  */
                =
 394    1       =   ANL$A$IND:  PROCEDURE;
 395    2       =       REG$ADDR=OPCODE AND 00000001B;
 396    2       =       IND$DATA=FETCH$IND(REG$ADDR);
 397    2       =       ACC=ACC AND IND$DATA;
```

```
398   2  =        PC=PC+1;
399   2  =    END ANL$A$IND;
           =
           =
           =    /*  "ANL      A,#data"  INSTRUCTION:   */
           =
400   1  =    ANL$A$IMM:  PROCEDURE;
401   2  =        IMM$DATA=FETCH$PROGRAM(PC+1);
402   2  =        ACC=ACC AND IMM$DATA;
403   2  =        PC=PC+2;
404   2  =    END ANL$A$IMM;
           =
           =
           =    /*  "ANL      direct,A"  INSTRUCTION:   */
           =
405   1  =    ANL$DIR$A:  PROCEDURE;
406   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
407   2  =        DIR$DATA=FETCH$DIR$INT(DIR$ADDR),
408   2  =        CALL STORE$DIR(DIR$ADDR,ACC AND DIR$DATA);
409   2  =        PC=PC+2;
410   2  =    END ANL$DIR$A;
           =
           =
           =    /*  "ANL      direct,#data"  INSTRUCTION:   */
           =
411   1  =    ANL$DIR$IMM:  PROCEDURE;
412   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
413   2  =        IMM$DATA=FETCH$PROGRAM(PC+2);
414   2  =        DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
415   2  =        CALL STORE$DIR(DIR$ADDR,IMM$DATA AND DIR$DATA);
416   2  =        PC=PC+3;
417   2  =    END ANL$DIR$IMM;
           =
           =
           =    /*  "ANL      C,bit"  INSTRUCTION:   */
           =
418   1  =    ANL$C$BIT:  PROCEDURE;
419   2  =        BIT$ADDR=FETCH$PROGRAM(PC+1);
420   2  =        BIT$DATA=FETCH$BIT(BIT$ADDR);
421   2  =        IF BIT$DATA = 0 THEN PSW=PSW AND 01111111B;
423   2  =        PC=PC+2;
424   2  =    END ANL$C$BIT;
           =
           =
           =    /*  "ANL      C,/bit"  INSTRUCTION:   */
           =
425   1  =    ANL$C$COMP$BIT:  PROCEDURE;
426   2  =        BIT$ADDR=FETCH$PROGRAM(PC+1);
427   2  =        BIT$DATA=FETCH$BIT(BIT$ADDR);
428   2  =        IF BIT$DATA = 1 THEN PSW=PSW AND 01111111B;
430   2  =        PC=PC+2;
431   2  =    END ANL$C$COMP$BIT;
           =
           =
           =    /*  "CJNE     A,direct,rel"  INSTRUCTION:   */
           =
432   1  =    CJNE$A$DIR$REL:  PROCEDURE;
```

```
433   2  =          DIR$ADDR=FETCH$PROGRAM(PC+1);
434   2  =          DIR$DATA=FETCH$DIR(DIR$ADDR);
435   2  =          DISPLACEMENT=FETCH$PROGRAM(PC+2);
436   2  =          IF ACC < DIR$DATA
      =              THEN PSW=(PSW OR  10000000B);
438   2  =              ELSE PSW=(PSW AND 01111111B);
439   2  =          PC=PC+3;
440   2  =          IF ACC <> DIR$DATA
      =              THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
442   2  =      END CJNE$A$DIR$REL;
      =
      =
      =      /*  "CJNE     A,#data,rel"  INSTRUCTION:   */
      =
443   1  =      CJNE$A$IMM$REL:  PROCEDURE;
444   2  =          IMM$DATA=FETCH$PROGRAM(PC+1);
445   2  =          DISPLACEMENT=FETCH$PROGRAM(PC+2);
446   2  =          IF ACC < IMM$DATA
      =              THEN PSW=(PSW OR  10000000B);
448   2  =              ELSE PSW=(PSW AND 01111111B);
449   2  =          PC=PC+3;
450   2  =          IF ACC <> IMM$DATA
      =              THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
452   2  =      END CJNE$A$IMM$REL;
      =
      =
      =      /*  "CJNE     Rn,#data,rel"  INSTRUCTION:   */
      =
453   1  =      CJNE$REG$IMM$REL:  PROCEDURE;
454   2  =          REG$ADDR=OPCODE AND 00000111B;
455   2  =          REG$DATA=FETCH$REG(REG$ADDR);
456   2  =          IMM$DATA=FETCH$PROGRAM(PC+1);
457   2  =          DISPLACEMENT=FETCH$PROGRAM(PC+2);
458   2  =          IF REG$DATA < IMM$DATA
      =              THEN PSW=(PSW OR  10000000B);
460   2  =              ELSE PSW=(PSW AND 01111111B);
461   2  =          PC=PC+3;
462   2  =          IF REG$DATA <> IMM$DATA
      =              THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
464   2  =      END CJNE$REG$IMM$REL;
      =
      =
      =      /*  "CJNE     @Ri,#data,rel"  INSTRUCTION:   */
      =
465   1  =      CJNE$IND$IMM$REL:  PROCEDURE;
466   2  =          REG$ADDR=OPCODE AND 00000001B;
467   2  =          IND$DATA=FETCH$IND(REG$ADDR);
468   2  =          IMM$DATA=FETCH$PROGRAM(PC+1);
469   2  =          DISPLACEMENT=FETCH$PROGRAM(PC+2);
470   2  =          IF IND$DATA < IMM$DATA
      =              THEN PSW=(PSW OR  10000000B);
472   2  =              ELSE PSW=(PSW AND 01111111B);
473   2  =          PC=PC+3;
474   2  =          IF IND$DATA <> IMM$DATA
      =              THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
476   2  =      END CJNE$IND$IMM$REL;
      =
```

```
                =
                =   /*  "CLR     A"  INSTRUCTION:  */
                =
477     1       =   CLR$A:  PROCEDURE;
478     2       =       ACC=0;
479     2       =       PC=PC+1;
480     2       =   END CLR$A;
                =
                =
                =   /*  "CLR     C"  INSTRUCTION:  */
                =
481     1       =   CLR$C:  PROCEDURE;
482     2       =       PSW=PSW AND 01111111B;
483     2       =       PC=PC+1;
484     2       =   END CLR$C;
                =
                =
                =   /*  "CLR     bit"  INSTRUCTION:  */
                =
485     1       =   CLR$BIT:  PROCEDURE;
486     2       =       BIT$ADDR=FETCH$PROGRAM(PC+1);
487     2       =       CALL STORE$BIT(BIT$ADDR,0);
488     2       =       PC=PC+2;
489     2       =   END CLR$BIT;
                =
                =
                =   /*  "CPL     A"  INSTRUCTION:  */
                =
490     1       =   CPL$A:  PROCEDURE;
491     2       =       ACC=ACC XOR 11111111B;
492     2       =       PC=PC+1;
493     2       =   END CPL$A;
                =
                =
                =   /*  "CPL     C"  INSTRUCTION:  */
                =
494     1       =   CPL$C:  PROCEDURE;
495     2       =       PSW=PSW XOR 10000000B;
496     2       =       PC=PC+1;
497     2       =   END CPL$C;
                =
                =
                =   /*  "CPL     bit"  INSTRUCTION:  */
                =
498     1       =   CPL$BIT:  PROCEDURE;
499     2       =       BIT$ADDR=FETCH$PROGRAM(PC+1);
500     2       =       BIT$DATA=FETCH$BIT$INT(BIT$ADDR);
501     2       =       BIT$DATA=BIT$DATA XOR 00000001B;
502     2       =       CALL STORE$BIT(BIT$ADDR,BIT$DATA);
503     2       =       PC=PC+2;
504     2       =   END CPL$BIT;
                =
                =
                =   /*  "DA      A"  INSTRUCTION:  */
                =
505     1       =   DA$A:  PROCEDURE;
506     2       =       IF ((ACC AND 0FH) > 09H) OR ((PSW AND 01000000B) <> 0)
```

```
          =             THEN DO;
508    3  =                 IF ACC >= OFAH THEN PSW=PSW OR 10000000B;
511    3  =                 ACC=ACC+6;
512    3  =                 END;
513    2  =         IF ((ACC AND OFOH) > 90H) OR ((PSW AND 10000000B) <> 0)
          =             THEN DO;
515    3  =                 IF ACC >= OAOH THEN PSW=PSW OR 10000000B;
          =                 ACC=ACC+60H;
518    3  =                 END;
519    2  =         PC=PC+1;
520    2  =     END DA$A;
          =
          =
          =     /*  "DEC      A"  INSTRUCTION:  */
          =
521    1  =     DEC$A:  PROCEDURE;
522    2  =         ACC=ACC-1;
523    2  =         PC=PC+1;
524    2  =     END DEC$A;
          =
          =
          =     /*  "DEC      Rn"  INSTRUCTION:  */
          =
525    1  =     DEC$REG:  PROCEDURE;
526    2  =         REG$ADDR=OPCODE AND 00000111B;
527    2  =         REG$DATA=FETCH$REG(REG$ADDR);
528    2  =         REG$DATA=REG$DATA-1;
529    2  =         CALL STORE$REG(REG$ADDR,REG$DATA);
530    2  =         PC=PC+1;
531    2  =     END DEC$REG;
          =
          =
          =     /*  "DEC      direct"  INSTRUCTION:  */
          =
532    1  =     DEC$DIR:  PROCEDURE;
533    2  =         DIR$ADDR=FETCH$PROGRAM(PC+1);
534    2  =         DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
535    2  =         DIR$DATA=DIR$DATA-1;
536    2  =         CALL STORE$DIR(DIR$ADDR,DIR$DATA);
537    2  =         PC=PC+2;
538    2  =     END DEC$DIR;
          =
          =
          =     /*  "DEC      @Ri"  INSTRUCTION:  */
          =
539    1  =     DEC$IND:  PROCEDURE;
540    2  =         REG$ADDR=OPCODE AND 00000001B;
541    2  =         IND$DATA=FETCH$IND(REG$ADDR);
542    2  =         IND$DATA=IND$DATA-1;
543    2  =         CALL STORE$IND(REG$ADDR,IND$DATA);
544    2  =         PC=PC+1;
545    2  =     END DEC$IND;
          =
          =
          =     /*  "DIV      AB"  INSTRUCTION:  */
          =
```

```
546   1   =   DIV$AB:  PROCEDURE;
547   2   =       IF B = 0 THEN PSW=PSW OR 00000100B;
549   2   =           ELSE DO;
550   3   =               PSW=PSW AND 11111011B;
551   3   =               DIV$TEMP=ACC / B;
552   3   =               B=ACC MOD B;
553   3   =               ACC=DIV$TEMP;
554   3   =               END;
555   2   =       PSW=PSW AND 01111111B;
556   2   =       PC=PC+1;
557   2   =   END DIV$AB;
              =
              =
              =   /*  "DJNZ    Rn,rel"  INSTRUCTION:  */
              =
558   1   =   DJNZ$REG$REL:  PROCEDURE;
559   2   =       REG$ADDR=OPCODE AND 00000111B;
560   2   =       DISPLACEMENT=FETCH$PROGRAM(PC+1);
561   2   =       REG$DATA=FETCH$REG(REG$ADDR);
562   2   =       REG$DATA=REG$DATA-1;
563   2   =       CALL STORE$REG(REG$ADDR,REG$DATA);
564   2   =       PC=PC+2;
565   2   =       IF REG$DATA <> 0
              =           THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
567   2   =   END DJNZ$REG$REL;
              =
              =
              =   /*  "DJNZ    direct,rel"  INSTRUCTION:  */
              =
568   1   =   DJNZ$DIR$REL:  PROCEDURE;
569   2   =       DIR$ADDR=FETCH$PROGRAM(PC+1);
570   2   =       DISPLACEMENT=FETCH$PROGRAM(PC+2);
571   2   =       DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
572   2   =       DIR$DATA=DIR$DATA-1;
573   2   =       CALL STORE$DIR(DIR$ADDR,DIR$DATA);
574   2   =       PC=PC+3;
575   2   =       IF DIR$DATA <> 0
              =           THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
577   2   =   END DJNZ$DIR$REL;
              =
              =
              =   /*  "INC     A"  INSTRUCTION:  */
              =
578   1   =   INC$A:  PROCEDURE;
579   2   =       ACC=ACC+1;
580   2   =       PC=PC+1;
581   2   =   END INC$A;
              =
              =
              =   /*  "INC     Rn"  INSTRUCTION:  */
              =
582   1   =   INC$REG:  PROCEDURE;
583   2   =       REG$ADDR=OPCODE AND 00000111B;
584   2   =       REG$DATA=FETCH$REG(REG$ADDR);
585   2   =       REG$DATA=REG$DATA+1;
586   2   =       CALL STORE$REG(REG$ADDR,REG$DATA);
587   2   =       PC=PC+1;
```

```
588   2  =    END INC$REG;
           =
           =
           =    /*  "INC      direct"  INSTRUCTION:  */
           =
589   1  =    INC$DIR:  PROCEDURE;
590   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
591   2  =        DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
592   2  =        DIR$DATA=DIR$DATA+1;
593   2  =        CALL STORE$DIR(DIR$ADDR,DIR$DATA);
594   2  =        PC=PC+2;
595   2  =    END INC$DIR;
           =
           =
           =    /*  "INC      @Ri"  INSTRUCTION:  */
           =
596   1  =    INC$IND:  PROCEDURE;
597   2  =        REG$ADDR=OPCODE AND 00000001B;
598   2  =        IND$DATA=FETCH$IND(REG$ADDR);
599   2  =        IND$DATA=IND$DATA+1;
600   2  =        CALL STORE$IND(REG$ADDR,IND$DATA);
601   2  =        PC=PC+1;
602   2  =    END INC$IND;
           =
           =
           =    /*  "INC      DPTR"  INSTRUCTION:  */
           =
603   1  =    INC$DPTR:  PROCEDURE;
604   2  =        DPL=DPL+1;
605   2  =        IF DPL=0 THEN DPH=DPH+1;
607   2  =        PC=PC+1;
608   2  =    END INC$DPTR;
           =
           =
           =    /*  "JB       bit,rel"  INSTRUCTION:  */
           =
609   1  =    JB$BIT$REL:  PROCEDURE;
610   2  =        BIT$ADDR=FETCH$PROGRAM(PC+1);
611   2  =        BIT$DATA=FETCH$BIT(BIT$ADDR);
612   2  =        DISPLACEMENT=FETCH$PROGRAM(PC+2);
613   2  =        PC=PC+3;
614   2  =        IF BIT$DATA=1
           =            THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
616   2  =    END JB$BIT$REL;
           =
           =
           =    /*  "JBC      bit,rel"  INSTRUCTION:  */
           =
617   1  =    JBC$BIT$REL:  PROCEDURE;
618   2  =        BIT$ADDR=FETCH$PROGRAM(PC+1);
619   2  =        BIT$DATA=FETCH$BIT$INT(BIT$ADDR);
620   2  =        DISPLACEMENT=FETCH$PROGRAM(PC+2);
621   2  =        PC=PC+3;
622   2  =        CALL STORE$BIT(BIT$ADDR,0);
623   2  =        IF BIT$DATA=1
           =            THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
625   2  =    END JBC$BIT$REL;
```

```
                =
                =
                =   /*  "JC      rel"  INSTRUCTION:  */
                =
626     1       =   JC$REL:  PROCEDURE;
627     2       =       DISPLACEMENT=FETCH$PROGRAM(PC+1);
628     2       =       PC=PC+2;
629     2       =       IF (PSW AND 10000000B) <> 0
                =           THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
631     2       =   END JC$REL;
                =
                =
                =   /*  "JMP     @A+DPTR"  INSTRUCTION:  */
                =
632     1       =   JMP$ADPTR:  PROCEDURE;
633     2       =       CODE$ADDR=(DPH*256)+DPL+ACC;
634     2       =       PC=CODE$ADDR;
635     2       =   END JMP$ADPTR;
                =
                =
                =   /*  "JNB     bit,rel"  INSTRUCTION:  */
                =
636     1       =   JNB$BIT$REL:  PROCEDURE;
637     2       =       BIT$ADDR=FETCH$PROGRAM(PC+1);
638     2       =       DISPLACEMENT=FETCH$PROGRAM(PC+2);
639     2       =       BIT$DATA=FETCH$BIT(BIT$ADDR);
640     2       =       PC=PC+3;
641     2       =       IF BIT$DATA=0
                =           THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
643     2       =   END JNB$BIT$REL;
                =
                =
                =   /*  "JNC     rel"  INSTRUCTION:  */
                =
644     1       =   JNC$REL:  PROCEDURE;
645     2       =       DISPLACEMENT=FETCH$PROGRAM(PC+1);
646     2       =       PC=PC+2;
647     2       =       IF (PSW AND 10000000B) = 0
                =           THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
649     2       =   END JNC$REL;
                =
                =
                =   /*  "JNZ     rel"  INSTRUCTION:  */
                =
650     1       =   JNZ$REL:  PROCEDURE;
651     2       =       DISPLACEMENT=FETCH$PROGRAM(PC+1);
652     2       =       PC=PC+2;
653     2       =       IF ACC <> 0
                =           THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
655     2       =   END JNZ$REL;
                =
                =
                =   /*  "JZ      rel"  INSTRUCTION:  */
                =
656     1       =   JZ$REL:  PROCEDURE;
657     2       =       DISPLACEMENT=FETCH$PROGRAM(PC+1);
658     2       =       PC=PC+2;
```

```
659   2  =          IF ACC = 0
           =              THEN PC=PC+SIGN$EXTENDED(DISPLACEMENT);
661   2  =      END JZ$REL;
           =
           =
           =      /*  "LCALL    addr16"  INSTRUCTION:   */
           =
662   1  =      LCALL$ADDR16:  PROCEDURE;
663   2  =          PAGE$CODE=FETCH$PROGRAM(PC+1);
664   2  =          PAGE$OFFSET=FETCH$PROGRAM(PC+2);
665   2  =          PC=PC+3;
666   2  =          CALL PUSH$STACK(LOW(PC));
667   2  =          CALL PUSH$STACK(HIGH(PC));
668   2  =          PC=(PAGE$CODE * 100H) + PAGE$OFFSET;
669   2  =      END LCALL$ADDR16;
           =
           =
           =      /*  "LJMP     addr16"  INSTRUCTION:   */
           =
670   1  =      LJMP$ADDR16:  PROCEDURE;
671   2  =          PAGE$CODE=FETCH$PROGRAM(PC+1);
672   2  =          PAGE$OFFSET=FETCH$PROGRAM(PC+2);
673   2  =          PC=(PAGE$CODE * 100H) + PAGE$OFFSET;
674   2  =      END LJMP$ADDR16;
           =
           =
           =      /*  "MOV      A,Rn"  INSTRUCTION:   */
           =
675   1  =      MOV$A$REG:  PROCEDURE;
676   2  =          REG$ADDR=OPCODE AND 00000111B;
677   2  =          ACC=FETCH$REG(REG$ADDR);
678   2  =          PC=PC+1;
679   2  =      END MOV$A$REG;
           =
           =
           =      /*  "MOV      A,direct"  INSTRUCTION:   */
           =
680   1  =      MOV$A$DIR:  PROCEDURE;
681   2  =          DIR$ADDR=FETCH$PROGRAM(PC+1);
682   2  =          ACC=FETCH$DIR(DIR$ADDR);
683   2  =          PC=PC+2;
684   2  =      END MOV$A$DIR;
           =
           =
           =      /*  "MOV      A,@Ri"  INSTRUCTION:   */
           =
685   1  =      MOV$A$IND:  PROCEDURE;
686   2  =          REG$ADDR=OPCODE AND 00000001B;
687   2  =          ACC=FETCH$IND(REG$ADDR);
688   2  =          PC=PC+1;
689   2  =      END MOV$A$IND;
           =
           =
           =      /*  "MOV      A,#data"  INSTRUCTION:   */
           =
690   1  =      MOV$A$IMM:  PROCEDURE;
691   2  =          ACC=FETCH$PROGRAM(PC+1);
```

```
692   2  =        PC=PC+2;
693   2  =    END MOV$A$IMM;
             =
             =
             =    /*  "MOV      Rn,A"  INSTRUCTION:  */
             =
694   1  =    MOV$REG$A:  PROCEDURE;
695   2  =        REG$ADDR=OPCODE AND 00000111B;
696   2  =        CALL STORE$REG(REG$ADDR,ACC);
697   2  =        PC=PC+1;
698   2  =    END MOV$REG$A;
             =
             =
             =    /*  "MOV      Rn,direct"  INSTRUCTION:  */
             =
699   1  =    MOV$REG$DIR:  PROCEDURE;
700   2  =        REG$ADDR=OPCODE AND 00000111B;
701   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
702   2  =        DIR$DATA=FETCH$DIR(DIR$ADDR);
703   2  =        CALL STORE$REG(REG$ADDR,DIR$DATA);
704   2  =        PC=PC+2;
705   2  =    END MOV$REG$DIR;
             =
             =
             =    /*  "MOV      Rn,#data"  INSTRUCTION:  */
             =
706   1  =    MOV$REG$IMM:  PROCEDURE;
707   2  =        REG$ADDR=OPCODE AND 00000111B;
708   2  =        IMM$DATA=FETCH$PROGRAM(PC+1);
709   2  =        CALL STORE$REG(REG$ADDR,IMM$DATA);
710   2  =        PC=PC+2;
711   2  =    END MOV$REG$IMM;
             =
             =
             =    /*  "MOV      direct,A"  INSTRUCTION:  */
             =
712   1  =    MOV$DIR$A:  PROCEDURE;
713   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
714   2  =        CALL STORE$DIR(DIR$ADDR,ACC);
715   2  =        PC=PC+2;
716   2  =    END MOV$DIR$A;
             =
             =
             =    /*  "MOV      direct,Rn"  INSTRUCTION:  */
             =
717   1  =    MOV$DIR$REG:  PROCEDURE;
718   2  =        REG$ADDR=OPCODE AND 00000111B;
719   2  =        REG$DATA=FETCH$REG(REG$ADDR);
720   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
721   2  =        CALL STORE$DIR(DIR$ADDR,REG$DATA);
722   2  =        PC=PC+2;
723   2  =    END MOV$DIR$REG;
             =
             =
             =    /*  "MOV      direct,direct"  INSTRUCTION:  */
             =
724   1  =    MOV$DIR$DIR:  PROCEDURE;
```

```
725   2  =        SOURCE$ADDR=FETCH$PROGRAM(PC+1);
726   2  =        DEST$ADDR=FETCH$PROGRAM(PC+2);
727   2  =        DIR$DATA=FETCH$DIR(SOURCE$ADDR);
728   2  =        CALL STORE$DIR(DEST$ADDR,DIR$DATA);
729   2  =        PC=PC+3;
730   2  =    END MOV$DIR$DIR;
             =
             =
             =    /*  "MOV      direct,@Ri"  INSTRUCTION:   */
             =
731   1  =    MOV$DIR$IND:  PROCEDURE;
732   2  =        REG$ADDR=OPCODE AND 00000001B;
733   2  =        IND$DATA=FETCH$IND(REG$ADDR);
734   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
735   2  =        CALL STORE$DIR(DIR$ADDR,IND$DATA);
736   2  =        PC=PC+2;
737   2  =    END MOV$DIR$IND;
             =
             =
             =    /*  "MOV      direct,#data"  INSTRUCTION:   */
             =
738   1  =    MOV$DIR$IMM:  PROCEDURE;
739   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
740   2  =        IMM$DATA=FETCH$PROGRAM(PC+2);
741   2  =        CALL STORE$DIR(DIR$ADDR,IMM$DATA);
742   2  =        PC=PC+3;
743   2  =    END MOV$DIR$IMM;
             =
             =
             =    /*  "MOV      @Ri,A"  INSTRUCTION:   */
             =
744   1  =    MOV$IND$A:  PROCEDURE;
745   2  =        REG$ADDR=OPCODE AND 00000001B;
746   2  =        CALL STORE$IND(REG$ADDR,ACC);
747   2  =        PC=PC+1;
748   2  =    END MOV$IND$A;
             =
             =
             =    /*  "MOV      @Ri,direct"  INSTRUCTION:   */
             =
749   1  =    MOV$IND$DIR:  PROCEDURE;
750   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
751   2  =        DIR$DATA=FETCH$DIR(DIR$ADDR);
752   2  =        REG$ADDR=OPCODE AND 00000001B;
753   2  =        CALL STORE$IND(REG$ADDR,DIR$DATA);
754   2  =        PC=PC+2;
755   2  =    END MOV$IND$DIR;
             =
             =
             =    /*  "MOV      @Ri,#data"  INSTRUCTION:   */
             =
756   1  =    MOV$IND$IMM:  PROCEDURE;
757   2  =        IMM$DATA=FETCH$PROGRAM(PC+1);
758   2  =        REG$ADDR=OPCODE AND 00000001B;
759   2  =        CALL STORE$IND(REG$ADDR,IMM$DATA);
760   2  =        PC=PC+2;
761   2  =    END MOV$IND$IMM;
```

```
                =
                =
                =   /*  "MOV      C,bit"  INSTRUCTION:   */
                =
 762    1       =   MOV$C$BIT:  PROCEDURE;
 763    2       =       BIT$ADDR=FETCH$PROGRAM(PC+1);
 764    2       =       BIT$DATA=FETCH$BIT(BIT$ADDR);
 765    2       =       IF BIT$DATA=0
                =           THEN PSW=(PSW AND 01111111B);
 767    2       =           ELSE PSW=(PSW OR  10000000B);
 768    2       =       PC=PC+2;
 769    2       =   END MOV$C$BIT;
                =
                =
                =   /*  "MOV      bit,C"  INSTRUCTION:   */
                =
 770    1       =   MOV$BIT$C:  PROCEDURE;
 771    2       =       BIT$ADDR=FETCH$PROGRAM(PC+1);
 772    2       =       BIT$DATA=((PSW AND 10000000B) / 128);
 773    2       =       CALL STORE$BIT(BIT$ADDR,BIT$DATA);
 774    2       =       PC=PC+2;
 775    2       =   END MOV$BIT$C;
                =
                =
                =   /*  "MOV      DPTR,#data16"  INSTRUCTION:   */
                =
 776    1       =   MOV$DPTR$IMM16:  PROCEDURE;
 777    2       =       DPH=FETCH$PROGRAM(PC+1);
 778    2       =       DPL=FETCH$PROGRAM(PC+2);
 779    2       =       PC=PC+3;
 780    2       =   END MOV$DPTR$IMM16;
                =
                =
                =   /*  "MOVC     A,@A+DPTR"  INSTRUCTION:   */
                =
 781    1       =   MOVC$A$ADPTR:  PROCEDURE;
 782    2       =       CODE$ADDR=(DPH*256)+DPL+ACC;
 783    2       =       ACC=FETCH$PROGRAM(CODE$ADDR);
 784    2       =       PC=PC+1;
 785    2       =   END MOVC$A$ADPTR;
                =
                =
                =   /*  "MOVC     A,@A+PC"  INSTRUCTION:   */
                =
 786    1       =   MOVC$A$APC:  PROCEDURE;
 787    2       =       PC=PC+1;
 788    2       =       CODE$ADDR=(PC+ACC);
 789    2       =       ACC=FETCH$PROGRAM(CODE$ADDR);
 790    2       =   END MOVC$A$APC;
                =
                =
                =   /*  "MOVX     A,@Ri"  INSTRUCTION:   */
                =
 791    1       =   MOVX$A$IND:  PROCEDURE;
 792    2       =       REG$ADDR=OPCODE AND 00000001B;
 793    2       =       PAGED$EXTERNAL$ADDR=FETCH$REG(REG$ADDR);
 794    2       =       ACC=FETCH$PAGED$EXTERNAL(PAGED$EXTERNAL$ADDR);
```

```
795   2  =        PC=PC+1;
796   2  =    END MOVX$A$IND;
           =
           =
           =    /*  "MOVX     @Ri,A"  INSTRUCTION:   */
           =
797   1  =    MOVX$IND$A:  PROCEDURE;
798   2  =        REG$ADDR=OPCODE AND 00000001B;
799   2  =        PAGED$EXTERNAL$ADDR=FETCH$REG(REG$ADDR);
800   2  =        CALL STORE$PAGED$EXTERNAL(PAGED$EXTERNAL$ADDR,ACC);
801   2  =        PC=PC+1;
802   2  =    END MOVX$IND$A;
           =
           =
           =    /*  "MOVX     A,@DPTR"  INSTRUCTION:   */
           =
803   1  =    MOVX$A$DPTR:  PROCEDURE;
804   2  =        ACC=FETCH$LONG$EXTERNAL((DPH*256)+DPL);
805   2  =        PC=PC+1;
806   2  =    END MOVX$A$DPTR;
           =
           =
           =    /*  "MOVX     @DPTR,A"  INSTRUCTION:   */
           =
807   1  =    MOVX$DPTR$A:  PROCEDURE;
808   2  =        CALL STORE$LONG$EXTERNAL((DPH*256)+DPL,ACC);
809   2  =        PC=PC+1;
810   2  =    END MOVX$DPTR$A;
           =
           =
           =    /*  "MUL      AB"  INSTRUCTION:   */
           =
811   1  =    MUL$AB:  PROCEDURE;
812   2  =        MUL$TEMP=ACC * B;
813   2  =        B=HIGH(MUL$TEMP);
814   2  =        ACC=LOW(MUL$TEMP);
815   2  =        PSW=PSW AND 01111111B;
816   2  =        IF B = 0
           =            THEN PSW=PSW AND 11111011B;
818   2  =            ELSE PSW=PSW OR  00000100B;
819   2  =        PC=PC+1;
820   2  =    END MUL$AB;
           =
           =
           =    /*  "NOP "  INSTRUCTION:   */
           =
821   1  =    NOP:  PROCEDURE;
822   2  =        PC=PC+1;
823   2  =    END NOP;
           =
           =
           =    /*  "ORL      A,Rn"  INSTRUCTION:   */
           =
824   1  =    ORL$A$REG:  PROCEDURE;
825   2  =        REG$ADDR=OPCODE AND 00000111B;
826   2  =        REG$DATA=FETCH$REG(REG$ADDR);
827   2  =        ACC=ACC OR REG$DATA,
```

```
828   2  =        PC=PC+1;
829   2  =    END ORL$A$REG;
           =
           =
           =    /*  "ORL      A,direct"  INSTRUCTION:  */
           =
830   1  =    ORL$A$DIR:  PROCEDURE;
831   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
832   2  =        DIR$DATA=FETCH$DIR(DIR$ADDR);
833   2  =        ACC=ACC OR DIR$DATA;
834   2  =        PC=PC+2;
835   2  =    END ORL$A$DIR;
           =
           =
           =    /*  "ORL      A,@Ri"  INSTRUCTION:  */
           =
836   1  =    ORL$A$IND:  PROCEDURE;
837   2  =        REG$ADDR=OPCODE AND 00000001B;
838   2  =        IND$DATA=FETCH$IND(REG$ADDR);
839   2  =        ACC=ACC OR IND$DATA;
840   2  =        PC=PC+1;
841   2  =    END ORL$A$IND;
           =
           =
           =    /*  "ORL      A,#data"  INSTRUCTION:  */
           =
842   1  =    ORL$A$IMM:  PROCEDURE;
843   2  =        IMM$DATA=FETCH$PROGRAM(PC+1);
844   2  =        ACC=ACC OR IMM$DATA;
845   2  =        PC=PC+2;
846   2  =    END ORL$A$IMM;
           =
           =
           =    /*  "ORL      direct,A"  INSTRUCTION:  */
           =
847   1  =    ORL$DIR$A:  PROCEDURE;
848   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
849   2  =        DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
850   2  =        CALL STORE$DIR(DIR$ADDR,ACC OR DIR$DATA);
851   2  =        PC=PC+2;
852   2  =    END ORL$DIR$A;
           =
           =
           =    /*  "ORL      direct,#data"  INSTRUCTION:  */
           =
853   1  =    ORL$DIR$IMM:  PROCEDURE;
854   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
855   2  =        IMM$DATA=FETCH$PROGRAM(PC+2);
856   2  =        DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
857   2  =        CALL STORE$DIR(DIR$ADDR,IMM$DATA OR DIR$DATA);
858   2  =        PC=PC+3;
859   2  =    END ORL$DIR$IMM;
           =
           =
           =    /*  "ORL      C,bit"  INSTRUCTION:  */
           =
860   1  =    ORL$C$BIT:  PROCEDURE;
```

```
861   2  =        BIT$ADDR=FETCH$PROGRAM(PC+1);
862   2  =        BIT$DATA=FETCH$BIT(BIT$ADDR);
863   2  =        IF BIT$DATA = 1 THEN PSW=PSW OR 10000000B;
865   2  =        PC=PC+2;
866   2  =    END ORL$C$BIT;
          =
          =
          =    /*  "ORL      C,/bit"  INSTRUCTION:   */
          =
867   1  =    ORL$C$COMP$BIT:  PROCEDURE;
868   2  =        BIT$ADDR=FETCH$PROGRAM(PC+1);
869   2  =        BIT$DATA=FETCH$BIT(BIT$ADDR);
870   2  =        IF BIT$DATA = 0 THEN PSW=PSW OR 10000000B;
872   2  =        PC=PC+2;
873   2  =    END ORL$C$COMP$BIT;
          =
          =
          =    /*  "POP      direct"  INSTRUCTION:   */
          =
874   1  =    POP$DIR:  PROCEDURE;
875   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
876   2  =        STACK$DATA=POP$STACK;
877   2  =        CALL STORE$DIR(DIR$ADDR,STACK$DATA);
878   2  =        PC=PC+2;
879   2  =    END POP$DIR;
          =
          =
          =    /*  "PUSH     direct"  INSTRUCTION:   */
          =
880   1  =    PUSH$DIR:  PROCEDURE;
881   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
882   2  =        DIR$DATA=FETCH$DIR(DIR$ADDR);
883   2  =        CALL PUSH$STACK(DIR$DATA);
884   2  =        PC=PC+2;
885   2  =    END PUSH$DIR;
          =
          =
          =    /*  "RET "  INSTRUCTION:   */
          =
886   1  =    RET:  PROCEDURE;
887   2  =        PAGE$CODE=POP$STACK;
888   2  =        PAGE$OFFSET=POP$STACK;
889   2  =        PC=(PAGE$CODE * 100H) + PAGE$OFFSET;
890   2  =    END RET;
          =
          =
          =    /*  "RETI"  INSTRUCTION:   */
          =
891   1  =    RETI:  PROCEDURE;
892   2  =        PAGE$CODE=POP$STACK;
893   2  =        PAGE$OFFSET=POP$STACK;
894   2  =        PC=(PAGE$CODE * 100H) + PAGE$OFFSET;
          =
          =    /*  RESTORE INTERRUPT SYSTEM TO LEVEL IN EFFECT
          =        BEFORE LAST INTERRUPT RECEIVED  */
          =
895   2  =    END RETI;
```

```
                =
                =
                =   /*  "RL      A"  INSTRUCTION:  */
                =
 896    1       =   RL$A:  PROCEDURE;
 897    2       =       LINK$BIT=(ACC AND 10000000B) / 128;
 898    2       =       ACC=(ACC * 2) + LINK$BIT;
 899    2       =       PC=PC+1;
 900    2       =   END RL$A;
                =
                =
                =   /*  "RLC     A"  INSTRUCTION:  */
                =
 901    1       =   RLC$A:  PROCEDURE;
 902    2       =       LINK$BIT=(ACC AND 10000000B) / 128;
 903    2       =       ACC=(ACC * 2) + ((PSW AND 10000000B) / 128);
 904    2       =       PSW=(PSW AND 01111111B) + (LINK$BIT * 128);
 905    2       =       PC=PC+1;
 906    2       =   END RLC$A;
                =
                =
                =   /*  "RR      A"  INSTRUCTION:  */
                =
 907    1       =   RR$A:  PROCEDURE;
 908    2       =       LINK$BIT=ACC AND 00000001B;
 909    2       =       ACC=(ACC / 2) + (LINK$BIT * 128);
 910    2       =       PC=PC+1;
 911    2       =   END RR$A;
                =
                =
                =   /*  "RRC     A"  INSTRUCTION:  */
                =
 912    1       =   RRC$A:  PROCEDURE;
 913    2       =       LINK$BIT=ACC AND 00000001B;
 914    2       =       ACC=(ACC / 2) + (PSW AND 10000000B);
 915    2       =       PSW=(PSW AND 01111111B)+(LINK$BIT * 128);
 916    2       =       PC=PC+1;
 917    2       =   END RRC$A;
                =
                =
                =   /*  "SETB    C"  INSTRUCTION:  */
                =
 918    1       =   SETB$C:  PROCEDURE;
 919    2       =       PSW=PSW OR 10000000B;
 920    2       =       PC=PC+1;
 921    2       =   END SETB$C;
                =
                =
                =   /*  "SETB    bit"  INSTRUCTION:  */
                =
 922    1       =   SETB$BIT:  PROCEDURE;
 923    2       =       BIT$ADDR=FETCH$PROGRAM(PC+1);
 924    2       =       CALL STORE$BIT(BIT$ADDR,1);
 925    2       =       PC=PC+2;
 926    2       =   END SETB$BIT;
                =
                =
```

```
                =   /*  "SJMP    rel"  INSTRUCTION:  */
                =
 927    1       =   SJMP$REL:  PROCEDURE;
 928    2       =       DISPLACEMENT=FETCH$PROGRAM(PC+1);
 929    2       =       PC=PC+2;
 930    2       =       PC=PC+SIGN$EXTENDED(DISPLACEMENT);
 931    2       =   END SJMP$REL;
                =
                =
                =   /*  "SUBB    A,<src-byte>"  FUNCTION:  */
                =
 932    1       =   SUBB$A:  PROCEDURE(DATA$BYTE);
 933    2       =       DECLARE DATA$BYTE BYTE;
 934    2       =       LINK$BIT=(PSW AND 10000000B) / 128;
 935    2       =       SUB$TEMP=DATA$BYTE;
 936    2       =       SUB$TEMP=SUB$TEMP+LINK$BIT;
 937    2       =       IF (ACC AND 0FH) < (SUB$TEMP AND 0FH)
                =           THEN PSW=PSW OR  01000000B;
 939    2       =           ELSE PSW=PSW AND 10111111B;
 940    2       =       IF (ACC AND 7FH) < (SUB$TEMP AND 7FH)
                =           THEN PSW=PSW OR  00000100B;
 942    2       =           ELSE PSW=PSW AND 11111011B;
 943    2       =       IF ACC < SUB$TEMP
                =           THEN PSW=(PSW OR  10000000B) XOR 00000100B;
 945    2       =           ELSE PSW=(PSW AND 01111111B);
 946    2       =       ACC=ACC-SUB$TEMP;
 947    2       =   END SUBB$A;
                =
                =
                =   /*  "SUBB    A,Rn"  INSTRUCTION:  */
                =
 948    1       =   SUBB$A$REG:  PROCEDURE;
 949    2       =       REG$ADDR=OPCODE AND 00000111B;
 950    2       =       REG$DATA=FETCH$REG(REG$ADDR);
 951    2       =       CALL SUBB$A(REG$DATA);
 952    2       =       PC=PC+1;
 953    2       =   END SUBB$A$REG;
                =
                =
                =   /*  "SUBB    A,direct"  INSTRUCTION:  */
                =
 954    1       =   SUBB$A$DIR:  PROCEDURE;
 955    2       =       DIR$ADDR=FETCH$PROGRAM(PC+1);
 956    2       =       DIR$DATA=FETCH$DIR(DIR$ADDR);
 957    2       =       CALL SUBB$A(DIR$DATA);
 958    2       =       PC=PC+2;
 959    2       =   END SUBB$A$DIR;
                =
                =
                =   /*  "SUBB    A,@Ri"  INSTRUCTION:  */
                =
 960    1       =   SUBB$A$IND:  PROCEDURE;
 961    2       =       REG$ADDR=OPCODE AND 00000001B;
 962    2       =       IND$DATA=FETCH$IND(REG$ADDR);
 963    2       =       CALL SUBB$A(IND$DATA);
 964    2       =       PC=PC+1;
 965    2       =   END SUBB$A$IND;
```

```
                =
                =
                =   /*  "SUBB    A,#data"  INSTRUCTION:  */
                =
 966    1       =   SUBB$A$IMM:  PROCEDURE;
 967    2       =       IMM$DATA=FETCH$PROGRAM(PC+1);
 968    2       =       CALL SUBB$A(IMM$DATA);
 969    2       =       PC=PC+2;
 970    2       =   END SUBB$A$IMM;
                =
                =
                =   /*  "SWAP    A"  INSTRUCTION:  */
                =
 971    1       =   SWAP$A:  PROCEDURE;
 972    2       =       LOW$NIB=ACC AND 00001111B;
 973    2       =       HIGH$NIB=ACC AND 11110000B;
 974    2       =       ACC=(LOW$NIB * 16) + (HIGH$NIB / 16);
 975    2       =       PC=PC+1;
 976    2       =   END SWAP$A;
                =
                =
                =   /*  "XCH     A,Rn"  INSTRUCTION:  */
                =
 977    1       =   XCH$A$REG:  PROCEDURE;
 978    2       =       REG$ADDR=OPCODE AND 00000111B;
 979    2       =       REG$DATA=FETCH$REG(REG$ADDR);
 980    2       =       CALL STORE$REG(REG$ADDR,ACC);
 981    2       =       ACC=REG$DATA;
 982    2       =       PC=PC+1;
 983    2       =   END XCH$A$REG;
                =
                =
                =   /*  "XCH     A,direct"  INSTRUCTION:  */
                =
 984    1       =   XCH$A$DIR:  PROCEDURE;
 985    2       =       DIR$ADDR=FETCH$PROGRAM(PC+1);
 986    2       =       DIR$DATA=FETCH$DIR(DIR$ADDR);
 987    2       =       CALL STORE$DIR(DIR$ADDR,ACC);
 988    2       =       ACC=DIR$DATA;
 989    2       =       PC=PC+2;
 990    2       =   END XCH$A$DIR;
                =
                =
                =   /*  "XCH     A,@Ri"  INSTRUCTION:  */
                =
 991    1       =   XCH$A$IND:  PROCEDURE;
 992    2       =       REG$ADDR=OPCODE AND 00000001B;
 993    2       =       IND$DATA=FETCH$IND(REG$ADDR);
 994    2       =       CALL STORE$IND(REG$ADDR,ACC);
 995    2       =       ACC=IND$DATA;
 996    2       =       PC=PC+1;
 997    2       =   END XCH$A$IND;
                =
                =
                =   /*  "XCHD    A,@Ri"  INSTRUCTION:  */
                =
 998    1       =   XCHD$A$IND:  PROCEDURE;
```

```
 999   2  =        REG$ADDR=OPCODE AND 00000001B;
1000   2  =        IND$DATA=FETCH$IND(REG$ADDR);
1001   2  =        LOW$SOURCE$NIB=IND$DATA AND 00001111B;
1002   2  =        IND$DATA=(IND$DATA AND 11110000B) + (ACC AND 00001111B);
1003   2  =        CALL STORE$IND(REG$ADDR, IND$DATA);
1004   2  =        ACC=(ACC AND 11110000B) + LOW$SOURCE$NIB;
1005   2  =        PC=PC+1;
1006   2  =    END XCHD$A$IND;
           =
           =
           =    /*  "XRL      A,Rn"  INSTRUCTION:   */
           =
1007   1  =    XRL$A$REG:  PROCEDURE;
1008   2  =        REG$ADDR=OPCODE AND 00000111B;
1009   2  =        REG$DATA=FETCH$REG(REG$ADDR);
1010   2  =        ACC=ACC XOR REG$DATA;
1011   2  =        PC=PC+1;
1012   2  =    END XRL$A$REG;
           =
           =
           =    /*  "XRL      A,direct"  INSTRUCTION:   */
           =
1013   1  =    XRL$A$DIR:  PROCEDURE;
1014   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
1015   2  =        DIR$DATA=FETCH$DIR(DIR$ADDR);
1016   2  =        ACC=ACC XOR DIR$DATA;
1017   2  =        PC=PC+2;
1018   2  =    END XRL$A$DIR;
           =
           =
           =    /*  "XRL      A,@Ri"  INSTRUCTION:   */
           =
1019   1  =    XRL$A$IND:  PROCEDURE;
1020   2  =        REG$ADDR=OPCODE AND 00000001B;
1021   2  =        IND$DATA=FETCH$IND(REG$ADDR);
1022   2  =        ACC=ACC XOR IND$DATA;
1023   2  =        PC=PC+1;
1024   2  =    END XRL$A$IND;
           =
           =
           =    /*  "XRL      A,#data"  INSTRUCTION:   */
           =
1025   1  =    XRL$A$IMM:  PROCEDURE;
1026   2  =        IMM$DATA=FETCH$PROGRAM(PC+1);
1027   2  =        ACC=ACC XOR IMM$DATA;
1028   2  =        PC=PC+2;
1029   2  =    END XRL$A$IMM;
           =
           =
           =    /*  "XRL      direct,A"  INSTRUCTION:   */
           =
1030   1  =    XRL$DIR$A:  PROCEDURE;
1031   2  =        DIR$ADDR=FETCH$PROGRAM(PC+1);
1032   2  =        DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
1033   2  =        CALL STORE$DIR(DIR$ADDR, ACC XOR DIR$DATA);
1034   2  =        PC=PC+2;
1035   2  =    END XRL$DIR$A;
```

```
                =
                =
                =   /*  "XRL      direct,#data"  INSTRUCTION:   */
                =
1036    1       =   XRL$DIR$IMM:  PROCEDURE;
1037    2       =       DIR$ADDR=FETCH$PROGRAM(PC+1);
1038    2       =       IMM$DATA=FETCH$PROGRAM(PC+2);
1039    2       =       DIR$DATA=FETCH$DIR$INT(DIR$ADDR);
1040    2       =       CALL STORE$DIR(DIR$ADDR, IMM$DATA XOR DIR$DATA);
1041    2       =       PC=PC+3;
1042    2       =   END XRL$DIR$IMM;
                =
```

```
                $EJECT

1043    1       INITIALIZE: PROCEDURE PUBLIC;
1044    2           PC=0000H;
1045    2           P0=0FFH;
1046    2           P1=0FFH;
1047    2           P2=0FFH;
1048    2           P3=0FFH;
1049    2           PSW=00H;
1050    2           SP=07H;
1051    2           DPL=00H;
1052    2           DPH=00H;
1053    2           ACC=00H;
1054    2           B=00H;
1055    2           TL0=00H;
1056    2           TH0=00H;
1057    2           TL1=00H;
1058    2           TH1=00H;
1059    2           TCON=00H;
1060    2           TMOD=00H;
1061    2           SCON=00H;
1062    2           IE=00H;
1063    2           IP=00H;
1064    2           MACH$CYC=0000H;   /*  SIMULATION ELAPSED TIME REGISTER.  */
1065    2       END INITIALIZE;


1066    1       DECLARE EXECUTION$TIME(256) BYTE DATA
                      (1, 2, 2, 1, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 2, 1, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 2, 1, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 2, 1, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 1, 2, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 1, 2, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 1, 2, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 2, 2, 1, 2, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 2, 2, 4, 2, 2, 2,    2, 2, 2, 2, 2, 2, 2, 2,
                       2, 2, 2, 2, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 1, 2, 4, 1, 2, 2,    2, 2, 2, 2, 2, 2, 2, 2,
                       2, 2, 1, 1, 2, 2, 2, 2,    2, 2, 2, 2, 2, 2, 2, 2,
                       2, 2, 1, 1, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 1, 1, 1, 2, 1, 1,    2, 2, 2, 2, 2, 2, 2, 2,
                       2, 2, 2, 2, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1,
                       2, 2, 2, 2, 1, 1, 1, 1,    1, 1, 1, 1, 1, 1, 1, 1);
```

```
                $EJECT

1067    1       STEP:  PROCEDURE (NEXT$INSTRUCTION) ADDRESS PUBLIC;
1068    2           DECLARE NEXT$INSTRUCTION ADDRESS;
1069    2           PC=NEXT$INSTRUCTION;
1070    2           OPCODE=USER$CODE(PC);
1071    2           DO CASE OPCODE;

                    /*  INSTRUCTIONS CORRESPONDING TO ROW 0 OF OPCODE MAP
                        (FORM 0XH):  */
1072    3                   CALL NOP;
1073    3                   CALL AJMP$ADDR11;
1074    3                   CALL LJMP$ADDR16;
1075    3                   CALL RR$A;
1076    3                   CALL INC$A;
1077    3                   CALL INC$DIR;
1078    3                   CALL INC$IND;
1079    3                   CALL INC$IND;
1080    3                   CALL INC$REG;
1081    3                   CALL INC$REG;
1082    3                   CALL INC$REG;
1083    3                   CALL INC$REG;
1084    3                   CALL INC$REG;
1085    3                   CALL INC$REG;
1086    3                   CALL INC$REG;
1087    3                   CALL INC$REG;


                    /*  INSTRUCTIONS CORRESPONDING TO ROW 1 OF OPCODE MAP
                        (FORM 1XH):  */
1088    3                   CALL JBC$BIT$REL;
1089    3                   CALL ACALL$ADDR11;
1090    3                   CALL LCALL$ADDR16;
1091    3                   CALL RRC$A;
1092    3                   CALL DEC$A;
1093    3                   CALL DEC$DIR;
1094    3                   CALL DEC$IND;
1095    3                   CALL DEC$IND;
1096    3                   CALL DEC$REG;
1097    3                   CALL DEC$REG;
1098    3                   CALL DEC$REG;
1099    3                   CALL DEC$REG;
1100    3                   CALL DEC$REG;
1101    3                   CALL DEC$REG;
1102    3                   CALL DEC$REG;
1103    3                   CALL DEC$REG;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW 2 OF OPCODE MAP
                    (FORM 2XH):  */
1104    3               CALL JB$BIT$REL;
1105    3               CALL AJMP$ADDR11;
1106    3               CALL RET;
1107    3               CALL RL$A;
1108    3               CALL ADD$A$IMM;
1109    3               CALL ADD$A$DIR;
1110    3               CALL ADD$A$IND;
1111    3               CALL ADD$A$IND;
1112    3               CALL ADD$A$REG;
1113    3               CALL ADD$A$REG;
1114    3               CALL ADD$A$REG;
1115    3               CALL ADD$A$REG;
1116    3               CALL ADD$A$REG;
1117    3               CALL ADD$A$REG;
1118    3               CALL ADD$A$REG;
1119    3               CALL ADD$A$REG;


                /*  INSTRUCTIONS CORRESPONDING TO ROW 3 OF OPCODE MAP
                    (FORM 3XH):  */
1120    3               CALL JNB$BIT$REL;
1121    3               CALL ACALL$ADDR11;
1122    3               CALL RETI;
1123    3               CALL RLC$A;
1124    3               CALL ADDC$A$IMM;
1125    3               CALL ADDC$A$DIR;
1126    3               CALL ADDC$A$IND;
1127    3               CALL ADDC$A$IND;
1128    3               CALL ADDC$A$REG;
1129    3               CALL ADDC$A$REG;
1130    3               CALL ADDC$A$REG;
1131    3               CALL ADDC$A$REG;
1132    3               CALL ADDC$A$REG;
1133    3               CALL ADDC$A$REG;
1134    3               CALL ADDC$A$REG;
1135    3               CALL ADDC$A$REG;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW 4 OF OPCODE MAP
                    (FORM 4XH):  */
1136    3               CALL JC$REL;
1137    3               CALL AJMP$ADDR11;
1138    3               CALL ORL$DIR$A;
1139    3               CALL ORL$DIR$IMM;
1140    3               CALL ORL$A$IMM;
1141    3               CALL ORL$A$DIR;
1142    3               CALL ORL$A$IND;
1143    3               CALL ORL$A$IND;
1144    3               CALL ORL$A$REG;
1145    3               CALL ORL$A$REG;
1146    3               CALL ORL$A$REG;
1147    3               CALL ORL$A$REG;
1148    3               CALL ORL$A$REG;
1149    3               CALL ORL$A$REG;
1150    3               CALL ORL$A$REG;
1151    3               CALL ORL$A$REG;


                /*  INSTRUCTIONS CORRESPONDING TO ROW 5 OF OPCODE MAP
                    (FORM 5XH):  */
1152    3               CALL JNC$REL;
1153    3               CALL ACALL$ADDR11;
1154    3               CALL ANL$DIR$A;
1155    3               CALL ANL$DIR$IMM;
1156    3               CALL ANL$A$IMM;
1157    3               CALL ANL$A$DIR;
1158    3               CALL ANL$A$IND;
1159    3               CALL ANL$A$IND;
1160    3               CALL ANL$A$REG;
1161    3               CALL ANL$A$REG;
1162    3               CALL ANL$A$REG;
1163    3               CALL ANL$A$REG;
1164    3               CALL ANL$A$REG;
1165    3               CALL ANL$A$REG;
1166    3               CALL ANL$A$REG;
1167    3               CALL ANL$A$REG;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW 6 OF OPCODE MAP
                    (FORM 6XH):  */
1168    3               CALL JZ$REL;
1169    3               CALL AJMP$ADDR11;
1170    3               CALL XRL$DIR$A;
1171    3               CALL XRL$DIR$IMM;
1172    3               CALL XRL$A$IMM;
1173    3               CALL XRL$A$DIR;
1174    3               CALL XRL$A$IND;
1175    3               CALL XRL$A$IND;
1176    3               CALL XRL$A$REG;
1177    3               CALL XRL$A$REG;
1178    3               CALL XRL$A$REG;
1179    3               CALL XRL$A$REG;
1180    3               CALL XRL$A$REG;
1181    3               CALL XRL$A$REG;
1182    3               CALL XRL$A$REG;
1183    3               CALL XRL$A$REG;


                /*  INSTRUCTIONS CORRESPONDING TO ROW 7 OF OPCODE MAP
                    (FORM 7XH):  */
1184    3               CALL JNZ$REL;
1185    3               CALL ACALL$ADDR11;
1186    3               CALL ORL$C$BIT;
1187    3               CALL JMP$ADPTR;
1188    3               CALL MOV$A$IMM;
1189    3               CALL MOV$DIR$IMM;
1190    3               CALL MOV$IND$IMM;
1191    3               CALL MOV$IND$IMM;
1192    3               CALL MOV$REG$IMM;
1193    3               CALL MOV$REG$IMM;
1194    3               CALL MOV$REG$IMM;
1195    3               CALL MOV$REG$IMM;
1196    3               CALL MOV$REG$IMM;
1197    3               CALL MOV$REG$IMM;
1198    3               CALL MOV$REG$IMM;
1199    3               CALL MOV$REG$IMM;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW 8 OF OPCODE MAP
                    (FORM 8XH):  */
1200    3               CALL SJMP$REL;
1201    3               CALL AJMP$ADDR11;
1202    3               CALL ANL$C$BIT;
1203    3               CALL MOVC$A$APC;
1204    3               CALL DIV$AB;
1205    3               CALL MOV$DIR$DIR;
1206    3               CALL MOV$DIR$IND;
1207    3               CALL MOV$DIR$IND;
1208    3               CALL MOV$DIR$REG;
1209    3               CALL MOV$DIR$REG;
1210    3               CALL MOV$DIR$REG;
1211    3               CALL MOV$DIR$REG;
1212    3               CALL MOV$DIR$REG;
1213    3               CALL MOV$DIR$REG;
1214    3               CALL MOV$DIR$REG;
1215    3               CALL MOV$DIR$REG;


                /*  INSTRUCTIONS CORRESPONDING TO ROW 9 OF OPCODE MAP
                    (FORM 9XH):  */
1216    3               CALL MOV$DPTR$IMM16;
1217    3               CALL ACALL$ADDR11;
1218    3               CALL MOV$BIT$C;
1219    3               CALL MOVC$A$ADPTR;
1220    3               CALL SUBB$A$IMM;
1221    3               CALL SUBB$A$DIR;
1222    3               CALL SUBB$A$IND;
1223    3               CALL SUBB$A$IND;
1224    3               CALL SUBB$A$REG;
1225    3               CALL SUBB$A$REG;
1226    3               CALL SUBB$A$REG;
1227    3               CALL SUBB$A$REG;
1228    3               CALL SUBB$A$REG;
1229    3               CALL SUBB$A$REG;
1230    3               CALL SUBB$A$REG;
1231    3               CALL SUBB$A$REG;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW A OF OPCODE MAP
                    (FORM AXH):  */
1232    3               CALL ORL$C$COMP$BIT;
1233    3               CALL AJMP$ADDR11;
1234    3               CALL MOV$C$BIT;
1235    3               CALL INC$DPTR;
1236    3               CALL MUL$AB;
1237    3               CALL NOP;
1238    3               CALL MOV$IND$DIR;
1239    3               CALL MOV$IND$DIR;
1240    3               CALL MOV$REG$DIR;
1241    3               CALL MOV$REG$DIR;
1242    3               CALL MOV$REG$DIR;
1243    3               CALL MOV$REG$DIR;
1244    3               CALL MOV$REG$DIR;
1245    3               CALL MOV$REG$DIR;
1246    3               CALL MOV$REG$DIR;
1247    3               CALL MOV$REG$DIR;


                /*  INSTRUCTIONS CORRESPONDING TO ROW B OF OPCODE MAP
                    (FORM BXH):  */
1248    3               CALL ANL$C$COMP$BIT;
1249    3               CALL ACALL$ADDR11;
1250    3               CALL CPL$BIT;
1251    3               CALL CPL$C;
1252    3               CALL CJNE$A$IMM$REL;
1253    3               CALL CJNE$A$DIR$REL;
1254    3               CALL CJNE$IND$IMM$REL;
1255    3               CALL CJNE$IND$IMM$REL;
1256    3               CALL CJNE$REG$IMM$REL;
1257    3               CALL CJNE$REG$IMM$REL;
1258    3               CALL CJNE$REG$IMM$REL;
1259    3               CALL CJNE$REG$IMM$REL;
1260    3               CALL CJNE$REG$IMM$REL;
1261    3               CALL CJNE$REG$IMM$REL;
1262    3               CALL CJNE$REG$IMM$REL;
1263    3               CALL CJNE$REG$IMM$REL;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW C OF OPCODE MAP
                    (FORM CXH):  */
1264    3               CALL PUSH$DIR;
1265    3               CALL AJMP$ADDR11;
1266    3               CALL CLR$BIT;
1267    3               CALL CLR$C;
1268    3               CALL SWAP$A;
1269    3               CALL XCH$A$DIR;
1270    3               CALL XCH$A$IND;
1271    3               CALL XCH$A$IND;
1272    3               CALL XCH$A$REG;
1273    3               CALL XCH$A$REG;
1274    3               CALL XCH$A$REG;
1275    3               CALL XCH$A$REG;
1276    3               CALL XCH$A$REG;
1277    3               CALL XCH$A$REG;
1278    3               CALL XCH$A$REG;
1279    3               CALL XCH$A$REG;


                /*  INSTRUCTIONS CORRESPONDING TO ROW D OF OPCODE MAP
                    (FORM DXH):  */
1280    3               CALL POP$DIR;
1281    3               CALL ACALL$ADDR11;
1282    3               CALL SETB$BIT;
1283    3               CALL SETB$C;
1284    3               CALL DA$A;
1285    3               CALL DJNZ$DIR$REL;
1286    3               CALL XCHD$A$IND;
1287    3               CALL XCHD$A$IND;
1288    3               CALL DJNZ$REG$REL;
1289    3               CALL DJNZ$REG$REL;
1290    3               CALL DJNZ$REG$REL;
1291    3               CALL DJNZ$REG$REL;
1292    3               CALL DJNZ$REG$REL;
1293    3               CALL DJNZ$REG$REL;
1294    3               CALL DJNZ$REG$REL;
1295    3               CALL DJNZ$REG$REL;
```

```
                $EJECT


                /*  INSTRUCTIONS CORRESPONDING TO ROW E OF OPCODE MAP
                    (FORM EXH):  */
1296    3               CALL MOVX$A$DPTR;
1297    3               CALL AJMP$ADDR11;
1298    3               CALL MOVX$A$IND;
1299    3               CALL MOVX$A$IND;
1300    3               CALL CLR$A;
1301    3               CALL MOV$A$DIR;
1302    3               CALL MOV$A$IND;
1303    3               CALL MOV$A$IND;
1304    3               CALL MOV$A$REG;
1305    3               CALL MOV$A$REG;
1306    3               CALL MOV$A$REG;
1307    3               CALL MOV$A$REG;
1308    3               CALL MOV$A$REG;
1309    3               CALL MOV$A$REG;
1310    3               CALL MOV$A$REG;
1311    3               CALL MOV$A$REG;


                /*  INSTRUCTIONS CORRESPONDING TO ROW F OF OPCODE MAP
                    (FORM FXH):  */
1312    3               CALL MOVX$DPTR$A;
1313    3               CALL ACALL$ADDR11;
1314    3               CALL MOVX$IND$A;
1315    3               CALL MOVX$IND$A;
1316    3               CALL CPL$A;
1317    3               CALL MOV$DIR$A;
1318    3               CALL MOV$IND$A;
1319    3               CALL MOV$IND$A;
1320    3               CALL MOV$REG$A;
1321    3               CALL MOV$REG$A;
1322    3               CALL MOV$REG$A;
1323    3               CALL MOV$REG$A;
1324    3               CALL MOV$REG$A;
1325    3               CALL MOV$REG$A;
1326    3               CALL MOV$REG$A;
1327    3               CALL MOV$REG$A;

1328    3               END;

1329    2           PSW=(PSW AND 11111110B) + PARITY$STATE(ACC);
1330    2           MACH$CYC=MACH$CYC + EXECUTION$TIME(OPCODE);
1331    2           RETURN PC;
1332    2       END STEP;


1333    1       END SIM51;
```

Appendix B
An Introduction to the Intel® MCS-51® Single-Chip Microcomputer Family

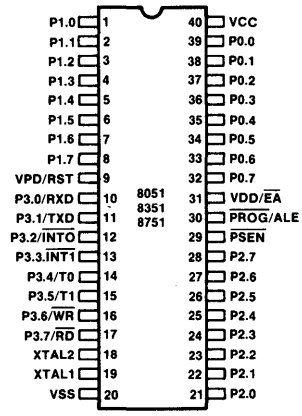


Figure 1a. 8051 Microcomputer Pinout Diagram

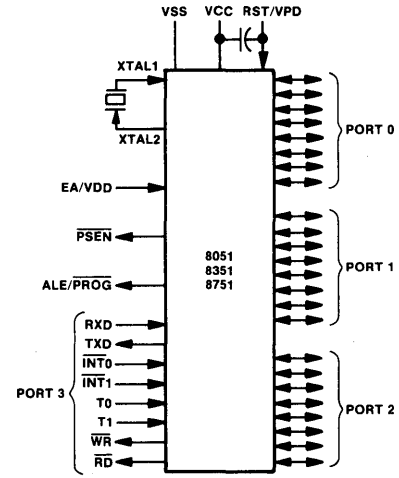


Figure 1b. 8051 Microcomputer Logic Symbol

1. INTRODUCTION

In 1976 Intel introduced the MCS-48™ family, consisting of the 8048, 8748, and 8035 microcomputers. These parts marked the first time a complete microcomputer system, including an eight-bit CPU, 1024 8-bit words of ROM or EPROM program memory, 64 words of data memory, I/O ports and an eight-bit timer/counter could be integrated onto a single silicon chip. Depending only on the program memory contents, one chip could control a limitless variety of products, ranging from appliances or automobile engines to text or data processing equipment. Follow-on products stretched the MCS-48™ architecture in several directions: the 8049 and 8039 doubled the amount of on-chip memory and ran 83% faster; the 8021 reduced costs by executing a subset of the 8048 instructions with a somewhat slower clock; and the 8022 put a unique two-channel 8-bit analog-to-digital converter on the same NMOS chip as the computer, letting the chip interface directly with analog transducers.

Now three new high-performance single-chip microcomputers—the Intel® 8051, 8751, and 8031—extend the advantages of Integrated Electronics to whole new product areas. Thanks to Intel's new HMOS technology, the MCS-51™ family provides four times the program memory and twice the data memory as the 8048 on a single chip. New I/O and peripheral capabilities both increase the range of applicability and reduce total system cost. Depending on the use, processing throughput increases by two and one-half to ten times.

This Application Note is intended to introduce the reader to the MCS-51™ architecture and features. While it does not assume intimacy with the MCS-48™ product line on the part of the reader, he/she should be familiar with some microprocessor (preferably Intel's, of course) or have a background in computer programming and digital logic.

Family Overview

Pinout diagrams for the 8051, 8751, and 8031 are shown in Figure 1. The devices include the following features:

- Single-supply 5 volt operation using HMOS technology.
- 4096 bytes program memory on-chip (not on 8031).
- 128 bytes data memory on-chip.
- Four register banks.
- 128 User-defined software flags.
- 64 Kilobytes each program and external RAM addressability.
- One microsecond instruction cycle with 12 MHz crystal.
- 32 bidirectional I/O lines organized as four 8-bit ports (16 lines on 8031).
- Multiple mode, high-speed programmable Serial Port.
- Two multiple mode, 16-bit Timer/Counters.
- Two-level prioritized interrupt structure.
- Full depth stack for subroutine return linkage and data storage.
- Augmented MCS-48™ instruction set.
- Direct Byte and Bit addressability.
- Binary or Decimal arithmetic.
- Signed-overflow detection and parity computation.
- Hardware Multiple and Divide in 4 μsec.
- Integrated Boolean Processor for control applications.
- Upwardly compatible with existing 8048 software.

AFN-01502A

All three devices come in a standard 40-pin Dual In-Line Package, with the same pin-out, the same timing, and the same electrical characteristics. The primary difference between the three is the on-chip program memory—different types are offered to satisfy differing user requirements.

The 8751 provides 4K bytes of ultraviolet-Erasable, Programmable Read Only Memory (EPROM) for program development, prototyping, and limited production runs. (By convention, 1K means $2^{10} = 1024$. 1k—with a lower case "k"—equals $10^3 = 1000$.) This part may be individually programmed for a specific application using Intel's Universal PROM Programmer (UPP). If software bugs are detected or design specifications change the same part may be "erased" in a matter of minutes by exposure to ultraviolet light and reprogrammed with the modified code. This cycle may be repeated indefinitely during the design and development phase.

The final version of the software must be programmed into a large number of production parts. The 8051 has 4K bytes of ROM which are mask-programmed with the customer's order when the chip is built. This part is considerably less expensive, but cannot be erased or altered after fabrication.

The 8031 does not have any program memory on-chip, but may be used with up to 64K bytes of external standard or multiplexed ROMs, PROMs, or EPROMs. The 8031 fits well in applications requiring significantly larger or smaller amounts of memory than the 4K bytes provided by its two siblings.

(The 8051 and 8751 automatically access external program memory for all addresses greater than the 4096 bytes on-chip. The External Access input is an override for all internal program memory—the 8051 and 8751 will each emulate an 8031 when pin 31 is low.)

Throughout this Note, "8051" is used as a generic term. Unless specifically stated otherwise, the point applies equally to all three components. Table 1 summarizes the quantitative differences between the members of the MCS-48™ and MCS-51™ families.

The remainder of this Note discusses the various MCS-51™ features and how they can be used. Software and/or hardware application examples illustrate many of the concepts. Several isolated tasks (rather than one complete system design example) are presented in the hope that some of them will apply to the reader's experiences or needs.

A document this short cannot detail all of a computer system's capabilities. By no means will all the 8051 instructions be demonstrated; the intent is to stress new or unique MCS-51™ operations and instructions generally used in conjunction with each other. For additional hardware information refer to the Intel **MCS-51™ Family User's Manual,** publication number 121517. The assembly language and use of ASM51, the MCS-51™ assembler, are further described in the **MCS-51™ Macro Assembler User's Guide,** publication number 9800937.

The next section reviews some of the basic concepts of microcomputer design and use. Readers familiar with the 8048 may wish to skim through this section or skip directly to the next, "ARCHITECTURE AND ORGANIZATION."

Microcomputer Background Concepts

Most digital computers use the binary (base 2) number system internally. All variables, constants, alphanumeric characters, program statements, etc., are represented by groups of binary digits ("bits"), each of which has the value 0 or 1. Computers are classified by how many bits they can move or process at a time.

The MCS-51™ microcomputers contain an eight-bit central processing unit (CPU). Most operations process variables eight bits wide. All internal RAM and ROM, and virtually all other registers are also eight bits wide. An eight-bit ("byte") variable (shown in Figure 2) may assume one of $2^8 = 256$ distinct values, which usually represent integers between 0 and 255. Other types of numbers, instructions, and so forth are represented by one or more bytes using certain conventions.

For example, to represent positive and negative values, the most significant bit (D7) indicates the sign of the other seven bits—0 if positive, 1 if negative—allowing integer variables between -128 and +127. For integers with extremely large magnitudes, several bytes are manipulated together as "multiple precision" signed or unsigned integers—16, 24, or more bits wide.

Table 1. Features of Intel's Single-Chip Microcomputers

EPROM Program Memory	ROM Program Memory	External Program Memory	Program Memory (Int/Max)	Data Memory (Bytes)	Instr. Cycle Time	Input/ Output Pins	Interrupt Sources	Reg. Banks
—	8021	—	1K/1K	64	8.4 μSec	21	0	1
—	8022	—	2K/2K	64	8.4 μSec	28	2	1
8748	8048	8035	1K/4K	64	2.5 μSec	27	2	2
—	8049	8039	2K/4K	128	1.36 μSec	27	2	2
8751	8051	8031	4K/64K	128	1.0 μSec	32	5	4

AFN-01502A

The letters "MCS" have traditionally indicated a system or family of compatible Intel® microcomputer components, including CPUs, memories, clock generators, I/O expanders, and so forth. The numerical suffix indicates the microprocessor or microcomputer which serves as the cornerstone of the family. Microcomputers in the MCS-48™ family currently include the 8048-series (8035, 8048, & 8748), the 8049-series (8039 & 8049), and the 8021 and 8022; the family also includes the 8243, an I/O expander compatible with each of the microcomputers. Each computer's CPU is derived from the 8048, with essentially the same architecture, addressing modes, and instruction set, and a single assembler (ASM48) serves each.

The first members of the MCS-51™ family are the 8051, 8751, and 8031. The architecture of the 8051-series, while derived from the 8048, is not strictly compatible; there are more addressing modes, more instructions, larger address spaces, and a few other hardware differences. In this Application Note the letters "MCS-51" are used when referring to *architectural* features of the 8051-series—features which would be included on possible future microcomputers based on the 8051 CPU. Such products could have different amounts of memory (as in the 8048/8049) or different peripheral functions (as in the 8021 and 8022) while leaving the CPU and instruction set intact. ASM51 is the assembler used by all microcomputers in the 8051 family.

Two digit decimal numbers may be "packed" in an eight-bit value, using four bits for the binary code of each digit. This is called Binary-Coded Decimal (BCD) representation, and is often used internally in programs which interact heavily with human beings.

Alphanumeric characters (letters, numbers, punctuation marks, etc.) are often represented using the American Standard Code for Information Interchange (ASCII) convention. Each character is associated with a unique seven-bit binary number. Thus one byte may represent a single character, and a word or sequence of letters may be represented by a series (or "string") of bytes. Since the ASCII code only uses 128 characters, the most significant bit of the byte is not needed to distinguish between characters. Often D7 is set to 0 for all characters. In some coding schemes, D7 is used to indicate the "parity" of the other seven bits—set or cleared as necessary to ensure that the total number of "1" bits in the eight-bit code is even ("even parity") or odd ("odd parity"). The 8051 includes hardware to compute parity when it is needed.

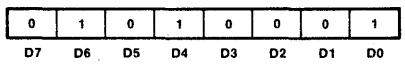


Figure 2. Representation of Bits Within an Eight-Bit "Byte" (Value shown = 01010001 Binary = 81 decimal).

A computer program consists of an ordered sequence of specific, simple steps to be executed by the CPU one-at-a-time. The method or sequence of steps used collectively to solve the user's application is called an "algorithm."

The program is stored inside the computer as a sequence of binary numbers, where each number corresponds to one of the basic operations ("opcodes") which the CPU is capable of executing. In the 8051, each program memory location is one byte. A complete instruction consists of a sequence of one or more bytes, where the first defines the operation to be executed and additional bytes (if needed) hold additional information, such as data values or variable addresses. No instruction is longer than three bytes.

The way in which binary opcodes and modifier bytes are assigned to the CPU's operations is called the computer's "machine language." Writing a program directly in machine language is time-consuming and tedious. Human beings think in words and concepts rather than encoded numbers, so each CPU operation and resource is given a name and standard abbreviation ("mnemonic"). Programs are more easily discussed using these standard mnemonics, or "assembly language," and may be typed into an Intel® Intellec® 800 or Series II® microcomputer development system in this form. The development system can mechanically translate the program from assembly language "source" form to machine language "object" code using a program called an "assembler." The MCS-51™ assembler is called ASM51.

There are several important differences between a computer's machine language and the assembly language used as a tool to represent it. The machine language or instruction set is the set of operations which the CPU can perform while a program is executing ("at run-time"), and is strictly determined by the microcomputer hardware design.

The assembly language is a standard (though more-or-less arbitrary) set of symbols including the instruction set mnemonics, but with additional features which further simplify the program design process. For example, ASM51 has controls for creating and formatting a program listing, and a number of directives for allocating variable storage and inserting arbitrary bytes of data into the object code for creating tables of constants.

AFN-01502A

In addition, ASM51 can perform sophisticated mathematical operations, computing addresses or evaluating arithmetic expressions to relieve the programmer from this drudgery. However, these calculations can only use information known at "assembly time."

For example, the 8051 performs arithmetic calculations at run-time, eight bits at a time. ASM51 can do similar operations 16 bits at a time. The 8051 can only do one simple step per instruction, while ASM51 can perform complex calculations in each line of source code. However, the operations performed by the assembler may only use parameter values fixed at assembly-time, not variables whose values are unknown until program execution begins.

For example, when the assembly language source line,

```
ADD     A,#(LOOP_COUNT + 1) * 3
```

is *assembled*, ASM51 will find the value of the previously-defined constant "LOOP_COUNT" in an internal symbol table, increment the value, multiply the sum by three, and (assuming it is between -256 and 255 inclusive) truncate the product to eight bits. When this instruction is *executed*, the 8051 ALU will just add that resulting constant to the accumulator.

Some similar differences exist to distinguish number system ("radix") specifications. The 8051 does all computations in binary (though there are provisions for then converting the result to decimal form). In the course of writing a program, though, it may be more convenient to specify constants using some other radix, such as base 10. On other occasions, it is desirable to specify the ASCII code for some character or string of characters without refering to tables. ASM51 allows several representations for constants, which are converted to binary as each instruction is assembled.

For example, binary numbers are represented in the assembly language by a series of ones and zeros (naturally), followed by the letter "B" (for Binary); octal numbers as a series of octal digits (0-7) followed by the letter "O" (for Octal) or "Q" (which doesn't stand for anything, but *looks* sort of like an "O" and is less likely to be confused with a zero).

Hexadecimal numbers are represented by a series of hexadecimal digits (0-9,A-F), followed by (you guessed it) the letter "H." A "hex" number must begin with a decimal digit; otherwise it would look like a user-defined symbol (to be discussed later). A "dummy" leading zero may be inserted before the first digit to meet this constraint. The character string "BACH" could be a legal label for a Baroque music synthesis routine; the string "0BACH" is the hexadecimal constant BAC_{16}. This is a case where adding 0 makes a big difference.

Decimal numbers are represented by a sequence of decimal digits, optionally followed by a "D." If a number has no suffix, it is assumed to be decimal—so it had better not contain any non-decimal digits. "0BAC" is not a legal representation for anything.

When an ASCII code is needed in a program, enclose the desired character between two apostrophes (as in '#') and the assembler will convert it to the appropriate code (in this case 23H). A string of characters between apostrophes is translated into a series of constants; 'BACH' becomes 42H, 41H, 43H, 48H.

These same conventions are used throughout the associated Intel documentation. Table 2 illustrates some of the different number formats.

2. ARCHITECTURE AND ORGANIZATION

Figure 3 blocks out the MCS-51™ internal organization. Each microcomputer combines a Central Processing Unit, two kinds of memory (data RAM plus program ROM or EPROM), Input/Output ports, and the mode,

Table 2. Notations Used to Represent Numbers

Bit Pattern	Binary	Octal	Hexa-Decimal	Decimal	Signed Decimal
0 0 0 0 0 0 0 0	0B	0Q	00H	0	0
0 0 0 0 0 0 0 1	1B	1Q	01H	1	+1
. . . . .	..	...	...	..	....
0 0 0 0 0 1 1 1	111B	7Q	07H	7	+7
0 0 0 0 1 0 0 0	1000B	10Q	08H	8	+8
0 0 0 0 1 0 0 1	1001B	11Q	09H	9	+9
0 0 0 0 1 0 1 0	1010B	12Q	0AH	10	+10
. . . . .	..	...	...	..	....
0 0 0 0 1 1 1 1	1111B	17Q	0FH	15	+15
0 0 0 1 0 0 0 0	10000B	20Q	10H	16	+16
. . . . .	..	...	...	..	....
0 1 1 1 1 1 1 1	1111111B	177Q	7FH	127	+127
1 0 0 0 0 0 0 0	10000000B	200Q	80H	128	-128
1 0 0 0 0 0 0 1	10000001B	201Q	81H	129	-127
. . . . .	.....	...	...	...	....
1 1 1 1 1 1 1 0	11111110B	376Q	0FEH	254	-2
1 1 1 1 1 1 1 1	11111111B	377Q	0FFH	255	-1

AFN-01502A

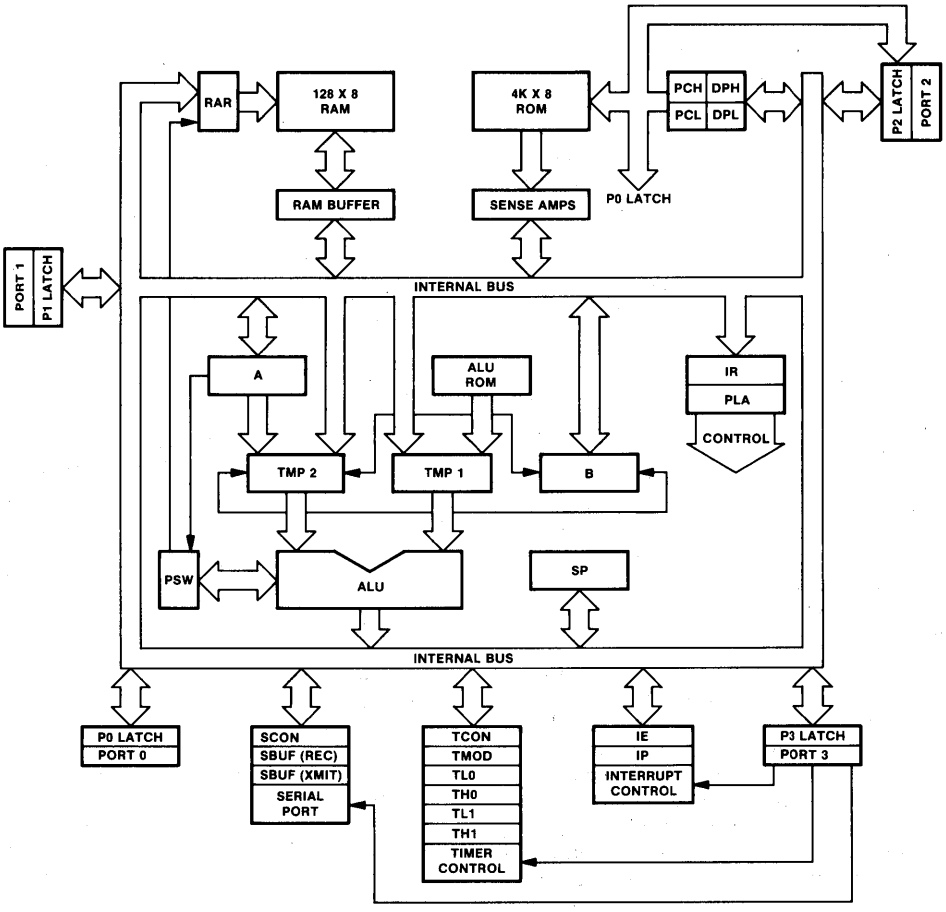


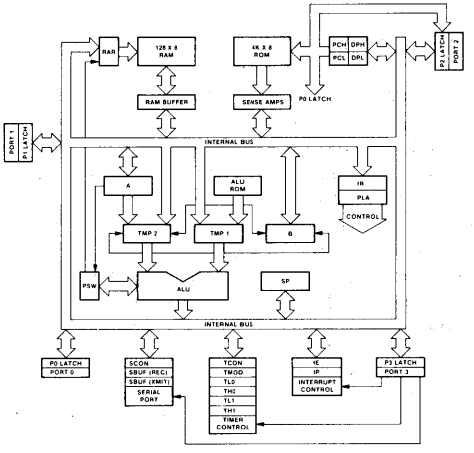
Figure 3. Block Diagram of 8051 Internal Structure

status, and data registers and random logic needed for a variety of peripheral functions. These elements communicate through an eight-bit data bus which runs throughout the chip, somewhat akin to indoor plumbing. This bus is buffered to the outside world through an I/O port when memory or I/O expansion is desired.

Let's summarize what each block does; later chapters dig into the CPU's instruction set and the peripheral registers in much greater detail.

Central Processing Unit

The CPU is the "brains" of the microcomputer, reading the user's program and executing the instructions stored therein. Its primary elements are an eight-bit Arithmetic/Logic Unit with associated registers A, B, PSW, and SP, and the sixteen-bit Program Counter and "Data Pointer" registers.



AFN-01502A

Arithmetic Logic Unit

The ALU can perform (as the name implies) arithmetic and logic functions on eight-bit variables. The former include basic addition, subtraction, multiplication, and division; the latter include the logical operations AND, OR, and Exclusive-OR, as well as rotate, clear, complement, and so forth. The ALU also makes conditional branching decisions, and provides data paths and temporary registers used for data transfers within the system. Other instructions are built up from these primitive functions: the addition capability can increment registers or automatically compute program destination addresses; subtraction is also used in decrementing or comparing the magnitude of two variables.

These primitive operations are automatically cascaded and combined with dedicated logic to build complex instructions such as incrementing a sixteen-bit register pair. To execute one form of the compare instruction, for example, the 8051 increments the program counter three times, reads three bytes of program memory, computes a register address with logical operations, reads internal data memory twice, makes an arithmetic comparison of two variables, computes a sixteen-bit destination address, and decides whether or not to make a branch—all in two microseconds!

An important and unique feature of the MCS-51 architecture is that the ALU can also manipulate one-bit as well as eight-bit data types. Individual bits may be set, cleared, or complemented, moved, tested, and used in logic computations. While support for a more primitive data type may initially seem a step backwards in an era of increasing word length, it makes the 8051 especially well suited for controller-type applications. Such algorithms *inherently* involve Boolean (true/false) input and output variables, which were heretofore difficult to implement with standard microprocessors. These features are collectively referred to as the MCS-51™ "Boolean Processor," and are described in the so-named chapter to come.

Thanks to this powerful ALU, the 8051 instruction set fares well at both real-time control and data intensive algorithms. A total of 51 separate operations move and manipulate three data types: Boolean (1-bit), byte (8-bit), and address (16-bit). All told, there are eleven addressing modes—seven for data, four for program sequence control (though only eight are used by more than just a few specialized instructions). Most operations allow several addressing modes, bringing the total number of instructions (operation/addressing mode combinations) to 111, encompassing 255 of the 256 possible eight-bit instruction opcodes.

Instruction Set Overview

Table 4 lists these 111 instructions classified into five groups:

- Arithmetic Operations
- Logical Operations for Byte Variables
- Data Transfer Instructions
- Boolean Variable Manipulation
- Program Branching and Machine Control

MCS-48™ programmers perusing Table 4 will notice the absence of special categories for Input/Output, Timer/Counter, or Control instructions. These functions are all still provided (and indeed many new functions are added), but as special cases of more generalized operations in other categories. To explicitly list all the useful instructions involving I/O and peripheral registers would require a table approximately four times as long.

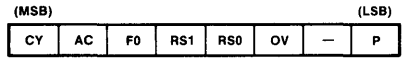
Observant readers will also notice that all of the 8048's page-oriented instructions (conditional jumps, JMPP, MOVP, MOVP3) have been replaced with corresponding but non-paged instructions. The 8051 instruction set is entirely *non*-page-oriented. The MCS-48™ "MOVP" instruction replacement and all conditional jump instructions operate relative to the program counter, with the actual jump address computed by the CPU during instruction execution. The "MOVP3" and "JMPP" replacements are now made relative to another sixteen-bit register, which allows the effective destination to be anywhere in the program memory space, regardless of where the instruction itself is located. There are even three-byte jump and call instructions allowing the destination to be *anywhere* in the 64K program address space.

The instruction set is designed to make programs efficient both in terms of code size and execution speed. No instruction requires more than three bytes of program memory, with the majority requiring only one or two bytes. Virtually all instructions execute in either one or two instruction cycles—one or two microseconds with a 12-MHz crystal—with the sole exceptions (multiply and divide) completing in four cycles.

Many instructions such as arithmetic and logical functions or program control, provide both a short and a long form for the same operation, allowing the programmer to optimize the code produced for a specific application. The 8051 usually fetches two instruction bytes per instruction cycle, so using a shorter form can lead to faster execution as well.

For example, any byte of RAM may be loaded with a constant with a three-byte, two-cycle instruction, but the commonly used "working registers" in RAM may be initialized in one cycle with a two-byte form. Any bit anywhere on the chip may be set, cleared, or complemented by a single three-byte logical instruction using two cycles. But critical control bits, I/O pins, and software flags may be controlled by two-byte, single cycle instructions. While three-byte jumps and calls can "go anywhere" in program memory, nearby sections of code may be reached by shorter relative or absolute versions.

AFN-01502A



Symbol	Position	Name and Significance
CY	PSW.7	Carry flag. Set/cleared by hardware or software during certain arithmetic and logical instructions.
AC	PSW.6	Auxiliary Carry flag. Set/cleared by hardware during addition or subtraction instructions to indicate carry or borrow out of bit 3.
F0	PSW.5	Flag 0 Set/cleared/tested by software as a user-defined status flag.
RS1	PSW.4	Register bank Select control bits 1 & 0. Set/cleared by software to determine
RS	PSW.3	working register bank (see Note).
OV	PSW.2	Overflow flag. Set/cleared by hardware during arithmetic instructions to indicate overflow conditions.
—	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the accumulator, i.e., even parity.

Note— the contents of (RS1, RS0) enable the working register banks as follows:

(0,0)—Bank 0 (00H-07H)
(0,1)—Bank 1 (08H-0FH)
(1,0)—Bank 2 (10H-17H)
(1,1)—Bank 3 (18H-1FH)

Figure 4. PSW—Program Status Word Organization

A significant side benefit of an instruction set more powerful than those of previous single-chip microcomputers is that it is easier to generate applications-oriented software. Generalized addressing modes for byte and bit instructions reduce the number of source code lines written and debugged for a given application. This leads in turn to proportionately lower software costs, greater reliability, and faster design cycles.

Accumulator and PSW

The 8051, like its 8048 predecessor, is primarily an accumulator-based architecture: an eight-bit register called the accumulator ("A") holds a source operand and receives the result of the arithmetic instructions (addition, subtraction, multiplication, and division). The accumulator can be the source or destination for logical operations and a number of special data movement instructions, including table look-ups and external RAM expansion. Several functions apply exclusively to the accumulator: rotates, parity computation, testing for zero, and so on.

Many instructions implicitly or explicitly affect (or are affected by) several status flags, which are grouped together to form the Program Status Word shown in Figure 4.

(The period within entries under the Position column is called the "dot operator," and indicates a particular bit position within an eight-bit byte. "PSW.5" specifies bit 5 of the PSW. Both the documentation and ASM51 use this notation.)

The most "active" status bit is called the carry flag (abbreviated "C"). This bit makes possible multiple precision arithmetic operations including addition, subtraction, and rotates. The carry also serves as a "Boolean accumulator" for one-bit logical operations and bit manipulation instructions. The overflow flag (OV) detects when arithmetic overflow occurs on signed integer operands, making two's complement arithmetic possible. The parity flag (P) is updated after every instruction cycle with the even-parity of the accumulator contents.

The CPU does not control the two register-bank select bits, RS1 and RS0. Rather, they are manipulated by software to enable one of the four register banks. The usage of the PSW flags is demonstrated in the Instruction Set chapter of this Note.

Even though the architecture is accumulator-based, provisions have been made to bypass the accumulator in common instruction situations. Data may be moved from any location on-chip to any register, address, or indirect address (and vice versa), any register may be loaded with a constant, etc., all without affecting the accumulator. Logical operations may be performed against registers or variables to alter fields of bits—without using or affecting the accumulator. Variables may be incremented, decremented, or tested without using the accumulator. Flags and control bits may be manipulated and tested without affecting anything else.

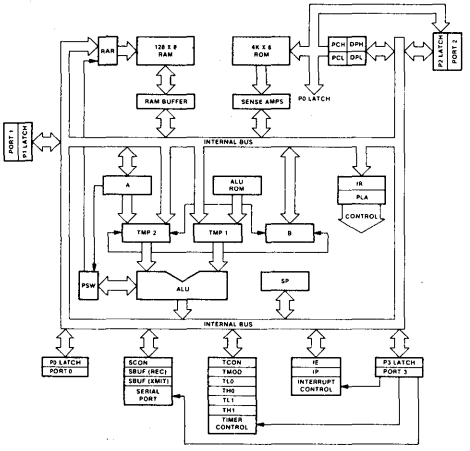
Other CPU Registers

A special eight-bit register ("B") serves in the execution of the multiply and divide instructions. This register is used in conjunction with the accumulator as the second input operand and to return eight-bits of the result.

The MCS-51 family processors include a hardware stack within internal RAM, useful for subroutine linkage,

passing parameters between routines, temporary variable storage, or saving status during interrupt service routines. The Stack Pointer (SP) is an eight-bit pointer register which indicates the address of the last byte pushed onto the stack. The stack pointer is automatically incremented or decremented on all push or pop instructions and all subroutine calls and returns. In theory, the stack in the 8051 may be up to a full 128 bytes deep. (In practice, even simple programs would use a handful of RAM locations for pointers, variables, and so forth—reducing the stack depth by that number.) The stack pointer defaults to 7 on reset, so that the stack will start growing up from location 8, just like in the 8048. By altering the pointer contents the stack may be relocated anywhere within internal RAM.

Finally, a 16-bit register called the data pointer (DPTR) serves as a base register in indirect jumps, table look-up instructions, and external data transfers. The high- and low-order halves of the data pointer may be manipulated as separate registers (DPH and DPL, respectively) or together using special instructions to load or increment all sixteen bits. Unlike the 8048, look-up tables can therefore start anywhere in program memory and be of arbitrary length.



Memory Spaces

Program memory is separate and distinct from data memory. Each memory type has a different addressing mechanism, different control signals, and a different function.

The program memory array (ROM or EPROM), like an elephant, is extremely large and never forgets information, even when power is removed. Program memory is used for information needed each time power is applied: initialization values, calibration constants, keyboard layout tables, etc., as well as the program itself. The program memory has a sixteen-bit address bus; its elements are addressed using the Program Counter or instructions which generate a sixteen-bit address.

To stretch our analogy just a bit, data memory is like a mouse: it is smaller and therefore quicker than program memory, and it goes into a random state when electrical power is applied. On-chip data RAM is used for variables which are determined or may change while the program is running.

A computer spends most of its time manipulating variables, not constants, and a relatively small number of variables at that. Since eight-bits is more than sufficient to uniquely address 128 RAM locations, the on-chip RAM address register is only one byte wide. In contrast to the program memory, data memory accesses need a single eight-bit value—a constant or another variable—to specify a unique location. Since this is the basic width of the ALU and the different memory types, those resources can be used by the addressing mechanisms, contributing greatly to the computer's operating efficiency.

The partitioning of program and data memory is extended to off-chip memory expansion. Each may be added independently, and each uses the same address and data busses, but with different control signals. External program memory is gated onto the external data bus by the $\overline{\text{PSEN}}$ (Program Store Enable) control output, pin 29. External data memory is read onto the bus by the $\overline{\text{RD}}$ output, pin 17, and written with data supplied from the microcomputer by the $\overline{\text{WR}}$ output, pin 16. (There is no control pin to write external program ROM, which is by definition Read Only.) While both types may be expanded to up to 64K bytes, the external data memory may optionally be expanded in 256 byte "pages" to preserve the use of P2 as an I/O port. This is useful with a relatively small expansion RAM (such as the Intel® 8155) or for addressing external peripherals.

Single-chip controller programs are finalized during the project design cycle, and are not modified after production. Intel's single-chip microcomputers are not "von Neumann" architectures common among main-frame and mini-computer systems: the MCS-51™ processor *data* memory—on-chip and external—may *not* be used for program code. Just as there is no write-control signal for program memory, there is no way for the CPU to execute instructions out of RAM. In return, this concession allows an architecture optimized for efficient controller applications: a large, fixed program located in ROM, a hundred or so variables in RAM, and different methods for efficiently addressing each.

(Von Neumann machines are helpful for software development and debug. An 8051 system could be modified to have a single off-chip memory space by gating together the two memory-read controls ($\overline{\text{PSEN}}$ and $\overline{\text{RD}}$) with a two-input AND gate (Figure 5). The CPU could then write data into the common memory array using $\overline{\text{WR}}$ and

AFN-01502A

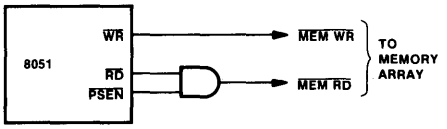


Figure 5. Combining External Program and Data Memory Arrays

external data transfer instructions, and read instructions or data with the AND gate output and data transfer or program memory look-up instructions.)

In addition to the memory arrays, there is (yet) another (albeit sparsely populated) physical address space. Connected to the internal data bus are a score of special-purpose eight-bit registers scattered throughout the chip. Some of these—B, SP, PSW, DPH, and DPL—have been discussed above. Others—I/O ports and peripheral function registers—will be introduced in the following sections. Collectively, these registers are designated as the "special-function register" address space. Even the accumulator is assigned a spot in the special-function register address space for additional flexibility and uniformity.

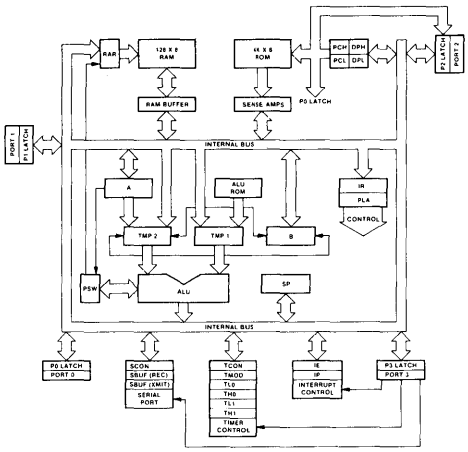
Thus, the MCS-51™ architecture supports several distinct "physical" address spaces, functionally separated at the hardware level by different addressing mechanisms, read and write control signals, or both:

- On-chip program memory;
- On-chip data memory;
- Off-chip program memory;
- Off-chip data memory;
- On-chip special-function registers.

What the *programmer sees,* though, are "logical" address spaces. For example, as far as the programmer is concerned, there is only one type of program memory, 64K bytes in length. The fact that it is formed by combining on- and off-chip arrays (split 4K/60K on the 8051 and 8751) is "invisible" to the programmer; the CPU automatically fetches each byte from the appropriate array, based on its address.

(Presumably, future microcomputers based on the MCS-51™ architecture may have a different physical split, with more or less of the 64K total implemented on-chip. Using the MCS-48™ family as a precedent, the 8048's 4K potential program address space was split 1K/3K between on- and off-chip arrays; the 8049's was split 2K/2K.)

Why go into such tedious details about address spaces? The logical addressing modes are described in the Instruction Set chapter in terms of physical address spaces. Understanding their differences now will pay off in understanding and using the chips later.



Input/Output Ports

The MCS-51™ I/O port structure is extremely versatile. The 8051 and 8751 each have 32 I/O pins configured as four eight-bit parallel ports (P0, P1, P2, and P3). Each pin will input or output data (or both) under software control, and each may be referenced by a wide repertoire of byte and bit operations.

In various operating or expansion modes, some of these I/O pins are also used for special input or output functions. Instructions which access external memory use Port 0 as a multiplexed address/data bus: at the beginning of an external memory cycle eight bits of the address are output on P0; later data is transferred on the same eight pins. External data transfer instructions which supply a sixteen-bit address, and any instruction accessing external program memory, output the high-order eight bits on P2 during the access cycle. (The 8031 *always* uses the pins of P0 and P2 for external addressing, but P1 and P3 are available for standard I/O.)

The eight pins of Port 3 (P3) each have a special function. Two external interrupts, two counter inputs, two serial data lines, and two timing control strobes use pins of P3 as described in Figure 6. Port 3 pins corresponding to functions not used are available for conventional I/O.

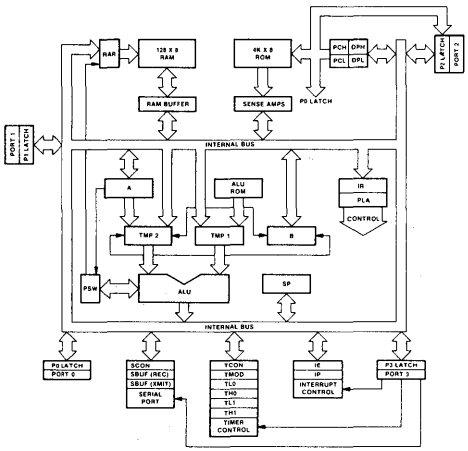
Even within a single port, I/O functions may be combined in many ways: input and output may be performed using different pins at the same time, or the same pins at different times; in parallel in some cases, and in serial in others; as test pins, or (in the case of Port 3) as additional special functions.

AFN-01502A

(MSB)							(LSB)
RD	WR	T1	T0	INT1	INT0	TXD	RXD

Symbol	Position	Name and Significance
RD	P3.7	Read data control output. Active low pulse generated by hardware when external data memory is read.
WR	P3.6	Write data control output. Active low pulse generated by hardware when external data memory is written.
T1	P3.5	Timer/counter 1 external input or test pin.
T0	P3.4	Timer/counter 0 external input or test pin.
INT1	P3.3	Interrupt 1 input pin. Low-level or falling-edge triggered.
INT0	P3.2	Interrupt 0 input pin. Low-level or falling-edge triggered.
TXD	P3.1	Transmit Data pin for serial port in UART mode. Clock output in shift register mode.
RXD	P3.0	Receive Data pin for serial port in UART mode. Data I/O pin in shift register mode.

Figure 6. P3—Alternate Special Functions of Port 3



Special Peripheral Functions

There are a few special needs common among control-oriented computer systems:

- keeping track of elapsed real-time;
- maintaining a count of signal transitions;
- measuring the precise width of input pulses;
- communicating with other systems or people;
- closely monitoring asynchronous external events.

Until now, microprocessor systems needed peripheral chips such as timer/counters, USARTs, or interrupt controllers to meet these needs. The 8051 integrates all of these capabilities on-chip!

Timer/Counters

There are two sixteen-bit multiple-mode Timer/Counters on the 8051, each consisting of a "High" byte (corresponding to the 8048 "T" register) and a low byte (similar to the 8048 prescaler, with the additional flexibility of being software-accessible). These registers are called, naturally enough, TH0, TL0, TH1, and TL1. Each pair may be independently software programmed to any of a dozen modes with a mode register designated TMOD (Figure 7), and controlled with register TCON (Figure 8).

The timer modes can be used to measure time intervals, determine pulse widths, or initiate events, with one-microsecond resolution, up to a maximum interval of 65,536 instruction cycles (over 65 milliseconds). Longer delays may easily be accumulated through software. Configured as a counter, the same hardware will accumulate external events at frequencies from D.C. to 500 KHz, with up to sixteen bits of precision.

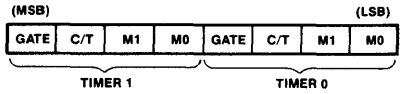
Serial Port Interface

Each microcomputer contains a high-speed, full-duplex, serial port which is software programmable to function in four basic modes: shift-register I/O expander, 8-bit UART, 9-bit UART, or interprocessor communications link. The UART modes will interface with standard I/O devices (e.g. CRTs, teletypewriters, or modems) at data rates from 122 baud to 31 kilobaud. Replacing the standard 12 MHz crystal with a 10.7 MHz crystal allows 110 baud. Even or odd parity (if desired) can be included with simple bit-handling software routines. Inter-processor communications in distributed systems takes place at 187 kilobaud with hardware for automatic address/data message recognition. Simple TTL or CMOS shift registers provide low-cost I/O expansion at a super-fast 1 Megabaud. The serial port operating modes are controlled by the contents of register SCON (Figure 9).

Interrupt Capability and Control

(Interrupt capability is generally considered a CPU function. It is being introduced here since, from an applications point of view, interrupts relate more closely to peripheral and system interfacing.)

AFN-01502A

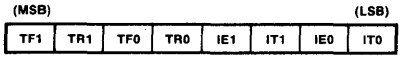


GATE — Gating control. When set, Timer/counter "x" is enabled only while "INTx" pin is high and "TRx" control bit is set. When cleared, timer/counter is enabled whenever "TRx" control bit is set.

C/T — Timer or Counter Selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

M1	M0	Operating Mode
0	0	MCS-48 Timer. "TLx" serves as five-bit prescaler.
0	1	16-bit timer/counter. "THx" and "TLx" are cascaded; there is no prescaler.
1	0	8-bit auto-reload timer/counter. "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an eight-bit timer/counter controlled by the standard Timer 0 control bits. TH0 is an eight-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/counter 1 stopped.

Figure 7. TMOD—Timer/Counter Mode Register



Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

Figure 8. TCON—Timer/Counter Control/Status Register

AFN-01502A

Symbol	Position	Name and Significance
SM0	SCON.7	Serial port Mode control bit 0. Set/cleared by software (see note).
SM1	SCON.6	Serial port Mode control bit 1. Set/cleared by software (see note).
SM2	SCON.5	Serial port Mode control bit 2. Set by software to disable reception of frames for which bit 8 is zero.
REN	SCON.4	Receiver Enable control bit. Set/cleared by software to enable/disable serial data reception.
TB8	SCON.3	Transmit Bit 8. Set/cleared by hardware to determine state of ninth data bit transmitted in 9-bit UART mode.
RB8	SCON.2	Receive Bit 8. Set/cleared by hardware to indicate state of ninth data bit received.
TI	SCON.1	Transmit Interrupt flag. Set by hardware when byte transmitted. Cleared by software after servicing.
RI	SCON.0	Received Interrupt flag. Set by hardware when byte received. Cleared by software after servicing.
	Note—	the state of (SM0,SM1) selects: (0,0)—Shift register I/O expansion. (0,1)—8 bit UART, variable data rate. (1,0)—9 bit UART, fixed data rate. (1,1)—9 bit UART, variable data rate.

Figure 9. SCON—Serial Port Control/Status Register

These peripheral functions allow special hardware to monitor real-time signal interfacing without bothering the CPU. For example, imagine serial data is arriving from one CRT while being transmitted to another, and one timer/counter is tallying high-speed input transitions while the other measures input pulse widths. During all of this the CPU is thinking about something else.

But how does the CPU know when a reception, transmission, count, or pulse is finished? The 8051 programmer can choose from three approaches.

TCON and SCON contain status bits set by the hardware when a timer overflows or a serial port operation is completed. The first technique reads the control register into the accumulator, tests the appropriate bit, and does a conditional branch based on the result. This "polling" scheme (typically a three-instruction sequence though additional instructions to save and restore the accumulator may sometimes be needed) will surely be familiar to programmers used to multi-chip microcomputer systems and peripheral controller chips. This process is rather cumbersome, especially when monitoring multiple peripherals.

As a second approach, the 8051 can perform a conditional branch based on the state of any control or status bit or input pin in a single instruction; a four instruction sequence could poll the four simultaneous happenings mentioned above in just eight microseconds.

Unfortunately, the CPU must still drop what it's doing to test these bits. A manager cannot do his own work well if he is continuously monitoring his subordinates; they should interrupt him (or her) only when they need attention or guidance. So it is with machines: ideally, the CPU would not have to worry about the peripherals until they require servicing. At that time, it would postpone the background task long enough to handle the appropriate device, then return to the point where it left off.

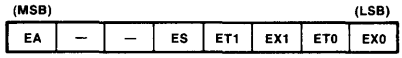
This is the basis of the third and generally optimal solution, hardware interrupts. The 8051 has five interrupt sources: one from the serial port when a transmission or reception is complete, two from the timers when overflows occur, and two from input pins INT0 and INT1. Each source may be independently enabled or disabled to allow polling on some sources or at some times, and each may be classified as high or low priority. A high priority source can interrupt a low priority service routine; the manager's boss can interrupt conferences with subordinates. These options are selected by the interrupt enable and priority control registers, IE and IP (Figures 10 and 11).

Each source has a particular program memory address associated with it (Table 3), starting at 0003H (as in the 8048) and continuing at eight-byte intervals. When an event enabled for interrupts occurs the CPU automatically executes an internal subroutine call to the corresponding address. A user subroutine starting at this location (or jumped to from this location) then performs the instructions to service that particular source. After completing the interrupt service routine, execution returns to the background program.

Table 3. 8051 Interrupt Sources and Service Vectors

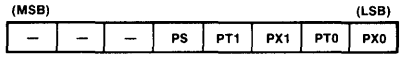
Interrupt Source	Service Routine Starting Address
(Reset)	0000H
External 0	0003H
Timer/Counter 0	000BH
External 1	0013H
Timer/Counter 1	001BH
Serial Port	0023H

AFN-01502A



Symbol	Position	Name and Significance
EA	IE.7	Enable All control bit. Cleared by software to disable all interrupts, independent of the state of IE.4-IE.0.
—	IE.6	(reserved)
—	IE.5	(reserved)
ES	IE.4	Enable Serial port control bit. Set/cleared by software to enable/disable interrupts from TI or RI flags.
ET1	IE.3	Enable Timer 1 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 1.
EX1	IE.2	Enable External interrupt 1 control bit. Set/cleared by software to enable/disable interrupts from INT1.
ET0	IE.1	Enable Timer 0 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 0
EX0	IE.0	Enable External interrupt 0 control bit. Set/cleared by software to enable/disable interrupts from INT0.

Figure 10. IE—Interrupt Enable Register



Symbol	Position	Name and Significance
—	IP.7	(reserved)
—	IP.6	(reserved)
—	IP.5	(reserved)
PS	IP.4	Serial port Priority control bit. Set/cleared by software to specify high/low priority interrupts for Serial port.
PT1	IP.3	Timer 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 1.
PX1	IP.2	External interrupt 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT1.
PT0	IP.1	Timer 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 0.
PX0	IP.0	External interrupt 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT0.

Figure 11. IP—Interrupt Priority Control Register

AFN-01502A

Table 4. MCS-51™ Instruction Set Description

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w/Borrow	1	1
SUBB	A,#data	Subtract immed. data from A w/Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

LOGICAL OPERATIONS

Mnemonic		Destination	Byte	Cyc
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND direct byte to Accumulator	2	1
ANL	A,@Ri	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

DATA TRANSFER

Mnemonic		Description	Byte	Cyc
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	2

DATA TRANSFER (cont.)

Mnemonic		Description	Byte	Cyc
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	1	2
MOVX	A,@Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move External RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to External RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind. RAM w/A	1	1

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Cyc
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry flag	2	2
ANL	C,/bit	AND complement of direct bit to Carry	2	2
ORL	C,bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry	2	2
MOV	C,bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2

PROGRAM AND MACHINE CONTROL

Mnemonic		Description	Byte	Cyc
ACALL	addr11	Absolute Subroutine Call	2	2
LCALL	addr16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute Jump	2	2
LJMP	addr16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg. & Jump if Not Equal	3	2
CJNE	@Ri,#data,rel	Comp. immed. to ind. & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn —Working register R0-R7
direct —128 internal RAM locations, any I/O port, control or status register
@Ri —Indirect internal RAM location addressed by register R0 or R1
#data —8-bit constant included in instruction
#data16 —16-bit constant included as bytes 2 & 3 of instruction
bit —128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16 —Destination address for LCALL & LJMP may be anywhere within the 64-Kilobyte program memory address space.
addr11 —Destination address for ACALL & AJMP will be within the same 2-Kilobyte page of program memory as the first byte of the following instruction.
rel —SJMP and all conditional jumps include an 8-bit offset byte. Range is +127/-128 bytes relative to first byte of the following instruction.

3. INSTRUCTION SET AND ADDRESSING MODES

The 8051 instruction set is extremely regular, in the sense that most instructions can operate with variables from several different physical or logical address spaces. Before getting deeply enmeshed in the instruction set proper, it is important to understand the details of the most common data addressing modes. Whereas Table 4 summarizes the instructions set broken down by functional group, this chapter starts with the addressing mode classes and builds to include the related instructions.

Data Addressing Modes

MCS-51 assembly language instructions consist of an operation mnemonic and zero to three operands separated by commas. In two operand instructions the destination is specified first, then the source. Many byte-wide data

operations (such as ADD or MOV) inherently use the accumulator as a source operand and/or to receive the result. For the sake of clarity the letter "A" is specified in the source or destination field in all such instructions. For example, the instruction,

ADD A,<source>

will add the variable<source>to the accumulator, leaving the sum in the accumulator.

The operand designated "<source>" above may use any of four common logical addressing modes:

- Register—one of the working registers in the currently enabled bank.
- Direct—an internal RAM location, I/O port, or special-function register.
- Register-indirect—an internal RAM location, pointed to by a working register.
- Immediate data—an eight-bit constant incorporated into the instruction.

The first three modes provide access to the internal RAM and Hardware Register address spaces, and may therefore be used as source or destination operands; the last mode accesses program memory and may be a source operand only.

(It is hard to show a "typical application" of any instruction without involving instructions not yet described. The following descriptions use only the self-explanatory ADD and MOV instructions to demonstrate how the four addressing modes are specified and used. Subsequent examples will become increasingly complex.)

Register Addressing

The 8051 programmer has access to eight "working registers," numbered R0-R7. The least-significant three-bits of the instruction opcode indicate one register within this logical address space. Thus, a function code and operand address can be combined to form a short (one byte) instruction (Figure 12.a).

The 8051 assembly language indicates register addressing with the symbol Rn (where n is from 0 to 7) or with a symbolic name previously defined as a register by the EQUate or SET directives. (For more information on assembler directives see the Macro Assembler Reference Manual.)

Example 1—Adding Two Registers Together

```
;REGADR ADD CONTENTS OF REGISTER 1
;       TO CONTENTS OF REGISTER 0
;
REGADR: MOV     A,R0
        ADD     A,R1
        MOV     R0,A
```

There are four such banks of working registers, only one of which is active at a time. Physically, they occupy the first 32 bytes of on-chip data RAM (addresses 0-1FH). PSW bits 4 and 3 determine which bank is active. A hardware reset enables register bank 0; to select a different bank the programmer modifies PSW bits 4 and 3 accordingly.

Example 2—Selecting Alternate Memory Banks

```
MOV     PSW,#00010000B  SELECT BANK 2
```

Register addressing in the 8051 is the same as in the 8048 family, with two enhancements: there are four banks rather than one or two, and 16 instructions (rather than 12) can access them.

Direct Byte Addressing

Direct addressing can access any on-chip variable or hardware register. An additional byte appended to the opcode specifies the location to be used (Figure 12.b).

Depending on the highest order bit of the direct address byte, one of two physical memory spaces is selected. When the direct address is between 0 and 127 (00H-7FH) one of the 128 low-order on-chip RAM locations is used. (Future microcomputers based on the MCS-51™ architecture may incorporate more than 128 bytes of on-chip RAM. Even if this is the case, only the low-order 128 bytes will be directly addressable. The remainder would be accessed indirectly or via the stack pointer.)

Example 3—Adding RAM Location Contents

```
;DIRADR ADD CONTENTS OF RAM LOCATION 41H
;       TO CONTENTS OF RAM LOCATION 40H
;
DIRADR: MOV     A,40H
        ADD     A,41H
        MOV     40H,A
```

All I/O ports and special function, control, or status registers are assigned addresses between 128 and 255 (80H-0FFH). When the direct address byte is between these limits the corresponding hardware register is accessed. For example, Ports 0 and 1 are assigned direct addresses 80H and 90H, respectively. A complete list is presented in Table 5. Don't waste your time trying to memorize the addresses in Table 5. Since programs using absolute addresses for function registers would be difficult to write or understand, ASM51 allows and understands the abbreviations listed instead.

Example 4—Adding Input Port Data to Output Port Data

```
;PRTADR ADD DATA INPUT ON PORT 1
;       TO DATA PREVIOUSLY OUTPUT
;       ON PORT 0.
;
PRTADR: MOV     A,P0
        ADD     A,P1
        MOV     P0,A
```

Direct addressing allows all special-function registers in the 8051 to be read, written, or used as instruction operands. In general, this is the *only* method used for accessing I/O ports and special-function registers. If direct addressing is used with special-function register addresses other than those listed, the result of the instruction is undefined.

AFN-01502A

The 8048 does not have or need any generalized direct addressing mode, since there are only five special registers (BUS, P1, P2, PSW, & T) rather than twenty. Instead, 16 special 8048 opcodes control output bits or read or write each register to the accumulator. These functions are all subsumed by four of the 27 direct addressing instructions of the 8051.

Table 5. 8051 Hardware Register Direct Addresses

Register	Address	Function
P0	80H*	Port 0
SP	81H	Stack Pointer
DPL	82H	Data Pointer (Low)
DPH	83H	Data Pointer (High)
TCON	88H*	Timer register
TMOD	89H	Timer Mode register
TL0	8AH	Timer 0 Low byte
TL1	8BH	Timer 1 Low byte
TH0	8CH	Timer 0 High byte
TH1	8DH	Timer 1 High byte
P1	90H*	Port 1
SCON	98H*	Serial Port Control register
SBUF	99H	Serial Port data Buffer
P2	0A0H*	Port 2
IE	0A8H*	Interrupt Enable register
P3	0B0H*	Port 3
IP	0B8H*	Interrupt Priority register
PSW	0D0H*	Program Status Word
ACC	0E0H*	Accumulator (direct address)
B	0F0H*	B register

* = bit addressable register.

Register-Indirect Addressing

How can you handle variables whose locations in RAM are determined, computed, or modified while the program is running? This situation arises when manipulating sequential memory locations, indexed entries within tables in RAM, and multiple precision or string operations. Register or Direct addressing cannot be used, since their operand addresses are fixed at assembly time.

The 8051 solution is "register-indirect RAM addressing." R0 and R1 of each register bank may operate as index or pointer registers, their contents indicating an address into RAM. The internal RAM location so addressed is the actual operand used. The least significant bit of the instruction opcode determines which register is used as the "pointer" (Figure 12.c).

In the 8051 assembly language, register-indirect addressing is represented by a commercial "at" sign ("@") preceding R0, R1, or a symbol defined by the user to be equal to R0 or R1.

Example 5—Indirect Addressing

```
; INDADR ADD CONTENTS OF MEMORY LOCATION
;       ADDRESSED BY REGISTER 1
;       TO CONTENTS OF RAM LOCATION
;       ADDRESSED BY REGISTER 0
;
INDADR: MOV     A,@R0
        ADD     A,@R1
        MOV     @R0,A
```

Indirect addressing on the 8051 is the same as in the 8048 family, except that all eight bits of the pointer register contents are significant; if the contents point to a non-existent memory location (i.e., an address greater than 7FH on the 8051) the result of the instruction is undefined. (Future microcomputers based on the MCS-51™ architecture could implement additional memory in the on-chip RAM logical address space at locations above 7FH.) The 8051 uses register-indirect addressing for five new instructions plus the 13 on the 8048.

Immediate Addressing

When a source operand is a constant rather than a variable (i.e.—the instruction uses a value known at assembly time), then the constant can be incorporated into the instruction. An additional instruction byte specifies the value used (Figure 12.d).

The value used is fixed at the time of ROM manufacture or EPROM programming and may not be altered during program execution. In the assembly language immediate operands are preceded by a number sign ("#"). The operand may be either a numeric string, a symbolic variable, or an arithmetic expression using constants.

Example 6—Adding Constants Using Immediate Addressing

```
; IMMADR ADD THE CONSTANT 12 (DECIMAL)
;       TO THE CONSTANT 34 (DECIMAL).
;       LEAVE SUM IN ACCUMULATOR.
;
IMMADR: MOV     A,#12
        ADD     A,#34
```

The preceding example was included for consistency; it has little practical value. Instead, ASM51 could compute the sum of two constants at assembly time.

Example 7—Adding Constants Using ASM51 Capabilities

```
; ASMSUM LOAD ACC WITH THE SUM OF
;       THE CONSTANT 12 (DECIMAL) AND
;       THE CONSTANT 34 (DECIMAL).
;
ASMSUM: MOV     A,#(12+34)
```

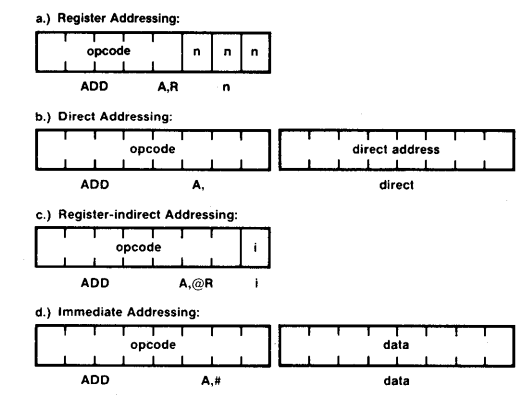


Figure 12. Data Addressing Machine Code Formats

AFN-01502A

Addressing Mode Combinations

The above examples all demonstrated the use of the four data-addressing modes in two-operand instructions (MOV, ADD) which use the accumulator as one operand. The operations ADDC, SUBB, ANL, ORL, and XRL (all to be discussed later) could be substituted for ADD in each example. The first three modes may be also be used for the XCH operation or, in combination with the Immediate Addressing mode (and an additional byte), loaded with a constant. The one-operand instructions INC and DEC, DJNZ, and CJNE may all operate on the accumulator, or may specify the Register, Direct, and Register-indirect addressing modes. Exception: as in the 8048, DJNZ cannot use the accumulator or indirect addressing. (The PUSH and POP operations cannot inherently address the accumulator as a special register either. However, all three can *directly* address the accumulator as one of the twenty special-function registers by putting the symbol "ACC" in the operand field.)

Advantages of Symbolic Addressing

Like most assembly or higher-level programming languages, ASM51 allows instructions or variables to be given appropriate, user-defined symbolic names. This is done for instruction lines by putting a label followed by a colon (":") before the instruction proper, as in the above examples. Such symbols must start with an alphabetic character (remember what distinguished BACH from 0BACH?), and may include any combination of letters, numbers, question marks ("?") and underscores ("_"). For **very** long names only the first 31 characters are relevant.

Assembly language programs may intermix upper- and lower-case letters arbitrarily, but ASM51 converts both to upper-case. For example, ASM51 will internally process an "I" for an "i" and, of course, "A_TOOTH" for "a_tooth."

The underscore character makes symbols easier to read and can eliminate potential ambiguity (as in the label for a subroutine to switch two entires on a stack, "S_EXCHANGE"). The underscore is significant, and would distinguish between otherwise-identical character strings.

ASM51 allows *all* variables (registers, ports, internal or external RAM addresses, constants, etc.) to be assigned labels according to these rules with the EQUate or SET directives.

Example 8—Symbolic Addressing of Variables Defined as RAM Locations

```
VAR_0   SET     20H
VAR_1   SET     21H
;
;SYMB_1: ADD CONTENTS OF VAR_1
;        TO CONTENTS OF VAR_0
;
SYMB_1: MOV     A,VAR_0
        ADD     A,VAR_1
        MOV     VAR_0,A
```

Notice from Table 4 that the MCS-51™ instruction set has relatively few instruction mnemonics (abbreviations) for the programmer to memorize. Different data types or addressing modes are determined by the operands specified, rather than variations on the mnemonic. For example, the mnemonic "MOV" is used by 18 different instructions to operate on three data types (bit, byte, and address). The fifteen versions which move byte variables between the logical address spaces are diagrammed in Figure 13. Each arrow shows the direction of transfer from source to destination.

Notice also that for most instructions allowing register addressing there is a corresponding direct addressing instruction and vice versa. This lets the programmer begin writing 8051 programs as if (s)he has access to 128 different registers. When the program has evolved to the point where the programmer has a fairly accurate idea how often each variable is used, he/she may allocate the working registers in each bank to the most "popular" variables. (The assembly cross-reference option will show exactly how often and where each symbol is referenced.) If symbolic addressing is used in writing the source program only the lines containing the symbol definition will need to be changed; the assembler will produce the appropriate instructions even though the rest of the program is left untouched. Editing only the first two lines of Example 8 will shrink the six-byte code segment produced in half.

How are instruction sets "counted"? There is no standard practice; different people assessing the same CPU using different conventions may arrive at different totals.

Each operation is then broken down according to the different addressing modes (or combinations of addressing modes) it can accommodate. The "CLR" mnemonic is used by two instructions with respect to bit variables ("CLR C" and "CLR bit") and once ("CLR A") with regards to bytes. This expansion yields the 111 separate instructions of Table 4.

The method used for the MCS-51® instruction set first breaks it down into "operations": a basic function applied to a single data type. For example, the four versions of the ADD instruction are grouped to form one operation — addition of eight-bit variables. The six forms of the ANL instruction for *byte* variables make up a different operation; the two forms of ANL which operate on *bits* are considered still another. The MOV mnemonic is used by three different operation classes, depending on whether bit, byte, or 16-bit values are affected. Using this terminology the 8051 can perform 51 different operations.

AFN-01502A

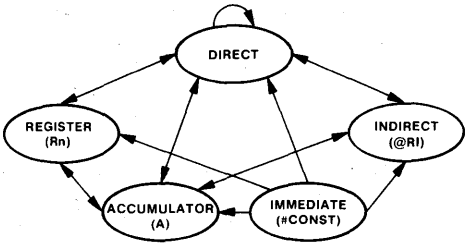


Figure 13. Road map for moving data bytes

Example 9—Redeclaring Example 8 Symbols as Registers

```
VAR_0   SET     R0
VAR_1   SET     R1
;
;SYMB_2 ADD CONTENTS OF VAR_1
;       TO CONTENTS OF VAR_0
;
SYMB_2: MOV     A,VAR_0
        ADD     A,VAR_1
        MOV     VAR_0,A
```

Arithmetic Instruction Usage — ADD, ADDC, SUBB and DA

The ADD instruction adds a byte variable with the accumulator, leaving the result in the accumulator. The carry flag is set if there is an overflow from bit 7 and cleared otherwise. The AC flag is set to the carry-out from bit 3 for use by the DA instruction described later. ADDC adds the previous contents of the carry flag with the two byte variables, but otherwise is the same as ADD.

The SUBB (subtract with borrow) instruction subtracts the byte variable indicated and the contents of the carry flag together from the accumulator, and puts the result back in the accumulator. The carry flag serves as a "Borrow Required" flag during subtraction operations; when a greater value is subtracted from a lesser value (as in subtracting 5 from 1) requiring a borrow into the highest order bit, the carry flag is set; otherwise it is cleared.

When performing signed binary arithmetic, certain combinations of input variables can produce results which seem to violate the Laws of Mathematics. For example, adding 7FH (127) to itself produces a sum of 0FEH, which is the two's complement representation of -2 (refer back to Table 2)! In "normal" arithmetic, two positive values can't have a negative sum. Similarly, it is normally impossible to subtract a positive value from a negative value and leave a positive result — but in two's complement there are instances where this too may happen. Fundamentally, such anomolies occur when the magnitude of the resulting value is too great to "fit" into the seven bits allowed for it; there is no one-byte two's complement representation for 254, the true sum of 127 and 127.

The MCS-51™ processors detect whether these situations occur and indicate such errors with the OV flag. (OV may be tested with the conditional jump instructions JB and JNB, described under the Boolean Processor chapter.)

At a hardware level, OV is set if there is a carry out of bit 6 but not out of bit 7, or a carry out of bit 7 but not out of bit 6. When adding signed integers this indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands; on SUBB this indicates a negative result after subtracting a negative number from a positive number, or a positive result when a positive number is subtracted from a negative number.

The ADDC and SUBB instructions incorporate the previous state of the carry (borrow) flag to allow multiple precision calculations by repeating the operation with successively higher-order operand bytes. In either case, the carry must be cleared before the first iteration.

If the input data for a multiple precision operation is an unsigned string of integers, upon completion the carry flag will be set if an overflow (for ADDC) or underflow (for SUBB) occurs. With two's complement signed data (i.e., if the most significant bit of the original input data indicates the sign of the string), the overflow flag will be set if overflow or underflow occurred.

Example 10—String Subtraction with Signed Overflow Detection

```
;SUBSTR SUBTRACT STRING INDICATED BY R1
;       FROM STRING INDICATED BY R0 TO
;       PRECISION INDICATED BY R2.
;       CHECK FOR SIGNED UNDERFLOW WHEN DONE.
;
SUBSTR: CLR     C           ;BORROW= 0.
SUBS1:  MOV     A,@R0
        SUBB    A,@R1       ;SUBTRACT NEXT PLACE
        MOV     @R0,A
        INC     R0          ;BUMP POINTERS
        INC     R1
        DJNZ    R2,SUBS1            ;LOOP AS NEEDED
;                           WHEN DONE, TEST IF OVERFLOW OCCURED
;                           ON LAST ITERATION OF LOOP.
        JNB     OV,OV_OK
;       ...     .....       (OVERFLOW RECOVERY ROUTINE)
OV_OK:  RET                 ;RETURN
```

Decimal addition is possible by using the DA instruction in conjunction with ADD and/or ADDC. The eight-bit binary value in the accumulator resulting from an earlier addition of two variables (each a packed BCD digit-pair) is adjusted to form two BCD digits of four bits each. If the contents of accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag had been set, six is added to the accumulator producing the proper BCD digit in the low-order nibble. (This addition might itself set — but would not clear — the carry flag.) If the carry flag is set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these bits are incremented by six. The carry flag is left set if originally set or if either addition of six produces a carry out of the highest-order bit, indicating the sum of the original two BCD variables is greater than or equal to decimal 100.

AFN-01502A

Example 11—Two Byte Decimal Add with Registers and Constants

```
;BCDADD ADD THE CONSTANT 1,234 (DECIMAL) TO THE
;       CONTENTS OF REGISTER PAIR <R3><R2>
;       (ALREADY A 4 BCD-DIGIT VARIABLE)
;
BCDADD: MOV     A,R2
        ADD     A,#34H
        DA      A
        MOV     R2,A
        MOV     A,R3
        ADDC    A,#12H
        DA      A
        MOV     R3,A
        RET
```

Multiplication and Division

The instruction "MUL AB" multiplies the unsigned eight-bit integer values held in the accumulator and B-registers. The low-order byte of the sixteen-bit product is left in the accumulator, the higher-order byte in B. If the high-order eight-bits of the product are all zero the overflow flag is cleared; otherwise it is set. The programmer can poll OV to determine when the B register is non-zero and must be processed.

"DIV AB" divides the unsigned eight-bit integer in the accumulator by the unsigned eight-bit integer in the B-register. The integer part of the quotient is returned in the accumulator; the remainder in the B-register. If the B-register originally contained 00H then the overflow flag will be set to indicate a division error, and the values returned will be undefined. Otherwise OV is cleared.

The divide instruction is also useful for purposes such as radix conversion or separating bit fields of the accumulator. A short subroutine can convert an eight-bit unsigned binary integer in the accumulator (between 0 & 255) to a three-digit (two byte) BCD representation. The hundred's digit is returned in one register (HUND) and the ten's and one's digits returned as packed BCD in another (TENONE).

Example 12—Use of DIV Instruction for Radix Conversion

```
;BINBCD CONVERT 8-BIT BINARY VARIABLE IN ACC
;       TO 3-DIGIT PACKED BCD FORMAT
;       HUNDREDS' PLACE LEFT IN VARIABLE 'HUND',
;       TENS' AND ONES' PLACES IN 'TENONE'
;
HUND    EQU     21H
TENONE  EQU     22H
;
BINBCD: MOV     B,#100  ;DIVIDE BY 100 TO
        DIV     AB      ;DETERMINE NUMBER OF HUNDREDS
        MOV     HUND,A
        MOV     A,#10   ;DIVIDE REMAINDER BY 10 TO
        XCH     A,B     ;DETERMINE # OF TENS LEFT
        DIV     AB      ;TENS DIGIT IN ACC, REMAINDER IS ONES
                        ;DIGIT
        SWAP    A
        ADD     A,B     ;PACK BCD DIGITS IN ACC
        MOV     TENONE,A
        RET
```

The divide instruction can also separate eight bits of data in the accumulator into sub-fields. For example, packed BCD data may be separated into two nibbles by dividing the data by 16, leaving the high-nibble in the accumulator and the low-order nibble (remainder) in B. The two digits may then be operated on individually or in conjunction with each other. This example receives two packed BCD digits in the accumulator and returns the product of the two individual digits in packed BCD format in the accumulator.

Example 13—Implementing a BCD Multiply Using MPY and DIV

```
;MULBCD UNPACK TWO BCD DIGITS RECEIVED IN ACC,
;       FIND THEIR PRODUCT, AND RETURN PRODUCT
;       IN PACKED BCD FORMAT IN ACC.
;
MULBCD: MOV     B,#10H  ;DIVIDE INPUT BY 16
        DIV     AB      ;A & B HOLD SEPARATED DIGITS
                        ;(EACH RIGHT JUSTIFIED IN REGISTER).
        MUL     AB      ;A HOLDS PRODUCT IN BINARY FORMAT (0 -
                        ;99(DECIMAL) = 0 - 63H)
        MOV     B,#10   ;DIVIDE PRODUCT BY 10.
        DIV     AB      ;A HOLDS # OF TENS; B HOLDS REMAINDER
        SWAP    A
        ORL     A,B     ;PACK DIGITS
        RET
```

Logical Byte Operations — ANL, ORL, XRL

The instructions ANL, ORL, and XRL perform the logical functions AND, OR, and/or Exclusive-OR on the two byte variables indicated, leaving the results in the first. No flags are affected. (A word to the wise — do not vocalize the first two mnemonics in mixed company.)

These operations may use all the same addressing modes as the arithmetics (ADD, etc.) but unlike the arithmetics, they are not restricted to operating on the accumulator. Directly addressed bytes may be used as the destination with either the accumulator or a constant as the source. These instructions are useful for clearing (ANL), setting (ORL), or complementing (XRL) one or more bits in a RAM, output ports, or control registers. The pattern of bits to be affected is indicated by a suitable mask byte. Use immediate addressing when the pattern to be affected is known at assembly time (Figure 14); use the accumulator versions when the pattern is computed at run-time.

I/O ports are often used for parallel data in formats other than simple eight-bit bytes. For example, the low-order five bits of port 1 may output an alphabetic character code (hopefully) without disturbing bits 7-5. This can be a simple two-step process. First, clear the low-order five pins with an ANL instruction; then set those pins corresponding to ones in the accumulator. (This example assumes the three high-order bits of the accumulator are originally zero.)

Example 14—Reconfiguring Port Size with Logical Byte Instructions

```
OUT_PX: ANL   P1,#11100000B ;CLEAR BITS P1.4 - P1.0
        ORL   P1,A          ;SET P1 PINS CORRESONDING TO SET ACC
                            ; BITS.
        RET
```

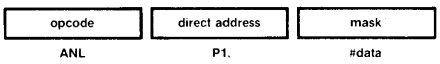


Figure 14. Instruction Pattern for Logical Operation Special Addressing Modes

AFN-01502A

In this example, low-order bits remaining high may "glitch" low for one machine cycle. If this is undesirable, use a slightly different approach. First, set all pins corresponding to accumulator one bits, then clear the pins corresponding to zeroes in low-order accumulator bits. Not all bits will change from original to final state at the same instant, but no bit makes an intermediate transition.

Example 15—Reconfiguring I/O Port Size without Glitching

```
ALT_PX: ORL     P1,A
        ORL     A,#11100000B
        ANL     P1,A
        RET
```

Program Control — Jumps, Calls, Returns

Whereas the 8048 only has a single form of the simple jump instruction, the 8051 has three. Each causes the program to unconditionally jump to some other address. They differ in how the machine code represents the destination address.

LJMP (Long Jump) encodes a sixteen-bit address in the second and third instruction bytes (Figure 15.a); the destination may be anywhere in the 64 Kilobyte program memory address space.

The two-byte AJMP (Absolute Jump) instruction encodes its destination using the same format as the 8048: address bits 10 through 8 form a three bit field in the opcode and address bits 7 through 0 form the second byte (Figure 15.b). Address bits 15-12 are unchanged from the (incremented) contents of the P.C., so AJMP can only be used when the destination is known to be within the same 2K memory block. (Otherwise ASM51 will point out the error.)

A different two-byte jump instruction is legal with any nearby destination, regardless of memory block boundaries or "pages." SJMP (Short Jump) encodes the destination with a program counter-relative address in the second byte (Figure 15.c). The CPU calculates the destination at run-time by adding the signed eight-bit displacement value to the incremented P.C. Negative offset values will cause jumps up to 128 bytes backwards; positive values up to 127 bytes forwards. (SJMP with 00H in the machine code offset byte will proceed with the following instruction).

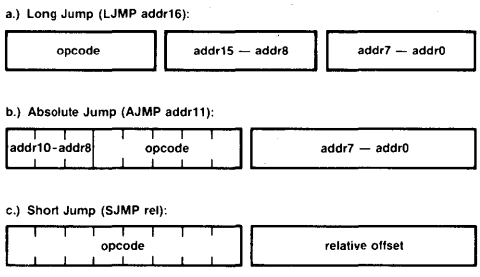


Figure 15. Jump Instruction Machine Code Formats

In keeping with the 8051 assembly language goal of minimizing the number of instruction mnemonics, there is a "generic" form of the three jump instructions. ASM51 recognizes the mnemonic JMP as a "pseudo-instruction," translating it into the machine instructions LJMP, AJMP, or SJMP, depending on the destination address.

Like SJMP, all conditional jump instructions use relative addressing. JZ (Jump if Zero) and JNZ (Jump if Not Zero) monitor the state of the accumulator as implied by their names, while JC (Jump on Carry) and JNC (Jump on No Carry) test whether or not the carry flag is set. All four are two-byte instructions, with the same format as Figure 15.c. JB (Jump on Bit), JNB (Jump on No Bit) and JBC (Jump on Bit then Clear Bit) can test any status bit or input pin with a three byte instruction; the second byte specifies which bit to test and the third gives the relative offset value.

There are two subroutine-call instructions, LCALL (Long Call) and ACALL (Absolute Call). Each increments the P.C. to the first byte of the following instruction, then pushes it onto the stack (low byte first). Saving both bytes increments the stack pointer by two. The subroutine's starting address is encoded in the same ways as LJMP and AJMP. The generic form of the call operation is the mnemonic CALL, which ASM51 will translate into LCALL or ACALL as appropriate.

The return instruction RET pops the high- and low-order bytes of the program counter successively from the stack, decrementing the stack pointer by two. Program execution continues at the address previously pushed: the first byte of the instruction immediately following the call.

When an interrupt request is recognized by the 8051 hardware, two things happen. Program control is automatically "vectored" to one of the interrupt service routine starting addresses by, in effect, forcing the CPU to process an LCALL instead of the next instruction. This automatically stores the return address on the stack. (Unlike the 8048, no status information is automatically saved.)

Secondly, the interrupt logic is disabled from accepting any other interrupts from the same or lower priority. After completing the interrupt service routine, executing an RETI (Return from Interrupt) instruction will return execution to the point where the background program was interrupted — just like RET — while restoring the interrupt logic to its previous state.

AFN-01502A

Operate-and-branch instructions — CJNE, DJNZ

Two groups of instructions combine a byte operation with a conditional jump based on the results.

CJNE (Compare and Jump if Not Equal) compares two byte operands and executes a jump if they disagree. The carry flag is set following the rules for subtraction: if the unsigned integer value of the first operand is less than that of the second it is set; otherwise, it is cleared. However, neither operand is modified.

The CJNE instruction provides, in effect, a one-instruction "case" statement. This instruction may be executed repeatedly, comparing the code variable to a list of "special case" value: the code segment following the instruction (up to the destination label) will be executed only if the operands match. Comparing the accumulator or a register to a series of constants is a convenient way to check for special handling or error conditions; if none of the cases match the program will continue with "normal" processing.

A typical example might be a word processing device which receives ASCII characters through the serial port and drives a thermal hard-copy printer. A standard routine translates "printing" characters to bit patterns, but control characters (<DEL>, <CR>, <LF>, <BEL>, <ESC> or <SP>) must invoke corresponding special routines. Any other character with an ASCII code less than 20H should be translated into the <NUL> value, 00H, and processed with the printing characters.

Example 16—Case Statements Using CJNE

```
CHAR    EQU     R7      ;CHARACTER CODE VARIABLE
;
INTERP: CJNE    CHAR,#7FH,INTP_1
;       ...     .....   (SPECIAL ROUTINE FOR RUBOUT CODE)
        RET
INTP_1: CJNE    CHAR,#07H,INTP_2
;       ...     .....   (SPECIAL ROUTINE FOR BELL CODE)
        RET
INTP_2: CJNE    CHAR,#0AH,INTP_3
;       ...     .....   (SPECIAL ROUTINE FOR LFEED CODE)
        RET
INTP_3: CJNE    CHAR,#0DH,INTP_4
;       ...     .....   (SPECIAL ROUTINE FOR RETURN CODE)
        RET
INTP_4: CJNE    CHAR,#1BH,INTP_5
;       ...     .....   (SPECIAL ROUTINE FOR ESCAPE CODE)
        RET
INTP_5: CJNE    CHAR,#20H,INTP_6
;       ...     .....   (SPECIAL ROUTINE FOR SPACE CODE)
        RET
INTP_6: JC      PRINTC  ;JUMP IF CODE > 20H
        MOV     CHAR,#0 ;REPLACE CONTROL CHARACTERS WITH
                        ; NULL CODE
PRINTC:                 ;PROCESS STANDARD PRINTING
;       ...     ...     ; CHARACTER
        RET
```

DJNZ (Decrement and Jump if Not Zero) decrements the register or direct address indicated and jumps if the result is not zero, without affecting any flags. This provides a simple means for executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. For example, a 99-usec. software delay loop can be added to code forcing an I/O pin low with only two instructions.

Example 17—Inserting a Software Delay with DJNZ

```
        CLR     WR
        MOV     R2,#49
        DJNZ    R2,$
        SETB    WR
```

The dollar sign in this example is a special character meaning "the address of this instruction." It is useful in eliminating instruction labels on the same or adjacent source lines. CJNE and DJNZ (like all conditional jumps) use program-counter relative addressing for the destination address.

Stack Operations — PUSH, POP

The PUSH instruction increments the stack pointer by one, then transfers the contents of the single byte variable indicated (direct addressing only) into the internal RAM location addressed by the stack pointer. Conversely, POP copies the contents of the internal RAM location addressed by the stack pointer to the byte variable indicated, then decrements the stack pointer by one.

(Stack Addressing follows the same rules, and addresses the same locations as Register-indirect. Future microcomputers based on the MCS-51™ CPU could have up to 256 bytes of RAM for the stack.)

Interrupt service routines must not change any variable or hardware registers modified by the main program, or else the program may not resume correctly. (Such a change might look like a spontaneous random error.) Resources used or altered by the service routine (Accumulator, PSW, etc.) must be saved and restored to their previous value before returning from the service routine. PUSH and POP provide an efficient and convenient way to save register states on the stack.

Example 18—Use of the Stack for Status Saving on Interrupts

```
LOC_TMP EQU     $       ;REMEMBER LOCATION COUNTER
;
        ORG     0003H   ;STARTING ADDRESS FOR INTERRUPT ROUTINE
        LJMP    SERVER  ;JUMP TO ACTUAL SERVICE ROUTINE LOCATED
                        ; ELSEWHERE
;       ...     .....
        ORG     LOC_TMP ;RESTORE LOCATION COUNTER
SERVER: PUSH    PSW
        PUSH    ACC     ;SAVE ACCUMULATOR (NOTE DIRECT ADDRESSING
                        ; NOTATION)
        PUSH    B       ;SAVE B REGISTER
        PUSH    DPL     ;SAVE DATA POINTER
        PUSH    DPH     ;
        MOV     PSW,#00001000B  ;SELECT REGISTER BANK 1
;       ...     .....
;       ...     .....
        POP     DPH     ;RESTORE REGISTERS IN REVERSE ORDER
        POP     DPL
        POP     B
        POP     ACC
        POP     PSW     ;RESTORE PSW AND RE-SELECT ORIGINAL
                        ; REGISTER BANK
        RETI            ;RETURN TO MAIN PROGRAM AND RESTORE
                        ; INTERRUPT LOGIC
```

If the SP register held 1FH when the interrupt was detected, then while the service routine was in progress the stack would hold the registers shown in Figure 16; SP would contain 26H.

The example shows the most general situation; if the service routine doesn't alter the B-register and data pointer, for example, the instructions saving and restoring those registers would not be necessary.

The stack may also pass parameters to and from subroutines. The subroutine can indirectly address the parameters derived from the contents of the stack pointer.

AFN-01502A

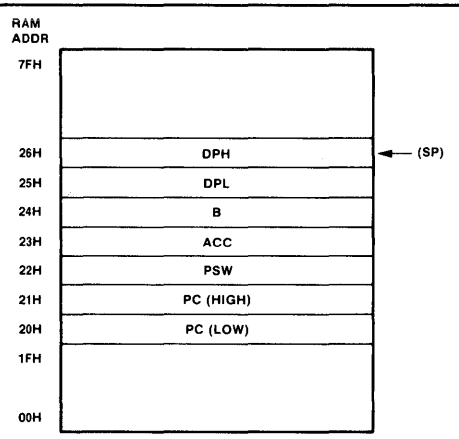


Figure 16. Stack contents during interrupt

One advantage here is simplicity. Variables need not be allocated for specific parameters, a potentially large number of parameters may be passed, and different calling programs may use different techniques for determining or handling the variables.

For example, the following subroutine reads out a parameter stored on the stack by the calling program, uses the low order bits to access a local look-up table holding bit patterns for driving the coils of a four phase stepper motor, and stores the appropriate bit pattern back in the same position on the stack before returning. The accumulator contents are left unchanged.

Example 19—Passing Variable Parameters to Subroutines Using the Stack

```
NXTPOS: MOV     R0, SP
        DEC     R0          ;ACCESS LOCATION PARAMETER PUSHED INTO
        DEC     R0
        XCH     A, @R0      ;READ INPUT PARAMETER AND SAVE
                            ; ACCUMULATOR
        ANL     A, #03H     ;MASK ALL BUT LOW-ORDER TWO BITS
        ADD     A, #2       ;ALLOW FOR OFFSET FROM MOVC TO TABLE
        MOVC    A, @A+PC    ;READ LOOK-UP TABLE ENTRY
        XCH     A, @R0      ;PASS BACK TRANSLATED VALUE AND RESTORE
                            ; ACC
        RET                 ;RETURN TO BACKGROUND PROGRAM
STPTBL: DB      01101111B           ;POSITION 0
        DB      01011111B           ;POSITION 1
        DB      10011111B           ;POSITION 2
        DB      10101111B           ;POSITION 3
```

The background program may reach this subroutine with several different calling sequences, all of which PUSH a value before calling the routine and POP the result after. A motor on Port 1 may be initialized by placing the desired position (zero) on the stack before calling the subroutine and outputing the results directly to a port afterwards.

Example 20—Sending and Receiving Data Parameters Via the Stack

```
        CLR     A
        PUSH    ACC
        CALL    NXTPOS
        POP     P1
```

If the position of the motor is determined by the contents of variable POSM1 (a byte in internal RAM) and the position of a second motor on Port 2 is determined by the data input to the low-order nibble of Port 2, a six-instruction sequence could update them both.

Example 21—Loading and Unloading Stack Direct from I/O Ports

```
POSM1   EQU     51
        .       .
        PUSH    POSM1
        CALL    NXTPOS
        POP     P1
        PUSH    P2
        CALL    NXTPOS
        POP     P2
```

Data Pointer and Table Look-up instructions — MOV, INC, MOVC, JMP

The data pointer can be loaded with a 16-bit value using the instruction MOV DPTR, #data16. The data used is stored in the second and third instruction bytes, high-order byte first. The data pointer is incremented by INC DPTR. A 16-bit increment is performed; an overflow from the low byte will carry into the high-order byte. Neither instruction affects any flags.

The MOVC (Move Constant) instructions (MOVC A,@A+DPTR and MOVC A,@A+PC) read into the accumulator bytes of data from the program memory logical address space. Both use a form of indexed addressing: the former adds the unsigned eight-bit accumulator contents with the sixteen-bit data pointer register, and uses the resulting sum as the address from which the byte is fetched. A sixteen-bit addition is performed; a carry-out from the low-order eight bits may propagate through higher-order bits, but the contents of the DPTR are not altered. The latter form uses the incremented program counter as the "base" value instead of the DPTR (figure 17). Again, neither version affects the flags.

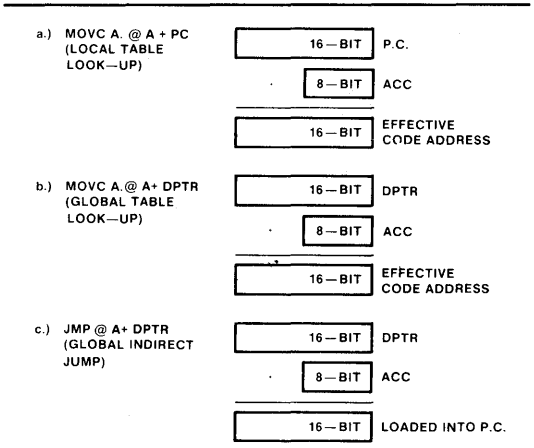


Figure 17. Operation of MOVC instructions

AFN-01502A

Each can be part of a three step sequence to access look-up tables in ROM. To use the DPTR-relative version, load the Data Pointer with the starting address of a look-up table; load the accumulator with (or compute) the index of the entry desired; and execute MOVC A,@A+DPTR. Unlike the similar MOVP3 instructions in the 8048, the table may be located anywhere in program memory. The data pointer may be loaded with a constant for short tables. Or to allow more complicated data structures, or tables with more than 256 entries, the values for DPH and DPL may be computed or modified with the standard arithmetic instruction set.

The PC-relative version has the advantage of not affecting the data pointer. Again, a look-up sequence takes three steps: load the accumulator with the index; compensate for the offset from the look-up instruction to the start of the table by adding the number of bytes separating them to the accumulator; then execute the MOVC A,@A+PC instruction.

Let's look at a non-trivial situation where this instruction would be used. Some applications store large multi-dimensional look-up tables of dot matrix patterns, non-linear calibration parameters, and so on in a linear (one-dimensional) vector in program memory. To retrieve data from the tables, variables representing matrix indices must be converted to the desired entry's memory address. For a matrix of dimensions (MDIMEN x NDIMEN) starting at address BASE and respective indices INDEXI and INDEXJ, the address of element (INDEXI, INDEXJ) is determined by the formula,

Entry Address = BASE + (NDIMEN x INDEXI) + INDEXJ

The code shown below can access any array with less than 255 entries (i.e., an 11x21 array with 231 elements). The table entries are defined using the Data Byte ("DB") directive, and will be contained in the assembly object code as part of the accessing subroutine itself.

Example 22—Use of MPY and Data Pointer Instructions to Access Entries from a Multi-dimensional Look-Up Table in ROM

```
;MATRX1 LOAD CONSTANT READ FROM TWO DIMENSIONAL LOOK-UP
;       TABLE IN PROGRAM MEMORY INTO ACCUMULATOR
;       USING LOCAL TABLE LOOK-UP INSTRUCTION.  'MOVC    A,@A+PC
;       THE TOTAL NUMBER OF TABLE ENTRIES IS ASSUMED TO
;       BE SMALL, I.E. LESS THAN ABOUT 250 ENTRIES.)
;       TABLE USED IN THIS EXAMPLE IS ( 11 X 21 )
;       DESIRED ENTRY ADDRESS IS GIVEN BY THE FORMULA,
;       [ (BASE ADDRESS) + (21 X INDEXI) + (INDEXJ) ]
;
INDEXI  EQU     R6      ;FIRST COORDINATE OF ENTRY (0-10).
INDEXJ  EQU     23H     ;SECOND COORDINATE OF ENTRY (0-20).
;
MATRX1: MOV     A,INDEXI
        MOV     B,#21
        MUL     AB
        ADD     A,INDEXJ
;       ALLOW FOR INSTRUCTION BYTE BETWEEN "MOVC" AND
;       ENTRY (0,0).
        INC     A
        MOVC    A,@A+PC
        RET
BASE1:  DB      1       ;(entry 0,0)
        DB      2       ;(entry 0,1)
;       ..      ....
        DB      21      ;(entry 0,20)
        DB      22      ;(entry 1,0)
;       ..      ....
        DB      42      ;(entry 1,20)
;       ..      ....
;
        DB      231     ;(entry 10,20)
```

There are several different means for branching to sections of code determined or selected at run time. (The single destination addresses incorporated into conditional and unconditional jumps are, of course, determined at assembly time). Each has advantages for different applications.

The most common is an N-way conditional jump based on some variable, with all of the potential destinations known at assembly time. One of a number of small routines is selected according to the value of an index variable determined while the program is running. The most efficient way to solve this problem is with the MOVC and an indirect jump instruction, using a short table of one byte offset values in ROM to indicate the relative starting addresses of the several routines.

JMP @A+DPTR is an instruction which performs an indirect jump to an address determined during program execution. The instruction adds the eight-bit unsigned accumulator contents with the contents of the sixteen-bit data pointer, just like MOVC A,@A+DPTR. The resulting sum is loaded into the program counter and is used as the address for subsequent instruction fetches. Again, a sixteen-bit addition is performed; a carry out from the low-order eight bits may propagate through the higher-order bits. In this case, neither the accumulator contents nor the data pointer is altered.

The example subroutine below reads a byte of RAM into the accumulator from one of four alternate address spaces, as selected by the contents of the variable MEMSEL. The address of the byte to be read is determined by the contents of R0 (and optionally R1). It might find use in a printing terminal application, where four different model printers all use the same ROM code but use different types and sizes of buffer memory for different speeds and options.

Example 23—N-Way Branch and Computed Jump Instructions via JMP @ ADPTR

```
MEMSEL  EQU     R3
;
JUMP_4: MOV     A,MEMSEL
        MOV     DPTR,#JMPTBL
        MOVC    A,@A+DPTR
        JMP     @A+DPTR
JMPTBL: DB      MEMSP0-JMPTBL
        DB      MEMSP1-JMPTBL
        DB      MEMSP2-JMPTBL
        DB      MEMSP3-JMPTBL
MEMSP0: MOV     A,@R0   ;READ FROM INTERNAL RAM
        RET
MEMSP1: MOVX    A,@R0   ;READ FROM 256 BYTES OF EXTERNAL RAM
        RET
MEMSP2: MOV     DPL,R0
        MOV DPH,R1
        MOVX    A,@DPTR ;READ FROM 64K BYTES OF EXTERNAL RAM
        RET
MEMSP3: MOV     A,R1
        ANL     A,#07H
        ANL     P1,#11111000B
        ORL     P1,A
        MOVX    A,@R0   ;READ FROM 4K BYTES OF EXTERNAL RAM
        RET
```

Note that this approach is suitable whenever the size of jump table plus the length of the alternate routines is less than 256 bytes. The jump table and routines may be located anywhere in program memory, independent of 256-byte program memory pages.

AFN-01502A

For applications where up to 128 destinations must be selected, all of which reside in the same 2K page of program memory which may be reached by the two-byte absolute jump instructions, the following technique may be used. In the above mentioned printing terminal example, this sequence could "parse" 128 different codes for ASCII characters arriving via the 8051 serial port.

Example 24—N-Way Branch with 128 Optional Destinations

```
OPTION  EQU     R3
;       ....    ....
;
JMP128: MOV     A,OPTION
        RL      A               ;MULTIPLY BY 2 FOR 2 BYTE JUMP TABLE
        MOV     DPTR,#INSTBL    ;FIRST ENTRY IN JUMP TABLE
        JMP     @A+DPTR         ;JUMP INTO JUMP TABLE
;
INSTBL: AJMP    PROC00  ;128 CONSECUTIVE
        AJMP    PROC01  ;AJMP INSTRUCTIONS
        AJMP    PROC02
;       ....    ....
;
        AJMP    PROC7E
        AJMP    PROC7F
```

The destinations in the jump table (PROC00-PROC7F) are not all necessarily unique routines. A large number of special control codes could each be processed with their own unique routine, with the remaining printing characters all causing a branch to a common routine for entering the character into the output queue.

In those rare situations where even 128 options are insufficient, or where the destination routines may cross a 2K page boundary, the above approach may be modified slightly as shown below.

Example 25—256-Way Branch Using Address Look-Up Tables

```
RTEMP   EQU     R7
;
JMP256: MOV     DPTR,#ADRTBL    ;FIRST ENTRY IN TABLE OF ADDRESSES
        MOV     A,OPTION
        CLR     C
        RLC     A               ;MULTIPLY BY 2 FOR 2 BYTE JUMP TABLE
        JNC     LOW128
        INC     DPH
LOW128: MOV     RTEMP,A         ;SAVE ACC FOR HIGH BYTE READ
        MOVC    A,@A+DPTR       ;READ LOW BYTE FROM JUMP TABLE
        XCH     A,RTEMP
        INC     A
        MOVC    A,@A+DPTR       ;GET LOW-ORDER BYTE FROM TABLE
        PUSH    ACC
        MOV     A,RTEMP
        MOVC    A,@A+DPTR       ;GET HIGH-ORDER BYTE FROM TABLE
        PUSH    ACC
;       THE TWO ACC PUSHES HAVE PRODUCED
;       A "RETURN ADDRESS" ON THE STACK WHICH CORRESPONDS
;       TO THE DESIRED STARTING ADDRESS
;       IT MAY BE REACHED BY POPPING THE STACK
;       INTO THE PC
        RET
;       ....    ....
ADRTBL: DW      PROC00  ;UP TO 256 CONSECUTIVE DATA
        DW      PROC01  ;WORDS INDICATING STARTING ADDRESSES
;       ....    ....
        DW      PROCFF
;
;
;       DUMMY CODE ADDRESS DEFINITIONS NEEDED BY ABOVE
;       TWO EXAMPLES:
;
PROC00: NOP
PROC01: NOP
PROC02: NOP
PROC7E: NOP
PROC7F: NOP
PROCFF: NOP
```

4. BOOLEAN PROCESSING INSTRUCTIONS

The commonly accepted terms for tasks at either end of the computational vs. control application spectrum are, respectively, "number-crunching" and "bit-banging". Prior to the introduction of the MCS-51™ family, nice number-crunchers made bad bit-bangers and vice versa. The 8051 is the industry's first single-chip microcomputer designed to crunch **and** bang. (In some circles, the latter technique is also referred to as "bit-twiddling". Either is correct.)

Direct Bit Addressing

A number of instructions operate on Boolean (one-bit) variables, using a direct bit addressing mode comparable to direct byte addressing. An additional byte appended to the opcode specifies the Boolean variable, I/O pin, or control bit used. The state of any of these bits may be tested for "true" or "false" with the conditional branch instructions JB (Jump on Bit) and JNB (Jump on Not Bit). The JBC (Jump on Bit and Clear) instruction combines a test-for-true with an unconditional clear.

As in direct byte addressing, bit 7 of the address byte switches between two physical address spaces. Values between 0 and 127 (00H-7FH) define bits in internal RAM locations 20H to 2FH (Figure 18a); address bytes between 128 and 255 (80H-0FFH) define bits in the 2 x "special-function" register address space (Figure 18b). If no 2 x "special-function" register corresponds to the direct bit address used the result of the instruction is undefined.

Bits so addressed have many wondrous properties. They may be set, cleared, or complemented with the two byte instructions SETB, CLR, or CPL. Bits may be moved to and from the carry flag with MOV. The logical ANL and ORL functions may be performed between the carry and either the addressed bit or its complement.

Bit Manipulation Instructions — MOV

The "MOV" mnemonic can be used to load an addressable bit into the carry flag ("MOV C, bit") or to copy the state of the carry to such a bit ("MOV bit, C"). These instructions are often used for implementing serial I/O algorithms via software or to adapt the standard I/O port structure.

It is sometimes desirable to "re-arrange" the order of I/O pins because of considerations in laying out printed circuit boards. When interfacing the 8051 to an immediately adjacent device with "weighted" input pins, such as keyboard column decoder, the corresponding pins are likely to be not aligned (Figure 19).

There is a trade-off in "scrambling" the interconnections with either interwoven circuit board traces or through software. This is extremely cumbersome (if not impossible) to do with byte-oriented computer architectures. The 8051's unique set of Boolean instructions makes it simple to move individual bits between arbitrary locations.

AFN-01502A

a.) RAM Bit Addresses.

RAM BYTE	(MSB)							(LSB)
7FH								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH	Bank 3							
18H								
17H	Bank 2							
10H								
0FH	Bank 1							
08H								
07H	Bank 0							
00H								

b.) Hardware Register Bit Addresses.

Direct Byte Address	(MSB)			Bit Addresses				(LSB)	Hardware Register Symbol
0FFH									
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0B8H	—	—	—	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
0A8H	AF	—	—	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Figure 18. Bit Address Maps

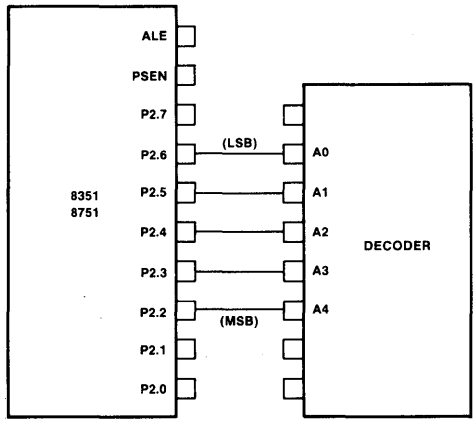


Figure 19. "Mismatch" Between I/O port and Decoder

Example 26—Re-ordering I/O Port Configuration

```
OUT_PZ: RRC     A       ;MOVE ORIGINAL ACC.0 INTO CY
        MOV     P2.6,C  ;STORE CARRY TO PIN P26
        RRC     A       ;MOVE ORIGINAL ACC.1 INTO CY
        MOV     P2.5,C  ;STORE CARRY TO PIN P25
        RRC     A       ;MOVE ORIGINAL ACC.2 INTO CY
        MOV     P2.4,C  ;STORE CARRY TO PIN P24
        RRC     A       ;MOVE ORIGINAL ACC.3 INTO CY
        MOV     P2.3,C  ;STORE CARRY TO PIN P23
        RRC     A       ;MOVE ORIGINAL ACC.4 INTO CY
        MOV     P2.2,C  ;STORE CARRY TO PIN P22
        RET
```

Solving Combinatorial Logic Equations — ANL, ORL

Virtually all hardware designers are familiar with the problem of solving complex functions using combinatorial logic. The technologies involved may vary greatly, from multiple contact relay logic, vacuum tubes, TTL, or CMOS to more esoteric approaches like fluidics, but in each case the goal is the same: a Boolean (true/false) function is computed on a number of Boolean variables.

AFN-01502A

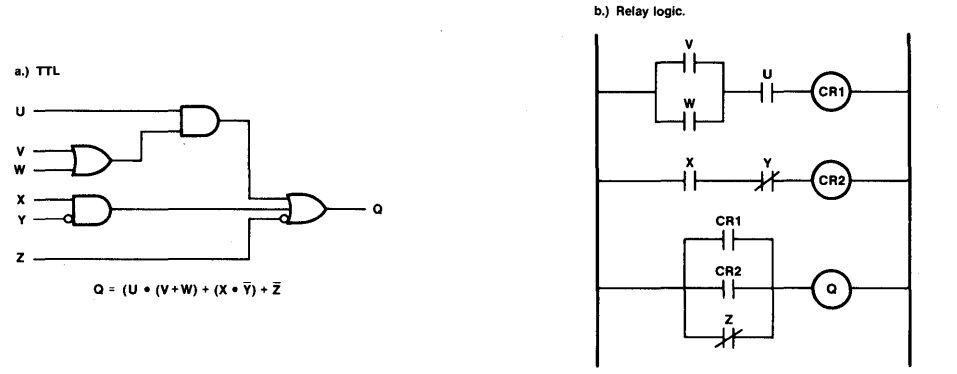


Figure 20. Implementations of Boolean functions

Figure 20 shows the logic diagram for an arbitrary function of six variables named U through Z using standard logic and relay logic symbols. Each is a solution of the equation,

$$Q = (U \cdot (V + W)) + (X \cdot \overline{Y}) + \overline{Z}$$

(While this equation could be reduced using Karnaugh Maps or algebraic techniques, that is not the purpose of this example. Even a minor change to the function equation would require re-reducing from scratch.)

Most digital computers can solve equations of this type with standard word-wide logical instructions and conditional jumps. Still, such software solutions seem somewhat sloppy because of the many paths through the program the computation can take.

Assume U and V are input pins being read by different input ports, W and X are status bits for two peripheral controllers (read as I/O ports), and Y and Z are software flags set or cleared earlier in the program. The end result must be written to an output pin on some third port.

For the sake of comparison we will implement this function with software drawn from three proper subsets of the MCS-51™ instruction set. The first two implementations follow the flow chart shown in Figure 21. Program flow would embark on a route down a test-and-branch tree and leaves either the "True" or "Not True" exit ASAP. These exits then write the output port with the data previously written to the same port with the result bit respectively one or zero.

In the first case, we assume there are no instructions for addressing individual bits other than special flags like the carry. This is typical of many older microprocessors and mainframe computers designed for number-crunching. MCS-51™ mnemonics are used here, though for most other machines the issue would be even further clouded by their use of operation-specific mnemonics like INPUT, OUTPUT, LOAD, STORE, etc., instead of the universal MOV.

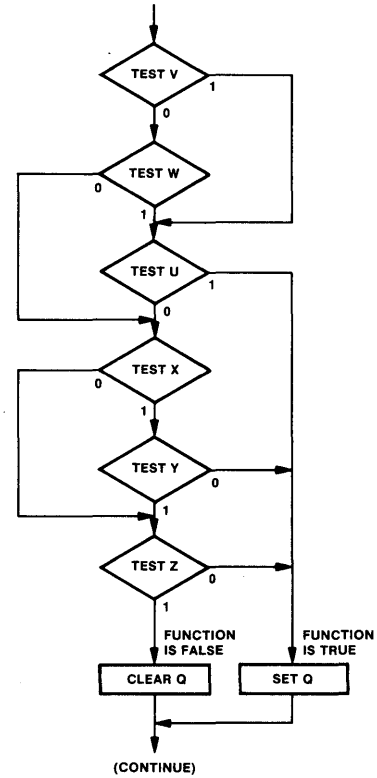


Figure 21. Flow chart for tree-branching logic implementation

AFN-01502A

Example 27—Software Solution to Logic Function of Figure 20, Using only Byte-Wide Logical Instructions

```
; BFUNC1 SOLVE A RANDOM LOGIC FUNCTION OF 6
;       VARIABLES BY LOADING AND MASKING THE APPROPRIATE
;       BITS IN THE ACCUMULATOR, THEN EXECUTING CONDITIONAL
;       JUMPS BASED ON ZERO CONDITION.
;       (APPROACH USED BY BYTE-ORIENTED ARCHITECTURES.)
;       BYTE AND MASK VALUES CORRESPOND TO RESPECTIVE
;       BYTE ADDRESS AND BIT POSITION.
;
OUTBUF  EQU     22H     ;OUTPUT PIN STATE MAP
;
TESTV:  MOV     A,P2
        ANL     A,#00000100B
        JNZ     TESTU
        MOV     A,TCON
        ANL     A,#00100000B
        JZ      TESTX
TESTU:  MOV     A,P1
        ANL     A,#00000010B
        JNZ     SETQ
TESTX:  MOV     A,TCON
        ANL     A,#00001000B
        JZ      TESTZ
        MOV     A,20H
        ANL     A,#00000001B
        JZ      SETQ
TESTZ:  MOV     A,21H
        ANL     A,#00000010B
        JZ      SETQ
CLRQ:   MOV     A,OUTBUF
        ANL     A,#11110111B
        JMP     OUTQ
SETQ:   MOV     A,OUTBUF
        ORL     A,#00001000B
OUTQ:   MOV     OUTBUF,A
        MOV     P3,A
```

Cumbersome, to say the least, and error prone. It would be hard to prove the above example worked in all cases without an exhaustive test.

Each move/mask/conditional jump instruction sequence may be replaced by a single bit-test instruction thanks to direct bit addressing. But the algorithm would be equally convoluted.

Example 28—Software Solution to Logic Function of Figure 20, Using only Bit-Test Instructions

```
; BFUNC2 SOLVE A RANDOM LOGIC FUNCTION OF 6
;       VARIABLES BY DIRECTLY POLLING EACH BIT.
;       (APPROACH USING MCS-51 UNIQUE BIT-TEST
;       INSTRUCTION CAPABILITY.)
;       SYMBOLS USED IN LOGIC DIAGRAM ASSIGNED TO
;       CORRESPONDING 8051 BIT ADDRESSES.
;
U       BIT     P1.1
V       BIT     P2.2
W       BIT     TF0
X       BIT     IE1
Y       BIT     20H.0
Z       BIT     21H.1
Q       BIT     P3.3
;       ...     ...
TEST_V: JB      V,TEST_U
        JNB     W,TEST_X
TEST_U: JB      U,SET_Q
TEST_X: JNB     X,TEST_Z
        JNB     Y,SET_Q
TEST_Z: JNB     Z,SET_Q
CLR_Q:  CLR     Q
        JMP     NXTTST
SET_Q:  SETB    Q
NXTTST:                 ;(CONTINUATION OF PROGRAM)
```

A more elegant and efficient 8051 implementation uses the Boolean ANL and ORL functions to generate the output function using straight-line code. These instructions perform the corresponding logical operations between the carry flag ("Boolean Accumulator") and the addressed bit, leaving the result in the carry. Alternate forms of each instruction (specified in the assembly language by placing a slash before the bit name) use the complement of the bit's state as the input operand.

These instructions may be "strung together" to simulate a multiple input logic gate. When finished, the carry flag contains the result, which may be moved directly to the destination or output pin. No flow chart is needed — it is simple to code directly from the logic diagrams in Figure 20.

Example 29—Software Solution to Logic Function of Figure 20, Using the MCS-51 (TM) Unique Logical Instructions on Boolean Variables

```
; BFUNC3 SOLVE A RANDOM LOGIC FUNCTION OF 6
;       VARIABLES USING STRAIGHT-LINE LOGICAL INSTRUCTIONS
;       ON MCS-51 BOOLEAN VARIABLES.
;
        MOV     C,V
        ORL     C,W     ;OUTPUT OF OR GATE
        ANL     C,U     ;OUTPUT OF TOP AND GATE
        MOV     F0,C    ;SAVE INTERMEDIATE STATE
        MOV     C,X
        ANL     C,/Y    ;OUTPUT OF BOTTOM AND GATE
        ORL     C,F0    ;INCLUDE VALUE SAVED ABOVE
        ORL     C,/Z    ;INCLUDE LAST INPUT VARIABLE
        MOV     Q,C     ;OUTPUT COMPUTED RESULT
        ...     .....
```

Simplicity itself. Fast, flexible, reliable, easy to design, and easy to debug.

The Boolean features are useful and unique enough to warrant a complete Application Note of their own. Additional uses and ideas are presented in Application Note AP-70, **Using the Intel® MCS-51® Boolean Processing Capabilities,** publication number 121519.

5. ON-CHIP PERIPHERAL FUNCTION OPERATION AND INTERFACING

I/O Ports

The I/O port versatility results from the "quasi-bidirectional" output structure depicted in Figure 22. (This is effectively the structure of ports 1, 2, and 3 for normal I/O operations. On port 0 resistor R2 is disabled except during multiplexed bus operations, providing

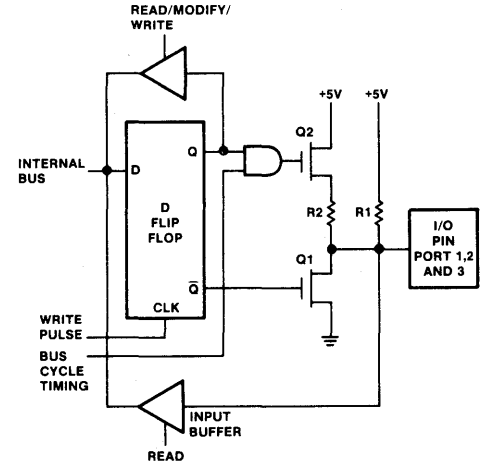


Figure 22. Pseudo-bidirectional I/O port circuitry

AFN-01502A

essentially open-collector outputs. For full electrical characteristics see the User's Manual.)

An output latch bit associated with each pin is updated by direct addressing instructions when that port is the destination. The latch state is buffered to the outside world by R1 and Q1, which may drive a standard TTL input. (In TTL terms, Q1 and R1 resemble an open-collector output with a pull-up resistor to Vcc.)

R2 and Q2 represent an "active pull-up" device enabled momentarily when a 0 previously output changes to a 1. This "jerks" the output pin to a 1 level more quickly than the passive pull-up, improving rise-time significantly if the pin is driving a capacitive load. Note that the active pull-up is **only** activated on 0-to-1 transitions at the output latch (unlike the 8048, in which Q2 is activated whenever a 1 is written out).

Operations using an input port or pin as the source operand use the logic level of the pin itself, rather than the output latch contents. This level is affected by both the microcomputer itself and whatever device the pin is connected to externally. The value read is essentially the "OR-tied" function of Q1 and the external device. If the external device is high-impedence, such as a logic gate input or a three state output in the third state, then reading a pin will reflect the logic level previously output. To use a pin for input, the corresponding output latch must be set. The external device may then drive the pin with either a high or low logic signal. Thus the same port may be used as both input and output by writing ones to all pins used as inputs on output operations, and ignoring all pins used as output on an input operation.

In one operand instructions (INC, DEC, DJNZ and the Boolean CPL) the output latch rather than the input pin level is used as the source data. Similarly, two operand instructions using the port as both one source and the destination (ANL, ORL, XRL) use the output latches. This ensures that latch bits corresponding to pins used as inputs will not be cleared in the process of executing these instructions.

The Boolean operation JBC tests the output latch bit, rather than the input pin, in deciding whether or not to jump. Like the byte-wise logical operations, Boolean operations which modify individual pins of a port leave the other bits of the output latch unchanged.

A good example of how these modes may play together may be taken from the host-processor interface expected by an 8243 I/O expander. Even though the 8051 does not include 8048-type instructions for interfacing with an 8243, the parts can be interconnected (Figure 23) and the protocol may be emulated with simple software.

Example 30—Mixing Parallel Output, Input, and Control Strobes on Port 2

```
; IN8243 INPUT DATA FROM AN 8243 I/O EXPANDER
;       CONNECTED TO P23-P20.
;       P25 & P24 MIMIC CS~/ & PROG.
;       P27-P26 USED AS INPUTS.
;       PORT TO BE READ IN ACC
;
IN8243: ORL     A,#11010000B
        MOV     P2,A        ;OUTPUT INSTRUCTION CODE
        CLR     P2.4        ;FALLING EDGE OF PROG
        ORL     P2,#00001111B   ;SET FOR INPUT
        MOV     A,P2        ;READ INPUT DATA
        SETB    P2.4        ;RETURN PROG HIGH
        SETB    P2.5        ;DE-SELECT CHIP
```

Serial Port and Timer applications

Configuring the 8051's Serial Port for a given data rate and protocol requires essentially three short sections of software. On power-up or hardware reset the serial port and timer control words must be initialized to the appropriate values. Additional software is also needed in the transmit routine to load the serial port data register and in the receive routine to unload the data as it arrives.

This is best illustrated through an arbitrary example. Assume the 8051 will communicate with a CRT operating at 2400 baud (bits per second). Each character is transmitted as seven data bits, odd parity, and one stop bit. This results in a character rate of 2400/10=240 characters per second.

For the sake of clarity, the transmit and receive subroutines are driven by simple-minded software status polling code rather than interrupts. (It might help to refer back to Figures 7-9 showing the control word formats.) The serial port must be initialized to 8-bit UART mode (M0, M1=01), enabled to receive all messages (M2=0, REN=1). The flag indicating that the transmit register is free for more data will be artificially set in order to let the output software know the output register is available. This can all be set up with one instruction.

Example 31—Serial Port Mode and Control Bits

```
;SPINIT INITIALIZE SERIAL PORT
;       FOR 8-BIT UART MODE
;       & SET TRANSMIT READY FLAG.
;
SPINIT: MOV     SCON,#01010010B
```

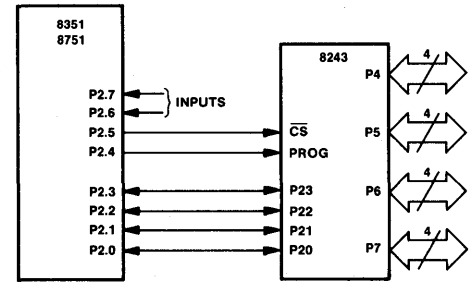


Figure 23. Connecting an 8051 with an 8243 I/O Expander

AFN-01502A

Timer 1 will be used in auto-reload mode as a data rate generator. To achieve a data rate of 2400 baud, the timer must divide the 1 MHz internal clock by 32 x (desired data rate):

$$\frac{1 \times 10^6}{(32)\ (2400)}$$

which equals 13.02 rounded down to 13 instruction cycles. The timer must reload the value -13, or 0F3H. (ASM51 will accept both the signed decimal or hexadecimal representations.)

Example 32—Initializing Timer Mode and Control Bits

```
;T1INIT INITIALIZE TIMER 1 FOR
;       AUTO-RELOAD AT 32*2400 HZ.
;       (TO USED AS GATED 16-BIT COUNTER.)
;
T1INIT: MOV     TCON,#11010010B
        MOV     TH1,#-13
        SETB    TR1
        ...     .....
```

A simple subroutine to transmit the character passed to it in the accumulator must first compute the parity bit, insert it into the data byte, wait until the transmitter is available, output the character, and return. This is nearly as easy said as done.

Example 33—Code for UART Output, Adding Parity, Transmitter Loading

```
;SP_OUT ADD ODD PARITY TO ACC AND
;       TRANSMIT WHEN SERIAL PORT READY.
;
SP_OUT: MOV     C,P
        CPL     C
        MOV     ACC.7,C
        JNB     TI,$
        CLR     TI
        MOV     SBUF,A
        RET
```

A simple minded routine to wait until a character is received, set the carry flag if there is an odd-parity error, and return the masked seven-bit code in the accumulator is equally short.

Example 34—Code for UART Reception and Parity Verification

```
;SP_IN  INPUT NEXT CHARACTER FROM SERIAL PORT.
;       SET CARRY IFF ODD-PARITY ERROR.
;
SP_IN:  JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        MOV     C,P
        CPL     C
        ANL     A,#7FH
        RET
```

6. SUMMARY

This Application Note has described the architecture, instruction set, and on-chip peripheral features of the first three members of the MCS-51™ microcomputer family. The examples used throughout were admittedly (and necessarily) very simple. Additional examples and techniques may be found in the MCS-51™ User's Manual and other application notes written for the MCS-48™ and MCS-51™ families.

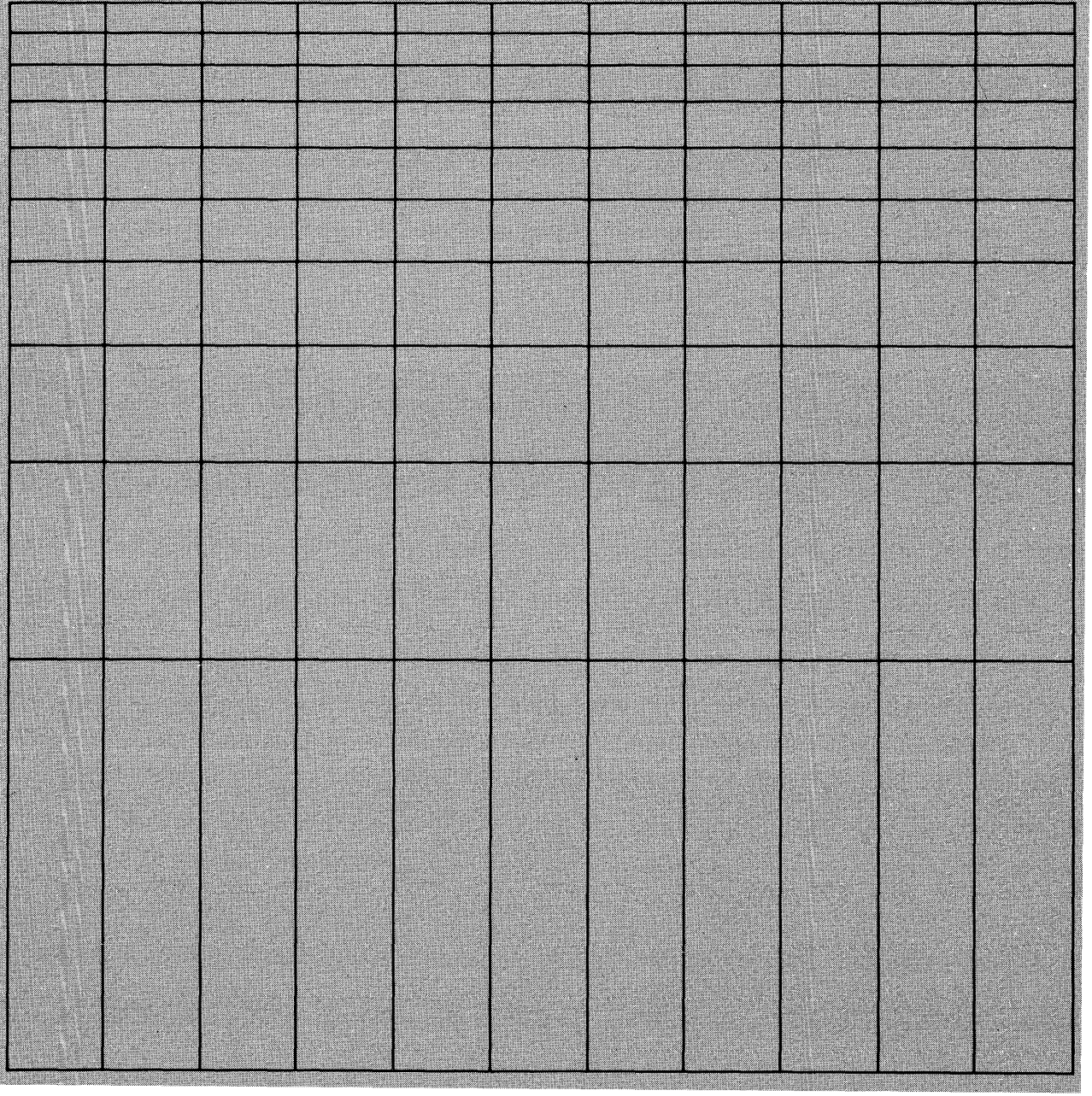
Since its introduction in 1977, the MCS-48™ family has become the industry standard single-chip microcomputer. The MCS-51™ architecture expands the addressing capabilities and instruction set of its predecessor while ensuring flexibility for the future, and maintaining basic software compatability with the past.

Designers already familiar with the 8048 or 8049 will be able to take with them the education and experience gained from past designs as ever-increasing system performance demands force them to move on to state-of-the-art products. Newcomers will find the power and regularity of the 8051 instruction set an advantage in streamlining both the learning and design processes.

Microcomputer system designers will appreciate the 8051 as basically a single-chip solution to many problems which previously required board-level computers. Designers of real-time control systems will find the high execution speed, on-chip peripherals, and interrupt capabilities vital in meeting the timing constraints of products previously requiring discrete logic designs. And designers of industrial controllers will be able to convert ladder diagrams directly from tested-and-true TTL or relay-logic designs to microcomputer software, thanks to the unique Boolean processing capabilities.

It has not been the intent of this note to gloss over the difficulty of designing microcomputer-based systems. To be sure, the hardware and software design aspects of any new computer system are nontrivial tasks. However, the system speed and level of integration of the MCS-51™ microcomputers, the power and flexibility of the instruction set, and the sophisticated assembler and other support products combine to give both the hardware and software designer as much of a head start on the problem as possible.

Appendix C
Using the Intel® MCS-51® Boolean Processing Capabilities



1. INTRODUCTION

The Intel microcontroller family now has three new members — the Intel® 8031, 8051, and 8751 single-chip microcomputers. These devices, shown in Figure 1, will allow whole new classes of products to benefit from recent advances in Integrated Electronics. Thanks to Intel's new HMOS® technology, they provide larger program and data memory spaces, more flexible I/O and peripheral capabilities, greater speed, and lower system cost than any previous-generation single-chip microcomputer.

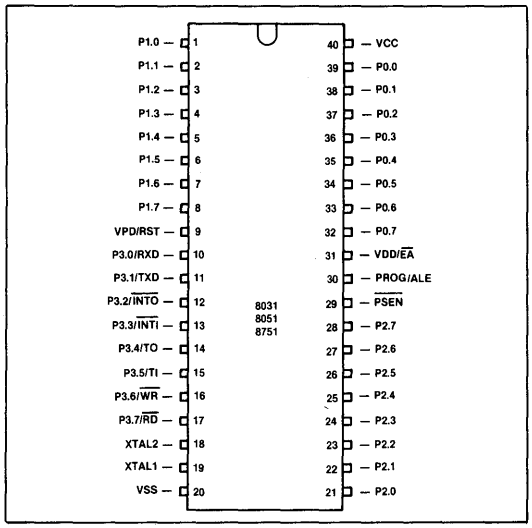


Figure 1. 8051 Family Pinout Diagram.

Table 1 summarizes the quantitative differences between the members of the MCS-48™ and 8051 families. The 8751 contains 4K bytes of EPROM program memory fabricated on-chip, while the 8051 replaces the EPROM with 4K bytes of lower-cost mask-programmed ROM. The 8031 has no program memory on-chip; instead, it accesses up to 64K bytes of program memory from external memory. Otherwise, the three new family members are identical. Throughout this Note, the term "8051" will represent all members of the 8051 Family, unless specifically stated otherwise.

The CPU in each microcomputer is one of the industry's fastest and most efficient for numerical calculations on byte operands. But controllers often deal with bits, not bytes: in the real world, switch contacts can only be open or closed, indicators should be either lit or dark, motors are either turned on or off, and so forth. For such control situations the most significant aspect of the MCS-51™ architecture is its complete hardware support for one-bit, or *Boolean* variables (named in honor of Mathematician George Boole) as a separate data type.

The 8051 incorporates a number of special features which support the direct manipulation and testing of individual bits and allow the use of single-bit variables in performing logical operations. Taken together, these features are referred to as the MCS-51™ *Boolean Processor*. While the bit-processing capabilities alone would be adequate to solve many control applications, their true power comes when they are used in conjunction with the microcomputer's byte-processing and numerical capabilities.

Many concepts embodied by the Boolean Processor will certainly be new even to experienced microcomputer system designers. The purpose of this Application Note is to explain these concepts and show how they are used. It is assumed the reader has read Application Note AP-69, **An Introduction to the Intel® MCS-51™ Single-Chip Microcomputer Family,** publication number 121518, or has been exposed to Intel's single-chip microcomputer product lines.

For detailed information on these parts refer to the **Intel MCS-51™ Family User's Manual**, publication number 121517. The instruction set, assembly language, and use of the 8051 assembler (ASM51) are further described in the **MCS-51™ Macro Assembler User's Guide**, publication number 9800937.

2. BOOLEAN PROCESSOR OPERATION

The Boolean Processing capabilities of the 8051 are based on concepts which have been around for some time. Digital computer systems of widely varying designs all have four functional elements in common (Figure 2):

Table 1. Features of Intel's Single-chip Microcomputers.

EPROM Program Memory	ROM Program Memory	External Program Memory	Program Memory (Int/Max)	Data Memory (Bytes)	Instr. Cycle Time	Input/ Output Pins	Interrupt Sources	Reg. Banks
—	8021	—	1K/1K	64	10 μSec	21	0	1
—	8022	—	2K/2K	64	10 μSec	28	2	1
8748	8048	8035	1K/4K	64	2.5 μSec	27	2	2
—	8049	8039	2K/4K	128	1.36μSec	27	2	2
8751	8051	8031	4K/64K	128	1.0 μSec	32	5	4

01489A

- a central processor (CPU) with the control, timing, and logic circuits needed to execute stored instructions;
- a memory to store the sequence of instructions making up a program or algorithm;
- data memory to store variables used by the program; and
- some means of communicating with the outside world.

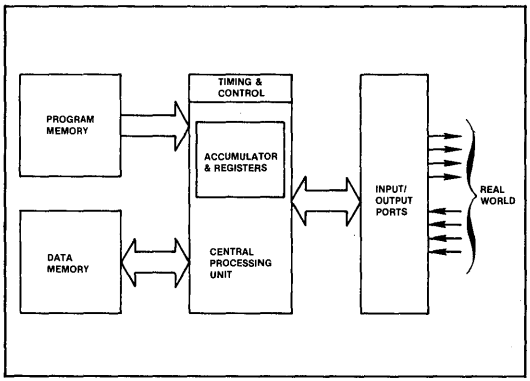


Figure 2. Block Diagram for Abstract Digital Computer.

The CPU usually includes one or more accumulators or special registers for computing or storing values during program execution. The instruction set of such a processor generally includes, at a minimum, operation classes to perform arithmetic or logical functions on program variables, move variables from one place to another, cause program execution to jump or conditionally branch based on register or variable states, and instructions to call and return from subroutines. The program and data memory functions sometimes share a single memory space, but this is not always the case. When the address spaces are separated, program and data memory need not even have the same basic word width.

A digital computer's flexibility comes in part from combining simple fast operations to produce more complex (albeit slower) ones, which in turn link together eventually solving the problem at hand. A four-bit CPU executing multiple precision subroutines can, for example, perform 64-bit addition and subtraction. The subroutines could in turn be building blocks for floating-point multiplication and division routines. Eventually, the four-bit CPU can simulate a far more complex "virtual" machine.

In fact, *any* digital computer with the above four functional elements can (given time) complete *any* algorithm (though the proverbial room full of chimpanzees at word processors might first re-create Shakespeare's classics and this Application Note)! This fact offers little consolation to product designers who want programs to run as quickly as possible. By definition, a real-time control algorithm *must* proceed quickly enough to meet the preordained speed constraints of other equipment.

One of the factors determining how long it will take a microcomputer to complete a given chore is the number of instructions it must execute. What makes a given computer architecture particularly well-or poorly-suited for a class of problems is how well its instruction set matches the tasks to be performed. The better the "primative" operations correspond to the steps taken by the control algorithm, the lower the number of instructions needed, and the quicker the program will run. All else being equal, a CPU supporting 64-bit arithmetic directly could clearly perform floating-point math faster than a machine bogged-down by multiple-precision subroutines. In the same way, direct support for bit manipulation naturally leads to more efficient programs handling the binary input and output conditions inherent in digital control problems.

Processing Elements

The introduction stated that the 8051's bit-handling capabilities alone would be sufficient to solve some control applications. Let's see how the four basic elements of a digital computer - a CPU with associated registers, program memory, addressable data RAM, and I/O capability - relate to Boolean variables.

CPU. The 8051 CPU incorporates special logic devoted to executing several bit-wide operations. All told, there are 17 such instructions, all listed in Table 2. Not shown are 94 other (mostly byte-oriented) 8051 instructions.

Program Memory. Bit-processing instructions are fetched from the same program memory as other arithmetic and logical operations. In addition to the instructions of Table 2, several sophisticated program control features like multiple addressing modes, subroutine nesting, and a two-level interrupt structure are useful in structuring Boolean Processor-based programs.

Boolean instructions are one, two, or three bytes long, depending on what function they perform. Those involving only the carry flag have either a single-byte opcode or an opcode followed by a conditional-branch destination byte (Figure 3.a). The more general instructions add a "direct address" byte after the opcode to specify the bit affected, yielding two or three byte encodings (Figure 3.b). Though this format allows potentially 256 directly addressable bit locations, not all of them are implemented in the 8051 family.

01489A

Table 2. MCS-51™ Boolean Processing Instruction Subset.

Mnemonic		Description	Byte	Cyc
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
MOV	C,bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2
ANL	C,bit	AND direct bit to Carry flag	2	2
ANL	C,/bit	AND complement of direct bit to Carry flag	2	2
ORL	C,bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry flag	2	2
JC	rel	Jump if Carry is flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2

Address mode abbreviations:

C — Carry flag.

bit — 128 software flags, any I/O pin, control or status bit

rel — All conditional jumps include an 8-bit offset byte. Range is +127/-128 bytes relative to first byte of the following instruction.

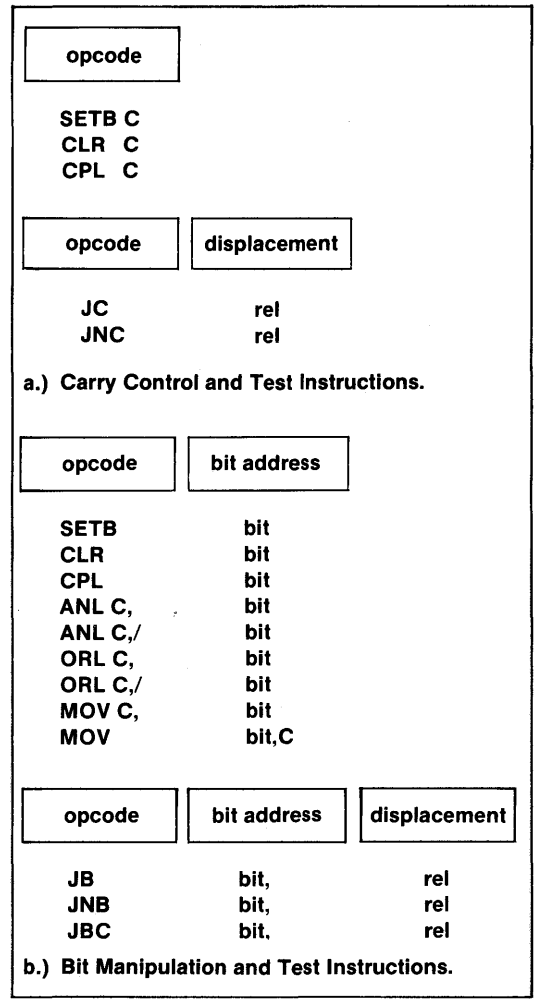


Figure 3. Bit Addressing Instruction Formats.

Data Memory. The instructions in Figure 3.b can operate directly upon 144 general purpose bits forming the Boolean processor "RAM." These bits can be used as sofware flags or to store program variables. Two operand instructions use the CPU's carry flag ("C") as a special one-bit register; in a sense, the carry is a "Boolean accumulator" for logical operations and data transfers.

Input/Output. All 32 I/O pins can be addressed as individual inputs, outputs, or both, in any combination. Any pin can be a control strobe output, status (Test) input, or serial I/O link implemented via software. An additional 33 individually addressable bits reconfigure, control, and monitor the status of the CPU and all on-chip peripheral functions (timer/counters, serial port modes, interrupt logic, and so forth).

Direct Bit Addressing

The most significant bit of the direct address byte selects one of two groups of bits. Values between 0 and 127 (00H and 7FH) define bits in a block of 32 bytes of on-chip RAM, between RAM addresses 20H and 2FH (Figure 4.a). They are numbered consecutively from the lowest-order byte's lowest-order bit through the highest-order byte's highest-order bit.

Bit addresses between 128 and 255(80H and 0FFH) correspond to bits in a number of special registers, mostly used for I/O or peripheral control. These positions are numbered with a different scheme than RAM: the five high-order address bits match those of the register's own address, while the three low-order bits identify the bit position within that register (Figure 4.b).

01489A

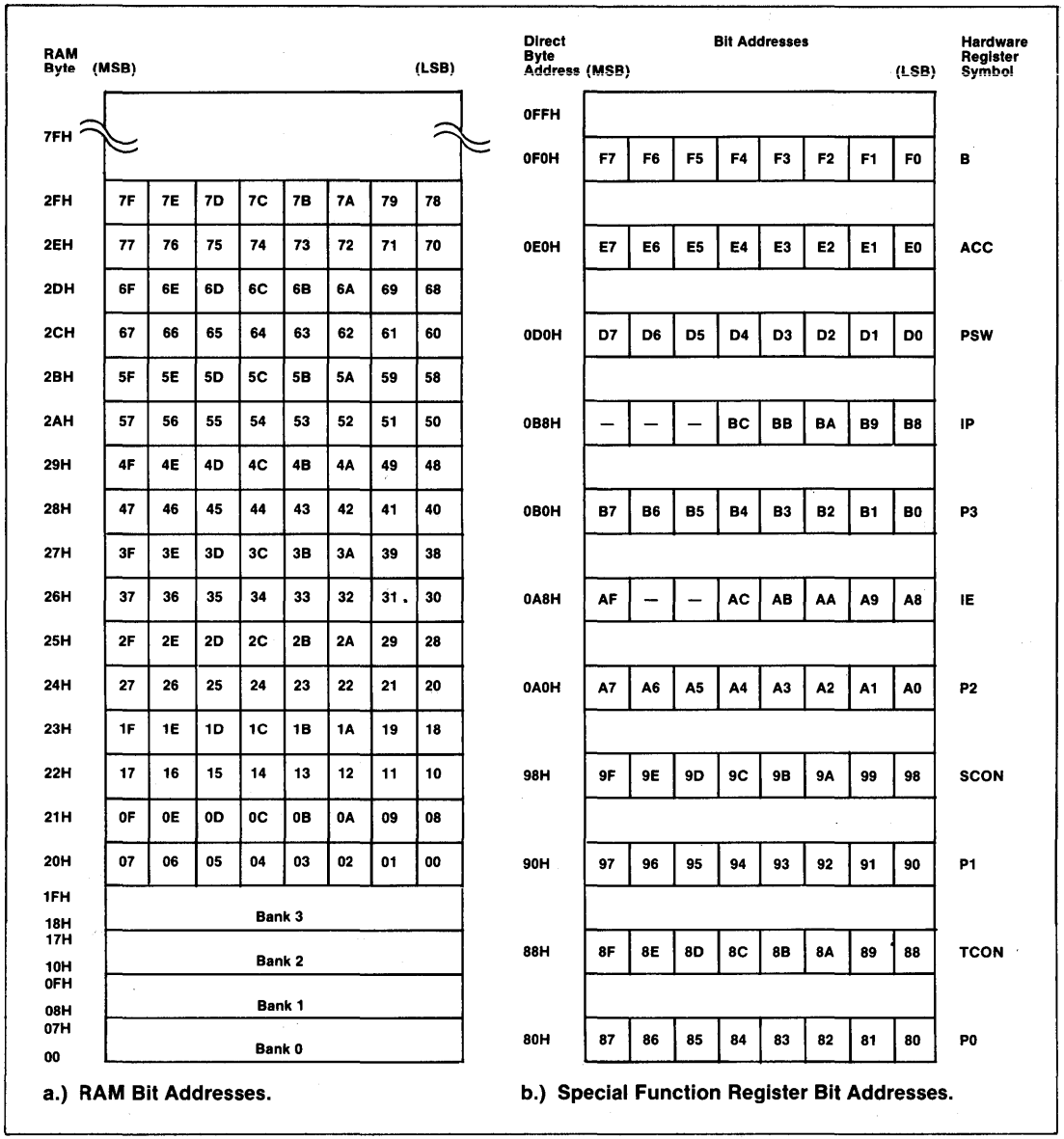


Figure 4. Bit Address Maps.

Notice the column labeled "Symbol" in Figure 5. Bits with special meanings in the PSW and other registers have corresponding symbolic names. General-purpose (as opposed to carry-specific) instructions may access the carry like any other bit by using the mnemonic CY in place of C, P0, P1, P2, and P3 are the 8051's four I/O ports; secondary functions assigned to each of the eight pins of P3 are shown in Figure 6.

Figure 7 shows the last four bit addressable registers. TCON (Timer Control) and SCON (Serial port Control) control and monitor the corresponding peripherals, while IE (Interrupt Enable) and IP (Interrupt Priority) enable and prioritize the five hardware interrupt sources. Like the reserved hardware register addresses, the five bits not implemented in IE and IP should not be accessed; they can *not* be used as software flags.

01489A

(MSB)							(LSB)
CY	AC	F0	RS1	RS0	OV	—	P

Symbol	Position	Name and significance
CY	PSW.7	Carry flag. Set/cleared by hardware or software during certain arithmetic and logical instructions.
AC	PSW.6	Auxiliary Carry flag. Set/cleared by hardware during addition or subtraction instructions to indicate carry or borrow out of bit 3.
F0	PSW.5	Flag 0. Set/cleared/tested by software as a user-defined status flag.
RS1	PSW.4	Register bank Select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	
OV	PSW.2	Overflow flag. Set/cleared by hardware during arithmetic instructions to indicate overflow conditions.
—	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/-even number of "one" bits in the accumulator, i.e., even parity.

Note - the contents of (RS1, RS0) enable the working register banks as follows:

(0,0) - Bank 0	(00H-07H)
(0,1) - Bank 1	(08H-0FH)
(1,0) - Bank 2	(10H-17H)
(1,1) - Bank 3	(18H-1FH)

Figure 5. PSW - Program Status Word organization.

(MSB)							(LSB)
RD	WR	T1	T0	INT1	INT0	TXD	RXD

Symbol	Position	Name and significance
RD	P3.7	Read data control output. Active low pulse generated by hardware when external data memory is read.
WR	P3.6	Write data control output. Active low pulse generated by hardware when external data memory is written.
T1	P3.5	Timer/counter 1 external input or test pin.
T0	P3.4	Timer/counter 0 external input or test pin.
INT1	P3.3	Interrupt 1 input pin. Low-level or falling-edge triggered.
INT0	P3.2	Interrupt 0 input pin. Low-level or falling-edge triggered.
TXD	P3.1	Transmit Data pin for serial port in UART mode. Clock output in shift register mode.
RXD	P3.0	Receive Data pin for serial port in UART mode. Data I/O pin in shift register mode.

Figure 6. P3 - Alternate I/O Functions of Port 3.

01489A

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol Position Name and significance

TF1 TCON.7 Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.

TR1 TCON.6 Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.

TF0 TCON.5 Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.

TR0 TCON.4 Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.

IE1 TCON.3 Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.

IT1 TCON.2 Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

IE0 TCON.1 Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.

IT0 TCON.0 Interrupt 0 Type control bit. Set/cleared by softrware to specify falling edge/low level triggered external interrupts.

a.) TCON - Timer/Counter Control/status register.

(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Symbol Position Name and significance

SM0 SCON.7 Serial port Mode control bit 0. Set/cleared by software (see note).

SM1 SCON.6 Serial port Mode control bit 1. Set/cleared by software (see note).

SM2 SCON.5 Serial port Mode control bit 2. Set by software to disable reception of frames for which bit 8 is zero.

REN SCON.4 Receiver Enable control bit. Set/cleared by software to enable/disable serial data reception.

TB8 SCON.3 Transmit Bit 8. Set/cleared by hardware to determine state of ninth data bit transmitted in 9-bit UART mode.

RB8 SCON.2 Receive Bit 8. Set/cleared by hardware to indicate state of ninth data bit received.

TI SCON.1 Transmit Interrupt flag. Set by hardware when byte transmitted. Cleared by software after servicing.

RI SCON.0 Receive Interrupt flag. Set by hardware when byte received. Cleared by software after servicing.

Note - the state of (SM0,SM1) selects:
(0,0) - Shift register I/O expansion.
(0,1) - 8 bit UART, variable data rate.
(1,0) - 9 bit UART, fixed data rate.
(1,1) - 9 bit UART, variable data rate.

b.) SCON - Serial Port Control/status register.

Figure 7. Peripheral Configuration Registers.

01489A

(MSB) (LSB)

EA	—	—	ES	ET1	EX1	ET1	EX0

Symbol	Position	Name and significance
EA	IE.7	Enable All control bit. Cleared by software to disable all interrupts, independent of the state of IE.4 - IE.0.
—	IE.6	(reserved)
—	IE.5	
ES	IE.4	Enable Serial port control bit. Set/cleared by software to enable/ disable interrupts from TI or RI flags.
ET1	IE.3	Enable Timer 1 control bit. Set/cleared by software to enable/ disable interrupts from timer/counter 1.
EX1	IE.2	Enable External interrupt 1 control bit. Set/cleared by software to enable/disable interrupts from INT1.
ET0	IE.1	Enable Timer 0 control bit. Set/cleared by software to enable/ disable interrupts from timer/counter 0.
EX0	IE.0	Enable External interrupt 0 control bit. Set/cleared by software to enable/disable interrupts from INT0.

c.) IE - Interrupt Enable Register.

(MSB) (LSB)

—	—	—	PS	PT1	PX1	PT0	PX0

Symbol	Position	Name and significance
—	IP.7	(reserved)
—	IP.6	(reserved)
—	IP.5	(reserved)
PS	IP.4	Serial port Priority control bit. Set/cleared by software to specify high/low priority interrupts for Serial port.
PT1	IP.3	Timer 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 1.
PX1	IP.2	External interrupt 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT1.
PT0	IP.1	Timer 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 0.
PX0	IP.0	External interrupt 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT0.

d.) IP - Interrupt Priority Control Register.

Figure 7. (continued)

Addressable Register Set. There are 20 special function registers in the 8051, but the advantages of bit addressing only relate to the 11 described below. Five potentially bit-addressable register addresses (0C0H, 0C8H, 0D8H, 0E8H, & 0F8H) are being reserved for possible future expansion in microcomputers based on the MCS-51™ architecture. Reading or writing non-existent registers in the 8051 series is pointless, and may cause unpredictable results. Byte-wide logical operations can be used to manipulate bits in all *non*-bit addressable registers and RAM.

The accumulator and B registers (A and B) are normally involved in byte-wide arithmetic, but their individual bits can also be used as 16 general software flags. Added with the 128 flags in RAM, this gives 144 general purpose variables for bit-intensive programs. The program status word (PSW) in Figure 5 is a collection of flags and machine status bits including the carry flag itself. Byte operations acting on the PSW can therefore affect the carry.

01489A

Instruction Set

Having looked at the bit variables available to the Boolean Processor, we will now look at the four classes of instructions that manipulate these bits. It may be helpful to refer back to Table 2 while reading this section.

State Control. Addressable bits or flags may be set, cleared, or logically complemented in one instruction cycle with the two-byte instructions SETB, CLR, and CPL. (The "B" affixed to SETB distinguishes it from the assembler "SET" directive used for symbol definition.) SETB and CLR are analogous to loading a bit with a constant: 1 or 0. Single byte versions perform the same three operations on the carry.

The MCS-51™ assembly language specifies a bit address in any of three ways:

- by a number or expression corresponding to the direct bit address (0-255);
- by the name or address of the register containing the bit, the *dot operator* symbol (a period: "."), and the bit's position in the register (7-0);
- in the case of control and status registers, by the predefined assembler symbols listed in the first columns of Figures 5-7.

Bits may also be given user-defined names with the assembler "BIT" directive and any of the above techniques. For example, bit 5 of the PSW may be cleared by any of the four instructions,

```
USR_FLG BIT   PSW.5     ; User Symbol Definition
;         ...   .....
          CLR   0D5H    ; Absolute Addressing
          CLR   PSW.5   ; Use of Dot Operator
          CLR   F0      ; Pre-Defined Assembler
                        ;              Symbol
          CLR   USR_FLG ; User-Defined Symbol
```

Data Transfers. The two-byte MOV instructions can transport any addressable bit to the carry in one cycle, or copy the carry to the bit in two cycles. A bit can be moved between two arbitrary locations via the carry by combining the two instructions. (If necessary, push and pop the PSW to preserve the previous contents of the carry.) These instructions can replace the multi-instruction sequence of Figure 8, a program structure appearing in controller applications whenever flags or outputs are conditionally switched on or off.

Logical Operations. Four instructions perform the logical-AND and logical-OR operations between the carry and another bit, and leave the results in the carry. The instruction mnemonics are ANL and ORL; the absence or presence of a

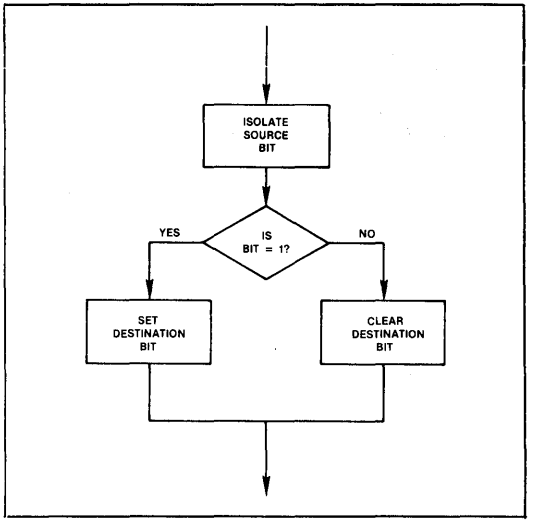


Figure 8. Bit Transfer Instruction Operation.

slash mark ("/") before the source operand indicates whether to use the positive-logic value or the logical complement of the addressed bit. (The source operand itself is never affected.)

Bit-test Instructions. The conditional jump instructions "JC rel" (Jump on Carry) and "JNC rel" (Jump on Not Carry) test the state of the carry flag, branching if it is a one or zero, respectively. (The letters "rel" denote relative code addressing.) The three-byte instructions "JB bit, rel" and "JNB bit,rel" (Jump on Bit and Jump on Not Bit) test the state of *any* addressable bit in a similar manner. A fifth instruction combines the Jump on Bit and Clear operations. "JBC bit,rel" conditionally branches to the indicated address, then clears the bit in the same two cycle instruction. This operation is the same as the MCS-48™ "JTF" instructions.

All 8051 conditional jump instructions use program counter-relative addressing, and all execute in two cycles. The last instruction byte encodes a signed displacement ranging from -128 to +127. During execution, the CPU adds this value to the incremented program counter to produce the jump destination. Put another way, a conditional jump to the immediately following instruction would encode 00H in the offset byte.

A section of program or subroutine written using only relative jumps to nearby addresses will have the same machine code independent of the code's location. An assembled routine may be repositioned anywhere in memory, even crossing memory page boundaries, without having to modify the program or recompute destination addresses. To facilitate this flexibility, there is an unconditional "Short Jump" (SJMP) which uses relative addressing as well. Since a pro-

01489A

grammer would have quite a chore trying to compute relative offset values from one instruction to another, ASM51 automatically computes the displacement needed given only the destination address or label. An error message will alert the programmer if the destination is "out of range."

(The so-called "Bit Test" instructions implemented on many other microprocessors simply perform the logical-AND operation between a byte variable and a constant mask, and set or clear a zero flag depending on the result. This is essentially equivalent to the 8051 "MOV C,bit" instruction. A second instruction is then needed to conditionally branch based on the state of the zero flag. This does *not* constitute abstract bit-addressing in the MCS-51™ sense. A flag exists only as a field within a register; to reference a bit the programmer must know and specify both the encompassing register and the bit's position therein. This constraint severely limits the flexibility of symbolic bit addressing and reduces the machine's code-efficiency and speed.)

Interaction with Other Instructions. The carry flag is also affected by the instructions listed in Table 3. It can be rotated through the accumulator, and altered as a side effect of arithmetic instructions. Refer to the User's Manual for details on how these instructions operate.

Simple Instruction Combinations

By combining general purpose bit operations with certain addressable bits, one can "custom build" several hundred useful instructions. All eight bits of the PSW can be tested directly with conditional jump instructions to monitor (among other things) parity and overflow status. Programmers can take advantage of 128 software flags to keep track of operating modes, resource usage, and so forth.

The Boolean instructions are also the most efficient way to control or reconfigure peripheral and I/O registers. All 32 I/O lines become "test pins," for example, tested by conditional jump instructions. Any output pin can be toggled (complemented) in a single instruction cycle. Setting or clearing the Timer Run flags (TR0 and TR1) turn the timer-/counters on or off; polling the same flags elsewhere lets the program determine if a timer is running. The respective overflow flags (TF0 and TF1) can be tested to determine when the desired period or count has elapsed, then cleared in preparation for the next repetition. (For the record, these bits are all part of the TCON register, Figure 7.a. Thanks to symbolic bit addressing, the programmer only needs to remember the mnemonic associated with each function. In other words, don't bother memorizing control word layouts.)

In the MCS-48® family, instructions corresponding to some of the above functions require specific opcodes. Ten different opcodes serve to clear/complement the software flags F0 and F1, enable/disable each interrupt, and start/stop the timer. In the 8051 instruction set, just three opcodes (SETB,

Table 3. Other Instructions Affecting the Carry Flag.

Mnemonic	Description	Byte	Cyc
ADD A,Rn	Add register to Accumulator	1	1
ADD A,direct	Add direct byte to Accumulator	2	1
ADD A,@Ri	Add indirect RAM to Accumulator	1	1
ADD A,#data	Add immediate data to Accumulator	2	1
ADDC A,Rn	Add register to Accumulator with Carry flag	1	1
ADDC A, direct	Add direct byte to Accumulator with Carry flag	2	1
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry flag	1	1
ADDC A,#data	Add immediate data to Acc with Carry flag	2	1
SUBB A,Rn	Subtract register from Accumulator with borrow	1	1
SUBB A,direct	Subtract direct byte from Acc with borrow	2	1
SUBB A,@Ri	Subtract indirect RAM from Acc with borrow	1	1
SUBB A,#data	Subtract immediate data from Acc with borrow	2	1
MUL AB	Multiply A & B	1	4
DIV AB	Divide A by B	1	4
DA A	Decimal Adjust Accumulator	1	1
RLC A	Rotate Accumulator Left through the Carry flag	1	1
RRC A	Rotate Accumulator Right through Carry flag	1	1
CJNE A,direct,rel	Compare direct byte to Acc & Jump if Not Equal	3	2
CJNE A,#data,rel	Compare immediate to Acc & Jump if Not Equal	3	2
CJNE Rn,#data,rel	Compare immed to register & Jump if Not Equal	3	2
CJNE @Ri,#data,rel	Compare immed to indirect & Jump if Not Equal	3	2

CLR, CPL) with a direct bit address appended perform the same functions. Two test instructions (JB and JNB) can be combined with bit addresses to test the software flags, the 8048 I/O pins T0, T1, and INT, and the eight accumulator bits, replacing 15 more different instructions.

Table 4.a shows how 8051 programs implement software flag and machine control functions associated with special

using awkward sequences of other basic operations. As mentioned earlier, any CPU can solve any problem given enough time.

Quantitatively, the differences between a solution allowed by the 8051 and those required by previous architectures are numerous. What the 8051 Family buys you is a faster, cleaner, lower-cost solution to microcontroller applications.

The opcode space freed by condensing many specific 8048

Table 4.a. Contrasting 8048 and 8051 Bit Control and Testing Instructions.

8048 Instruction	Bytes	Cycles	uSec	8x51 Instruction	Bytes	Cycles & uSec
Flag Control						
CLR C	1	1	2.5	CLR C	1	1
CPL F0	1	1	2.5	CPL F0	2	1
Flag Testing						
JNC offset	2	2	5.0	JNC rel	2	2
JF0 offset	2	2	5.0	JB F0,rel	3	2
JB7 offset	2	2	5.0	JB ACC.7,rel	3	2
Peripheral Polling						
JT0 offset	2	2	5.0	JB T0,rel	3	2
JNI offset	2	2	5.0	JNB INT0,rel	3	2
JTF offset	2	2	5.0	JBC TF0,rel	3	2
Machine and Peripheral Control						
STRT T	1	1	2.5	SETB TR0	2	1
EN I	1	1	2.5	SETB EX0	2	1
DIS TCNTI	1	1	2.5	CLR ET0	2	1

Table 4.b. Replacing 8048 instruction sequences with single 8x51 instructions.

8048 Instructions	Bytes	Cycles	uSec	8051 Instructions	Bytes	Cycles & uSec
Flag Control Set carry: CLR C CPL C	= 2	2	5.0	SETB C	1	1
Set Software Flag: CLR F0 CPL F0	= 2	2	5.0	SETB F0	2	1

opcodes in the 8048. In every case the MCS-51™ solution requires the same number of machine cycles, and executes 2.5 times faster.

3. BOOLEAN PROCESSOR APPLICATIONS

So what? Then what does all this buy you?

Qualitatively, nothing. All the same capabilities *could* be (and often have been) implemented on other machines

instructions into a few general operations has been used to add new functionality to the MCS-51™ architecture - both for byte and bit operations. 144 software flags replace the 8048's two. These flags (and the carry) may be directly set, not just cleared and complemented, and all can be tested for either state, not just one. Operating mode bits previously inaccessible may be read, tested, or saved. Situations where the 8051 instruction set provides new capabilities are contrasted with 8048 instruction sequences in Table 4.b. Here the 8051 speed advantage ranges from 5x to 15x!

01489A

Table 4b. (Continued)

8048 Instructions		Bytes	Cycles	uSec	8x51 Instructions		Bytes	Cycles & uSec
Turn Off Output Pin:								
ANL P1,#0FBH	=	2	2	5.0	CLR	P1.2	2	1
Complement Output Pin:								
IN A,P1								
XRL A,#04H								
OUTL P1,A	=	4	6	15.0	CPL	P1.2	2	1
Clear Flag in RAM:								
MOV R0,#FLGADR								
MOV A,@R0								
ANL A,#FLGMASK								
MOV @R0,A	=	6	6	15.0	CLR	USER_FLG	2	1
Flag Testing								
Jump if Software Flag is 0:								
JF0 $+4								
JMP offset	=	4	4	10.0	JNB	F0,rel	3	2
Jump if Accumulator bit is 0:								
CPL A								
JB7 offset								
CPL A	=	4	4	10.0	JNB	ACC.7,rel	3	2
Peripheral Polling								
Test if Input Pin is Grounded:								
IN A,P1								
CPL A								
JB3 offset	=	4	5	12.5	JNB	P1.3,rel	3	2
Test if Interrupt Pin is High:								
JNI $+4								
JMP offset	=	4	4	10.0	JB	INT0,rel	3	2

Combining Boolean and byte-wide instructions can produce great synergy. An MCS-51™ based application will prove to be:

- simpler to write since the architecture correlates more closely with the problems being solved;
- easier to debug because more individual instructions have no unexpected or undesirable side-effects;
- more byte efficient due to direct bit addressing and program counter relative branching;
- faster running because fewer bytes of instruction need to be fetched and fewer conditional jumps are processed;
- lower cost because of the high level of system-intergration within one component.

These rather unabashed claims of excellence shall not go unsubstantiated. The rest of this chapter examines less trivial tasks simplified by the Boolean processor. The first three compare the 8051 with other microprocessors; the last two go into 8051-based system designs in much greater depth.

Design Example #1 - Bit Permutation

First off, we'll use the bit-transfer instructions to permute a lengthy pattern of bits.

A steadily increasing number of data communication products use encoding methods to protect the security of sensitive information. By law, interstate financial transactions involving the Federal banking system must be transmitted using the Federal Information Processing *Data Encryption Standard* (DES).

Basically, the DES combines eight bytes of "plaintext" data (in binary, ASCII, or any other format) with a 56-bit "key", producing a 64-bit encrypted value for transmission. At the receiving end the same algorithm is applied to the incoming data using the same key, reproducing the original eight byte message. The algorithm used for these permutations is fixed; different user-defined keys ensure data privacy.

01489A

It is not the purpose of this note to describe the DES in any detail. Suffice it to say that encryption/decryption is a long, iterative process consisting of rotations, exclusive -OR operations, function table look-ups, and an extensive (and quite bizarre) sequence of bit permutation, packing, and unpacking steps. (For further details refer to the June 21, 1979 issue of **Electronics** magazine.) The bit manipulation steps are included, it is rumored, to impede a general purpose digital supercomputer trying to "break" the code. Any algorithm implementing the DES with previous generation microprocessors would spend virtually all of its time diddling bits.

The bit manipulation performed is typified by the Key Schedule Calculation represented in Figure 9. This step is repeated 16 times for each key used in the course of a transmission. In essence, a seven-byte, 56-bit "Shifted Key Buffer" is transformed into an eight-byte, "Permutation Buffer" without altering the shifted Key. The arrows in Figure 9 indicate a few of the translation steps. Only six bits of each byte of the Permutation Buffer are used; the two high-order bits of each byte are cleared. This means only 48 of the 56 Shifted Key Buffer bits are used in any one iteration.

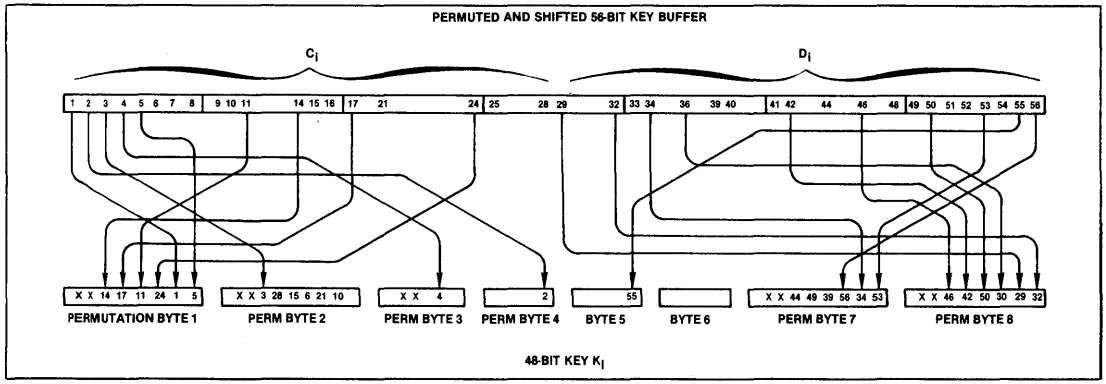


Figure 9. DES Key Schedule Transformation.

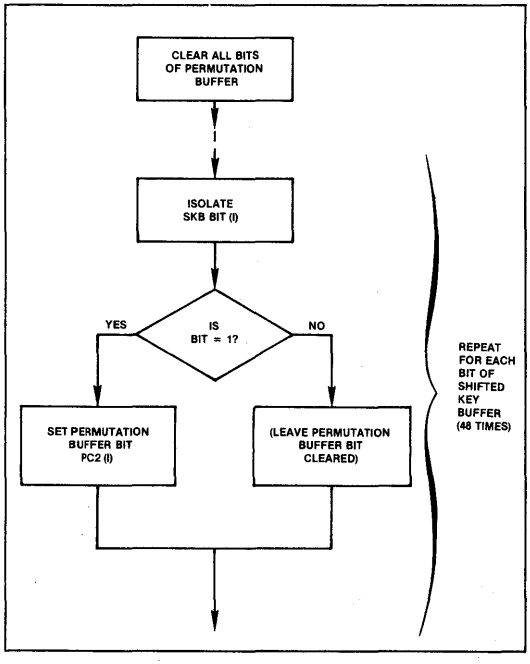


Figure 10.a. Flowchart for Key permutation attempted with a byte processor.

Different microprocessor architectures would best implement this type of permutation in different ways. Most approaches would share the steps of Figure 10.a:

- Initialize the Permutation Buffer to default state (ones or zeroes);
- Isolate the state of a bit of a byte from the Key Buffer. Depending on the CPU, this might be accomplished by rotating a word of the Key Buffer through a carry flag or testing a bit in memory or an accumulator against a mask byte;
- Perform a conditional jump based on the carry or zero flag if the Permutation Buffer default state is correct;
- Otherwise reverse the corresponding bit in the permutation buffer with logical operations and mask bytes.

Each step above may require several instructions. The last three steps must be repeated for all 48 bits. Most microprocessors would spend 300 to 3,000 microseconds on each of the 16 iterations.

Notice, though, that this flow chart looks a lot like Figure 8. The Boolean Processor can permute bits by simply moving

01489A

them from the source to the carry to the destination—a total of two instructions taking four bytes and three microseconds per bit. Assume the Shifted Key Buffer and Permutation Buffer both reside in bit-addressable RAM, with the bits of the former assigned symbolic names SKB_1, SKB_2, . . . SKB_56, and that the bytes of the latter are named PB_1, . . . PB_8. Then working from Figure 9, the software for the permutation algorithm would be that of Example 1.a. The total routine length would be 192 bytes, requiring 144 microseconds.

The algorithm of Figure 10.b is just slightly more efficient in this time-critical application and illustrates the synergy of an integrated byte and bit processor. The bits needed for each byte of the Permutation Buffer are assimilated by loading each bit into the carry (1 usec.) and shifting it into the accumulator (1 usec.). Each byte is stored in RAM when completed. Forty-eight bits thus need a total of 112 instructions, some of which are listed in Example 1.b.

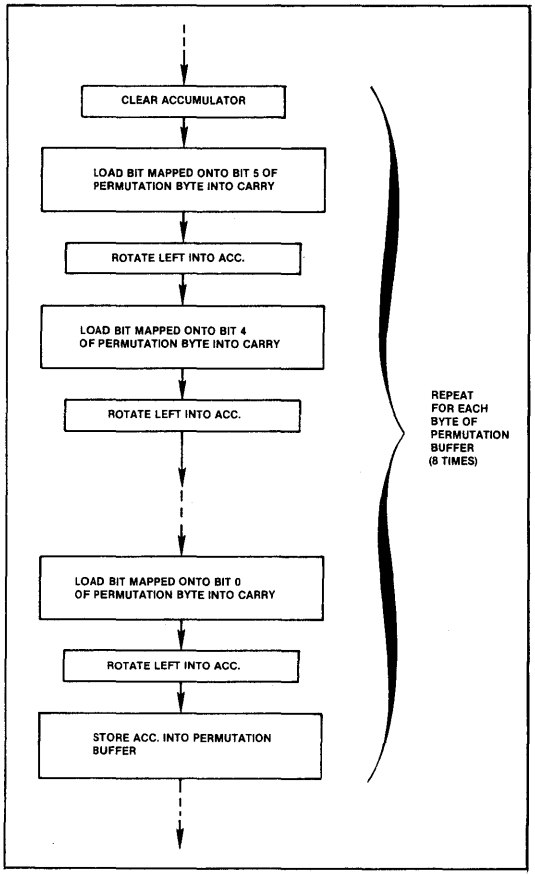


Figure 10.b. DES Key Permutation with Boolean Processor.

Worst-case execution time would be 112 microseconds, since each instruction takes a single cycle. Routine length would also decrease, to 168 bytes. (Actually, in the context of the complete encryption algorithm, each permuted byte would be processed as soon as it is assimilated—saving memory and cutting execution time by another 8 usec.)

Example 1. DES Key Permutation Software.

a.) "Brute Force" technique.

```
        MOV     C,SKB_1
        MOV     PB_1.1,C
        MOV     C,SKB_2
        MOV     PB_4.0,C
        MOV     C,SKB_3
        MOV     PB_2.5,C
        MOV     C,SKB_4
        MOV     PB_1.0,C
;       . . .   . . . . .
;       . . .   . . . . .
        MOV     C,SKB_55
        MOV     PB_5.0,C
        MOV     C,SKB_56
        MOV     PB_7.2,C
```

b.) Using Accumulator to Collect Bits.

```
        CLR     A
        MOV     C,SKB_14
        RLC     A
        MOV     C,SKB_17
        RLC     A
        MOV     C,SKB_11
        RLC     A
        MOV     C,SKB_24
        RLC     A
        MOV     C,SKB_1
        RLC     A
        MOV     C,SKB_5
        RLC     A
        MOV     PB_1,A
;       . . .   . . . . .
;       . . .   . . . . .
        MOV     C,SKB_29
        RLC     A
        MOV     C,SKB_32
        RLC     A
        MOV     PB_8,A
```

To date, most banking terminals and other systems using the DES have needed special boards or peripheral controller chips just for the encryption/decryption process, and still more hardware to form a serial bit stream for transmission (Figure 11.a). An 8051 solution could pack most of the entire system onto the one chip (Figure 11.b). The whole DES algorithm would require less than one-

01489A

fourth of the on-chip program memory, with the remaining bytes free for operating the banking terminal (or whatever) itself.

Moreover, since transmission and reception of data is performed through the on-board UART, the unencrypted data (plaintext) never even exists outside the microcomputer! Naturally, this would afford a high degree of security from data interception.

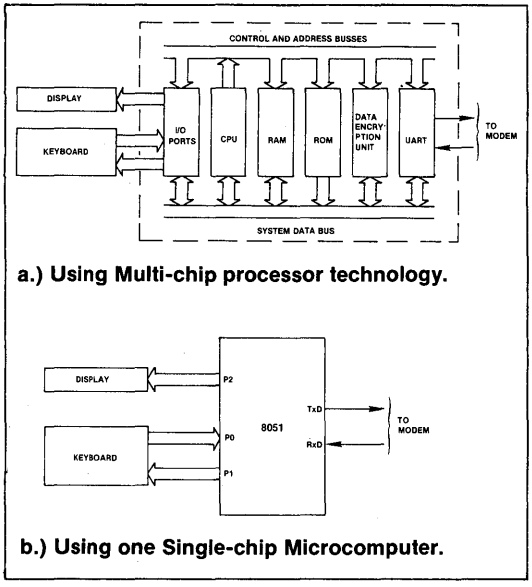


Figure 11. Secure Banking Terminal Block Diagram.

Design Example #2 - Software Serial I/O

An exercise often imposed on beginning microcomputer students is to write a program simulating a UART. (See, for example, Application Notes AP24, AP29, and AP49.) Though doing this with the 8051 Family may appear to be a moot point (given that the hardware for a full UART is on-chip), it is still instructive to see how it would be done, and maintains a product line tradition.

As it turns out, the 8051 microcomputers can receive or transmit serial data via software very efficiently using the Boolean instruction set. Since any I/O pin may be a serial input or output, several serial links could be maintained at once.

Figures 12.a and 12.b show algorithms for receiving or transmitting a byte of data. (Another section of program would invoke this algorithm eight times, synchronizing it with a start bit, clock signal, software delay, or timer interrupt.) Data is received by testing an input pin, setting the carry to the same state, shifting the carry into a data buffer, and saving the partial frame in internal RAM. Data is transmitted by shifting an output buffer through the carry, and generating each bit on an output pin.

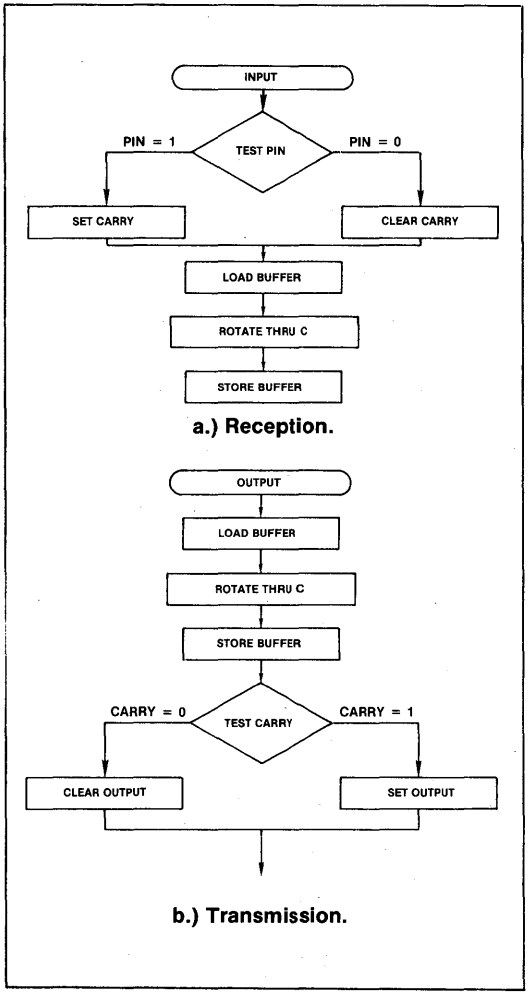


Figure 12. Serial I/O Algorithms.

A side-by-side comparison of the software for this common "bit-banging" application with three different microprocessor architectures is shown in Table 5.a and 5.b. The 8051 solution is more efficient than the others on every count!

01489A

Table 5. Serial I/O Programs for Various Microprocessors.

a.) Input Routine.

```
          8085                    8048                    8051
          IN    SERPORT                                   MOV   C,SERPIN
          ANI   MASK              CLR   C
          JZ    LO                JNT0  LO
          CMC                     CPL   C
LO:       LXI   HL,SERBUF         MOV   R0,#SERBUF
          MOV   A,M               MOV   A,@R0             MOV   A,SERBUF
          RR                      RRC   A                 RRC   A
          MOV   M,A               MOV   @R0,A             MOV   SERBUF,A

RESULTS:
           8 INSTRUCTIONS          7 INSTRUCTIONS          4 INSTRUCTIONS
          14 BYTES                 9 BYTES                 7 BYTES
          56 STATES                9 CYCLES                4 CYCLES
          19 uSEC.                22.5 uSEC.              4 uSEC.
```

b.) Output Routine.

```
          8085                    8048                    8051
          LXI   HL,SERBUF         MOV   R0,#SERBUF
          MOV   A,M               MOV   A,@R0             MOV   A,SERBUF
          RR                      RRC   A                 RRC   A
          MOV   M,A               MOV   @R0,A             MOV   SERBUF,A
          IN    SERPORT
          JC    HI                JC    HI
LO:       ANI   NOT MASK          ANL   SERPRT,#NOT MASK  MOV   SERPIN,C
          JMP   CNT               JMP   CNT
HI:       ORI   MASK        HI:   ORL   SERPRT,#MASK
CNT:      OUT   SERPORT     CNT:

RESULTS:
          10 INSTRUCTIONS          8 INSTRUCTIONS          4 INSTRUCTIONS
          20 BYTES                13 BYTES                 7 BYTES
          72 STATES               11 CYCLES                5 CYCLES
          24 uSEC.                27.5 uSEC.               5 uSEC.
```

Design Example #3 - Combinatorial Logic Equations

Next we'll look at some simple uses for bit-test instructions and logical operations. (This example is also presented in Application Note AP-69.)

Virtually all hardware designers have solved complex functions using combinatorial logic. While the hardware involved may vary from relay logic, vacuum tubes, or TTL or to more esoteric technologies like fluidics, in each case the goal is the same: to solve a problem represented by a logical function of several Boolean variables.

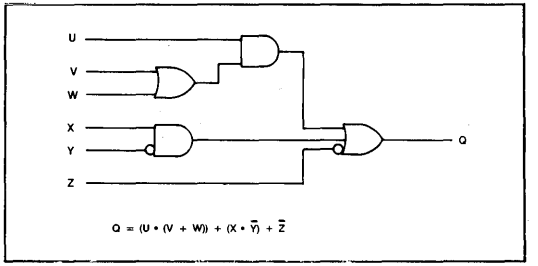
Figure 13 shows TTL and relay logic diagrams for a function of the six variables U through Z. Each is a solution of the equation,

$$Q = (U \cdot (V + W)) + (X \cdot \overline{Y}) + \overline{Z} \quad .$$

Equations of this sort might be reduced using Karnaugh Maps or algebraic techniques, but that is not the purpose of this example. As the logic complexity increases, so does the difficulty of the reduction process. Even a minor change to the function equations as the design evolves would require tedious re-reduction from scratch.

01489A

Figure 13. Hardware Implementations of Boolean functions.



a.) Using TTL:

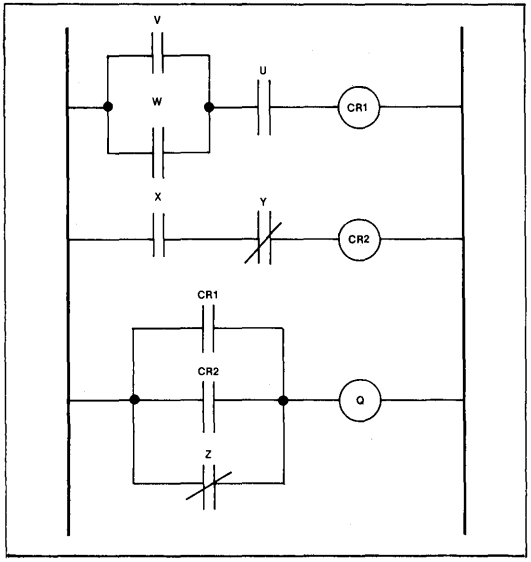
For the sake of comparison we will implement this function three ways, restricting the software to three proper subsets of the MCS-51™ instruction set. We will also assume that U and V are input pins from different input ports, W and X are status bits for two peripheral controllers, and Y and Z are software flags set up earlier in the program. The end result must be written to an output pin on some third port. The first two implementations follow the flow-chart shown in Figure 14. Program flow would embark on a route down a test-and-branch tree and leaves either the "True" or "Not True" exit ASAP — as soon as the proper result has been determined. These exits then rewrite the output port with the result bit respectively one or zero.

Other digital computers must solve equations of this type with standard word-wide logical instructions and conditional jumps. So for the first implementation, we won't use any generalized bit-addressing instructions. As we shall soon see, being constrained to such an instruction subset produces somewhat sloppy software solutions. MCS-51™ mnemonics are used in Example 2.a; other machines might further cloud the situation by requiring operation-specific mnemonics like INPUT, OUTPUT, LOAD, STORE, etc., instead of the MOV mnemonic used for all variable transfers in the 8051 instruction set.

The code which results is cumbersome and error prone. It would be difficult to prove whether the software worked for all input combinations in programs of this sort. Furthermore, execution time will vary widely with input data.

Thanks to the direct bit-test operations, a single instruction can replace each move/mask/conditional jump sequence in Example 2.a, but the algorithm would be equally convoluted (see Example 2.B). To lessen the confusion "a bit" each input variable is assigned a symbolic name.

A more elegant and efficient implementation (Example 2.c) strings together the Boolean ANL and ORL functions to generate the output function with straight-line code.



b.) Using Relay Logic:

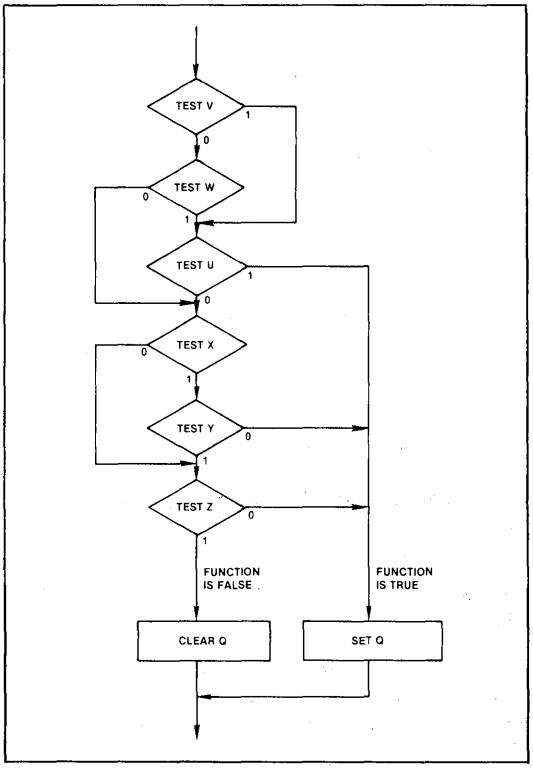


Figure 14. Flow chart for tree-branching algorithm.

01489A

When finished, the carry flag contains the result, which is simply copied out to the destination pin. No flow chart is needed—code can be written directly from the logic diagrams in Figure 14. The result is simplicity itself: fast, flexible, reliable, easy to design, and easy to debug.

An 8051 program can simulate an N-input AND or OR gate with at most N+1 lines of source program—one for each input and one line to store the results. To simulate NAND and NOR gates, complement the carry after computing the function. When some inputs to the gate have "inversion bubbles," perform the ANL or ORL operation on inverted operands. When the first input is inverted, either load the operand into the carry and then complement it, or use DeMorgan's Theorem to convert the gate to a different form.

Example 2. Software Solutions to Logic Function of Figure 13.

a.) Using only byte-wide logical instructions.

```
;BFUNC1 SOLVE RANDOM LOGIC FUNCTION
;       OF 6 VARIABLES BY LOADING AND
;       MASKING THE APPROPRIATE BITS
;       IN THE ACCUMULATOR, THEN
;       EXECUTING CONDITIONAL JUMPS
;       BASED ON ZERO CONDITION.
;       (APPROACH USED BY BYTE-
;       ORIENTED ARCHITECTURES.)
;       BYTE AND MASK VALUES
;       CORRESPOND TO RESPECTIVE BYTE
;       ADDRESS AND BIT POSITIONS.
;
OUTBUF  DATA 22H  ;OUTPUT PIN STATE MAP
;
TESTV;  MOV  A,P2
        ANL  A,#00000100B
        JNZ  TESTU
        MOV  A,TCON
        ANL  A,#00100000B
        JZ   TESTX
TESTU:  MOV  A,P1
        ANL  A,#00000010B
        JNZ  SETQ
TESTX:  MOV  A,TCON
        ANL  A,#00001000B
        JZ   TESTZ
        MOV  A,20H
        ANL  A,#00000001B
        JZ   SETQ
TESTZ:  MOV  A,21H
        ANL  A,#00000010B
        JZ   SETQ
CLRQ:   MOV  A,OUTBUF
        ANL  A,#11110111B
        JMP  OUTQ
SETQ:   MOV  A,OUTBUF
        ORL  A,#00001000B
OUTQ:   MOV  OUTBUF,A
        MOV  P3,A
```

b.) Using only bit-test instructions.

```
;BFUNC2 SOLVE A RANDOM LOGIC FUNCTION
;       OF 6 VARIABLES BY DIRECTLY
;       POLLING EACH BIT.
;       (APPROACH USING MCS-51 UNIQUE
;       BIT-TEST INSTRUCTION CAPABILITY.)
;       SYMBOLS USED IN LOGIC DIAGRAM
;       ASSIGNED TO CORRESPONDING 8x51
;       BIT ADDRESSES.
;
U       BIT   P1.1
V       BIT   P2.2
W       BIT   TF0
X       BIT   IE1
Y       BIT   20H.0
Z       BIT   21H.1
Q       BIT   P3.3
;       ...   ....
TEST_V: JB    V,TEST_U
        JNB   W,TEST_X
TEST_U: JB    U,SET_Q
TEST_X: JNB   X,TEST_Z
        JNB   Y,SET_Q
TEST_Z: JNB   Z,SET_Q
CLR_Q:  CLR   Q
        JMP   NXTTST
SET_Q:  SETB  Q
NXTTST:               ;(CONTINUATION OF
                      ;PROGRAM)
```

c.) Using logical operations on Boolean variables.

```
;FUNC3  SOLVE A RANDOM LOGIC FUNCTION
;       OF 6 VARIABLES USING
;       STRAIGHT_LINE LOGICAL
;       INSTRUCTIONS ON MCS-51 BOOLEAN
;       VARIABLES.
;
MOV  C,V
ORL  C,W   ;OUTPUT OF OR GATE
ANL  C,U   ;OUPUT OF TOP AND GATE
MOV  F0,C  ;SAVE INTERMEDIATE STATE
MOV  C,X
ANL  C,/Y  ;OUTPUT OF BOTTOM AND GATE
ORL  C,F0  ;INCLUDE VALUE SAVED ABOVE
ORL  C,/Z  ;INCLUDE LAST INPUT VARIABLE
MOV  Q,C   ;OUTPUT COMPUTED RESULT
```

01489A

An upper-limit can be placed on the complexity of software to simulate a large number of gates by summing the total number of inputs and outputs. The *actual* total should be somewhat shorter, since calculations can be "chained," as shown above. The output of one gate is often the first input to another, bypassing the intermediate variable to eliminate two lines of source.

Design Example #4 - Automotive Dashboard Functions

Now let's apply these techniques to designing the software for a complete controller system. This application is patterned after a familiar real-world application which isn't nearly as trivial as it might first appear: automobile turn signals.

Imagine the three position turn lever on the steering column as a single-pole, triple-throw toggle switch. In its central position all contacts are open. In the up or down positions contacts close causing corresponding lights in the rear of the car to blink. So far very simple.

Two more turn signals blink in the front of the car, and two others in the dashboard. All six bulbs flash when an emergency switch is closed. A thermo-mechanical relay (accessible under the dashboard in case it wears out) causes the blinking.

Applying the brake pedal turns the tail light filaments on constantly . . . unless a turn is in progress, in which case the blinking tail light is not affected. (Of course, the front turn signals and dashboard indicators are not affected by the brake pedal.) Table 6 summarizes these operating modes.

But we're not done yet. Each of the exterior turn signal (but not the dashboard) bulbs has a second, somewhat dimmer filament for the parking lights. Figure 15 shows TTL circuitry which could control all six bulbs. The signals labeled "High Freq." and "Low Freq." represent two square-wave inputs. Basically, when one of the turn switches is closed or the emergency switch is activated the low frequency signal (about 1 Hz) is gated through to the appropriate dashboard indicator(s) and turn signal(s). The rear signals are also activated when the brake pedal is depressed provided a turn is not being made in the same direction. When the parking light switch is closed the higher frequency oscillator is gated to each front and rear turn signal, sustaining a low-intensity background level. (This is to eliminate the need for additional parking light filaments.)

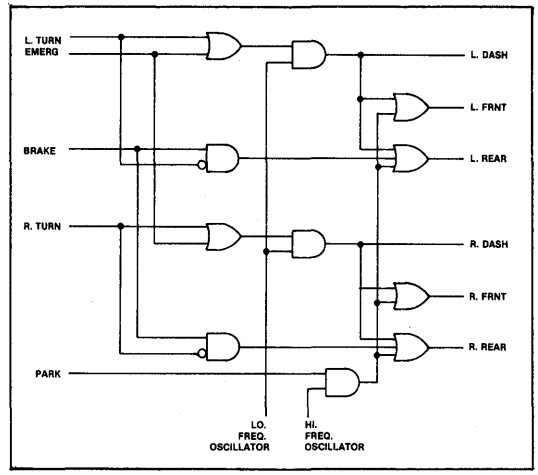


Figure 15. TTL logic implementation of automotive turn signals.

Table 6. Truth table for turn-signal operation.

INPUT SIGNALS				OUTPUT SIGNALS			
BRAKE SWITCH	EMERG. SWITCH	LEFT TURN SWITCH	RIGHT TURN SWITCH	LEFT FRONT & DASH	RIGHT FRONT & DASH	LEFT REAR	RIGHT REAR
0	0	0	0	OFF	OFF	OFF	OFF
0	0	0	1	OFF	BLINK	OFF	BLINK
0	0	1	0	BLINK	OFF	BLINK	OFF
0	1	0	0	BLINK	BLINK	BLINK	BLINK
0	1	0	1	BLINK	BLINK	BLINK	BLINK
0	1	1	0	BLINK	BLINK	BLINK	BLINK
1	0	0	0	OFF	OFF	ON	ON
1	0	0	1	OFF	BLINK	ON	BLINK
1	0	1	0	BLINK	OFF	BLINK	ON
1	1	0	0	BLINK	BLINK	ON	ON
1	1	0	1	BLINK	BLINK	ON	BLINK
1	1	1	0	BLINK	BLINK	BLINK	ON

01489A

In most cars, the switching logic to generate these functions requires a number of multiple-throw contacts. As many as 18 conductors thread the steering column of some automobiles solely for turn-signal and emergency blinker functions. (The author discovered this recently to his astonishment and dismay when replacing the whole assembly because of one burned contact.)

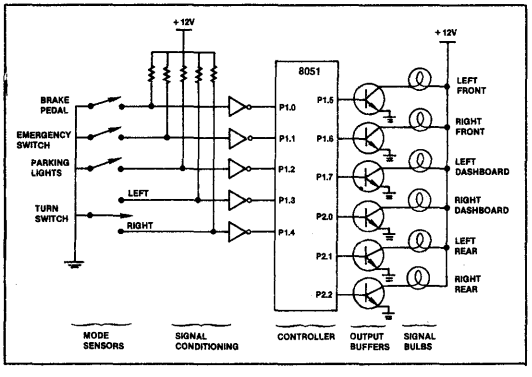
A multiple-conductor wiring harness runs to each corner of the car, behind the dash, up the steering column, and down to the blinker relay below. Connectors at each termination for each filament lead to extra cost and labor during construction, lower reliability and safety, and more costly repairs. And considering the system's present complexity, increasing its reliability or detecting failures would be quite difficult.

There are two reasons for going into such painful detail describing this example. First, to show that the messiest part of many system designs is determining what the controller should do. Writing the software to solve these functions will be comparatively easy. Secondly, to show the many potential failure points in the system. Later we'll see how the peripheral functions and intelligence built into a microcomputer (with a little creativity) can greatly reduce external interconnections and mechanical part count.

The Single-chip Solution

The circuit shown in Figure 16 indicates five input pins to the five input variables—left-turn select, right-turn select, brake pedal down, emergency switch on, and parking lights on. Six output pins turn on the front, rear, and dashboard indicators for each side. The microcomputer implements all logical functions through software, which periodically updates the output signals as time elapses and input conditions change.

Figure 16. Microcomputer Turn-signal Connections.



Design Example #3 demonstrated that symbolic addressing with user-defined bit names makes code and documentation easier to write and maintain. Accordingly, we'll assign these I/O pins names for use throughout the program. (The format of this example will differ somewhat from the others. Segments of the overall program will be presented in sequence as each is described.)

```
;
;          INPUT PIN DECLARATIONS:
;     (ALL INPUTS ARE POSITIVE-TRUE LOGIC)
;
BRAKE    BIT P1.0   ; BRAKE PEDAL DEPRESSED
EMERG    BIT P1.1   ; EMERGENCY BLINKER
                      ACTIVATED
PARK     BIT P1.2   ; PARKING LIGHTS ON
L_TURN   BIT P1.3   ; TURN LEVER DOWN
R_TURN   BIT P1.4   ; TURN LEVER UP
;
;          OUTPUT PIN DECLARATIONS:
;
L_FRNT   BIT P1.5   ; FRONT LEFT-TURN
                      INDICATOR
R_FRNT   BIT P1.6   ; FRONT RIGHT-TURN
                      INDICATOR
L_DASH   BIT P1.7   ; DASHBOARD LEFT-TURN
                      INDICATOR
R_DASH   BIT P2.0   ; DASHBOARD RIGHT-TURN
                      INDICATOR
L_REAR   BIT P2.1   ; REAR LEFT-TURN
                      INDICATOR
R_REAR   BIT P2.2   ; REAR RIGHT-TURN
                      INDICATOR
;
```

Another key advantage of symbolic addressing will appear further on in the design cycle. The locations of cable connectors, signal conditioning circuitry, voltage regulators, heat sinks, and the like all affect P.C. board layout. It's quite likely that the somewhat arbitrary pin assignment defined early in the software design cycle will prove to be less than optimum; rearranging the I/O pin assignment could well allow a more compact module, or eliminate costly jumpers on a single-sided board. (These considerations apply especially to automotive and other cost-sensitive applications needing single-chip controllers.) Since other architectures mask bytes or use "clever" algorithms to isolate bits by rotating them into the carry, re-routing an input signal (from bit 1 of port 1, for example, to bit 4 of port 3) could require extensive modifications throughout the software.

The Boolean Processor's direct bit addressing makes such changes absolutely trivial. The number of the port containing the pin is irrelevent, and masks and complex program structures are not needed. Only the initial Boolean varia-

```
;                ...          .....
; INTERRUPT RATE SUBDIVIDER
SUB_DIV          DATA    20H
; HIGH-FREQUENCY OSCILLATOR BIT
HI_FREQ          BIT     SUB_DIV.0
; LOW-FREQUENCY OSCILLATOR BIT
LO_FREQ          BIT     SUB_DIV.7
;                ...
                 ORG     0000H
JMP              INIT
;                ...          .....
                 ORG     100H
; PUT TIMER 0 IN MODE 1
INIT:            MOV     TMOD,#00000001B
; INITIALIZE TIMER REGISTERS
                 MOV     TL0,#0
                 MOV     TH0,#-16
; SUBDIVIDE INTERRUPT RATE BY 244
                 MOV     SUB_DIV,#244
; ENABLE TIMER INTERRUPTS
                 SETB    ET0
; GLOBALLY ENABLE ALL INTERRUPTS
                 SETB    EA
; START TIMER
                 SETB    TR0
;
;(CONTINUE WITH BACKGROUND PROGRAM)
;
; PUT TIMER 0 IN MODE 1
; INITIALIZE TIMER REGISTERS

; SUBDIVIDE INTERRUPT RATE BY 244
; ENABLE TIMER INTERRUPTS
; GLOBALLY ENABLE ALL INTERRUPTS
; START TIMER
```

ble declarations need to be changed; ASM51 automatically adjusts all addresses and symbolic references to the reassigned variables. The user is assured that no additional debugging or software verification will be required.

Timer 0 (one of the two on-chip timer/counters) replaces the thermo-mechanical blinker relay in the dashboard controller. During system initialization it is configured as a timer in mode 1 by setting the least significant bit of the timer mode register (TMOD). In this configuration the low-order byte (TL0) is incremented every machine cycle, overflowing and incrementing the high-order byte (TH0) every 256 μSec. Timer interrupt 0 is enabled so that a hardware interrupt will occur each time TH0 overflows. (For details of the numerous timer operating modes see the MCS-51™ User's Manual.)

An eight-bit variable in the bit-addressable RAM array will be needed to further subdivide the interrupts via software. The lowest-order bit of this counter toggles very fast to modulate the parking lights; bit 7 will be "tuned" to approximately 1 Hz for the turn- and emergency-indicator blinking rate.

Loading TH0 with -16 will cause an interrupt after 4.096 msec. The interrupt service routine reloads the high-order byte of timer 0 for the next interval, saves the CPU registers likely to be affected on the stack, and then decrements SUB_DIV. Loading SUB_DIV. with 244 initially and each time it decrements to zero will produce a 0.999 second period for the highest-order bit.

```
ORG   000BH     ; TIMER 0 SERVICE VECTOR
MOV   TH0,#-16
PUSH  PSW
PUSH  ACC
PUSH  B
DJNZ  SUB_DIV,T0SERV
MOV   SUB_DIV,#244
```

The code to sample inputs, perform calculations, and update outputs—the real "meat" of the signal controller algorithm—may be performed either as part of the interrupt service routine or as part of a background program loop. The only concern is that it must be executed at least several dozen times per second to prevent parking light flickering. We will assume the former case, and insert the code into the timer 0 service routine.

First, notice from the logic diagram (Figure 15) that the subterm (PARK · H_FREQ), asserted when the parking lights are to be on dimly, figures into four of the six output functions. Accordingly, we will first compute that term and save it in a temporary location named "DIM". The PSW contains two general purpose flags: F0, which corresponds to the 8048 flag of the same name, and PSW.1. Since The PSW has been saved and will be restored to its previous state after servicing the interrupt, we can use either bit for temporary storage.

```
DIM  BIT        PSW.1  ; DECLARE TEMP.
                       STORAGE FLAG
...  .....
MOV  C,PARK             ; GATE PARKING
                        LIGHT SWITCH
ANL  HI_FREQ            ; WITH HIGH
                        FREQUENCY
                        SIGNAL
MOV  DIM,C              ; AND SAVE IN
                        TEMP. VARIABLE.
```

This simple three-line section of code illustrates a remarkable point. The software indicates in very abstract terms exactly what function is being performed, independent of

01489A-22

the hardware configuration. The fact that these three bits include an input pin, a bit within a program variable, and a software flag in the PSW is totally invisible to the programmer.

Now generate and output the dashboard left turn signal.

```
;
   MOV  C,L_TURN        ; SET CARRY IF
                          TURN
   ORL  C,EMERG         ; OR EMERGENCY
                          SELECTED.
   ANL  C,LO_FREQ       ; GATE IN 1 HZ
                          SIGNAL
   MOV  L_DASH,C        ; AND OUTPUT TO
                          DASHBOARD.
```

To generate the left front turn signal we only need to add the parking light function in F0. But notice that the function in the carry will also be needed for the rear signal. We can save effort later by saving its current state in F0.

```
;
   MOV  F0,C            ; SAVE FUNCTION
                          SO FAR.
   ORL  C,DIM           ; ADD IN PARKING
                          LIGHT FUNCTION
   MOV  L_FRNT,C        ; AND OUTPUT TO
                          TURN SIGNAL.
```

Finally, the rear left turn signal should also be on when the brake pedal is depressed, provided a left turn is not in progress.

```
   MOV  C,BRAKE         ; GATE BRAKE
                          PEDAL SWITCH
   ANL  C,/L_TURN       ; WITH TURN
                          LEVER.
   ORL  C,F0            ; INCLUDE TEMP.
                          VARIABLE FROM
                          DASH
   ORL  C,DIM           ; AND PARKING
                          LIGHT FUNCTION
   MOV  L_REAR,C        ; AND OUTPUT TO
                          TURN SIGNAL.
```

Now we have to go through a similar sequence for the right-hand equivalents to all the left-turn lights. This also gives us a chance to see how the code segments above look when combined.

```
   MOV  C,R_TURN        ; SET CARRY IF
                          TURN
   ORL  C,EMERG         ; OR EMERGENCY
                          SELECTED.
   ANL  C,LO_FREQ       ; IF SO, GATE IN 1
                          HZ SIGNAL
   MOV  R_DASH,C        ; AND OUTPUT TO
                          DASHBOARD.
   MOV  F0,C            ; SAVE FUNCTION
                          SO FAR.
   ORL  C,DIM           ; ADD IN PARKING
                          LIGHT FUNCTION
   MOV  R_FRNT,C        ; AND OUTPUT TO
                          TURN SIGNAL.
   MOV  C,BRAKE         ; GATE BRAKE
                          PEDAL SWITCH
   ANL  C,/R_TURN       ; WITH TURN
                          LEVER.
   ORL  C,F0            ; INCLUDE TEMP.
                          VARIABLE FROM
                          DASH
   ORL  C,DIM           ; AND PARKING
                          LIGHT FUNCTION
   MOV  R_REAR,C        ; AND OUTPUT TO
                          TURN SIGNAL.
```

(The perceptive reader may notice that simply rearranging the steps could eliminate one instruction from each sequence.)

Now that all six bulbs are in the proper states, we can return from the interrupt routine, and the program is finished. This code essentially needs to reverse the status saving steps at the beginning of the interrupt.

```
   POP  B               ; RESTORE CPU
                          REGISTERS.
   POP  ACC
   POP  PSW
   RETI
```

Program Refinements. The luminescence of an incandescent light bulb filament is generally non-linear; the 50% duty cycle of HI_FREQ may not produce the desired intensity. If the application requires, duty cycles of 25%, 75%, etc. are easily achieved by ANDing and ORing in additional low-order bits of SUB_DIV. For example, 30 Hz signals of seven different duty cycles could be produced by considering bits 2—0 as shown in Table 7. The only software change required would be to the code which sets-up variable DIM:

```
   MOV  C,SUB_DIV.1     ; START WITH 50
                          PERCENT
   ANL  C,SUB_DIV.0     ; MASK DOWN TO 25
                          PERCENT
   ORL  C,SUB_DIV.2     ; AND BUILD BACK TO
                          62 PERCENT
   MOV  DIM,C           ; DUTY CYCLE FOR
                          PARKING LIGHTS.
```

01489A

Table 7. Non-trivial Duty Cycles.

SUB_DIV BITS								DUTY CYCLES						
7	6	5	4	3	2	1	0	12.5%	25.0%	37.5%	50.0%	62.5%	75.0%	87.5%
X	X	X	X	X	0	0	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF
X	X	X	X	X	0	0	1	OFF	OFF	OFF	OFF	OFF	OFF	ON
X	X	X	X	X	0	1	0	OFF	OFF	OFF	OFF	OFF	ON	ON
X	X	X	X	X	0	1	1	OFF	OFF	OFF	OFF	ON	ON	ON
X	X	X	X	X	1	0	0	OFF	OFF	OFF	ON	ON	ON	ON
X	X	X	X	X	1	0	1	OFF	OFF	ON	ON	ON	ON	ON
X	X	X	X	X	1	1	0	OFF	ON	ON	ON	ON	ON	ON
X	X	X	X	X	1	1	1	ON	ON	ON	ON	ON	ON	ON

Interconnections increase cost and decrease reliability. The simple buffered pin-per-function circuit in Figure 16 is insufficient when many outputs require higher-than-TTL drive levels. A lower-cost solution uses the 8051 serial port in the shift-register mode to augment I/O. In mode 0, writing a byte to the serial port data buffer (SBUF) causes the data to be output sequentially through the "RXD" pin while a burst of eight clock pulses is generated on the "TXD" pin. A shift register connected to these pins (Figure 17) will load the data byte as it is shifted out. A number of special peripheral driver circuits combining shift-register inputs with high drive level outputs have been introduced recently.

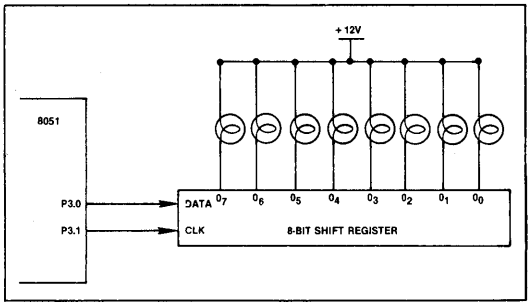


Figure 17. Output expansion using serial port.

Cascading multiple shift registers end-to-end will expand the number of outputs even further. The data rate in the I/O expansion mode is one megabaud, or 8 usec. per byte. This is the mode which the serial port defaults to following a reset, so no initialization is required.

The software for this technique uses the B register as a "map" corresponding to the different output functions. The program manipulates these bits instead of the output pins. After all functions have been calculated the B register is shifted by the serial port to the shift-register/driver. (While some outputs may glitch as data is shifted through them, at 1 Megabaud most people wouldn't notice. Some shift registers provide an "enable" bit to hold the output states while new data is being shifted in.)

This is where the earlier decision to address bits symbolically throughout the program is going to pay off. This major I/O restructuring is nearly as simple to implement as rearranging the input pins. Again, only the bit declarations need to be changed.

```
L_FRNT   BIT  B.0  ; FRONT LEFT-TURN
                    INDICATOR
R_FRNT   BIT  B.1  ; FRONT RIGHT-TURN
                    INDICATOR
L_DASH   BIT  B.2  ; DASHBOARD LEFT-TURN
                    INDICATOR
R_DASH   BIT  B.3  ; DASHBOARD RIGHT-TURN
                    INDICATOR
L_REAR   BIT  B.4  ; REAR LEFT-TURN
                    ; INDICATOR
R_REAR   BIT  B.5  ; REAR RIGHT-TURN
                    ; INDICATOR
```

The original program to compute the functions need not change. After computing the output variables, the control map is transmitted to the buffered shift register through the serial port:

```
MOV   SBUF,B  ; LOAD BUFFER AND TRANSMIT
```

The Boolean Processor solution holds a number of advantages over older methods. Fewer switches are required. Each is simpler, requiring fewer poles and lower current contacts. The flasher relay is eliminated entirely. Only six filaments are driven, rather than 10. The wiring harness is therefore simpler and less expensive—one conductor for each of the six lamps and each of the five sensor switches. The fewer conductors use far fewer connectors. The whole system is more reliable.

And since the system is much simpler it would be feasible to implement redundancy and/or fault detection on the four main turn indicators. Each could **still** be a standard double filament bulb, but with the filaments driven in parallel to tolerate single-element failures.

Even with redundancy, the lights will eventually fail. To handle this inescapable fact current or voltage sensing

01489A

circuits on each main drive wire can verify that each bulb and its high-current driver is functioning properly. Figure 18 shows one such circuit.

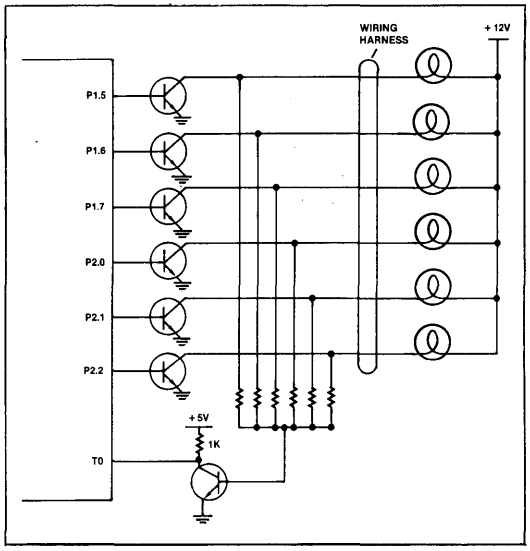


Figure 18.

Assume all of the lights are turned on except one; i.e., all but one of the collectors are grounded. For the bulb which is turned off, if there is continuity from +12 V through the bulb base and filament, the control wire, all connectors, and the P.C. board traces, and if the transistor is indeed not shorted to ground, then the collector will be pulled to +12 V. This turns on the base of Q8 through the corresponding resistor, and grounds the input pin, verifying that the bulb circuit is operational. The continuity of each circuit can be checked by software in this way.

Now turn *all* the bulbs on, grounding all the collectors. Q7 should be turned off, and the Test pin should be high. However, a control wire shorted to +12 V or an open-circuited drive transistor would leave one of the collectors at the higher voltage even now. This too would turn on Q7, indicating a different type of failure. Software could perform these checks once per second by executing the routine every time the software counter SUB_DIV is reloaded by the interrupt routine.

```
    DJNZ SUB_DIV,T0SERV
    MOV  SUB_DIV,#244       ; RELOAD COUNTER
    ORL  P1,#11100000B      ; SET CONTROL
                              OUTPUTS HIGH
    ORL  P2,#00000111B
    CLR  L_FRNT             ; FLOAT DRIVE
                              COLLECTOR
    JB   T0,FAULT           ; T0 SHOULD BE
                              PULLED LOW
    SETB L_FRNT             ; PULL COLLECTOR
                              BACK DOWN
    CLR  L_DASH
    JB   T0,FAULT
    SETB L_DASH
    CLR  L_REAR
    JB   T0,FAULT
    SETB L_REAR
    CLR  R_FRNT
    JB   T0,FAULT
    SETB R_FRNT
    CLR  R_DASH
    JB   T0,FAULT
    SETB R_DASH
    CLR  R_REAR
    JB   T0,FAULT
    SETB R_REAR
;
; WITH ALL COLLECTORS GROUNDED, T0
  SHOULD BE HIGH
; IF SO, CONTINUE WITH INTERRUPT ROUTINE.
    JB   T0,T0SERV
FAULT:                  ; ELECTRICAL FAILURE
                        ;PROCESSING ROUTINE
                        ; (LEFT TO READER'S
                        ; IMAGINATION)
T0SERV:                 ; CONTINUE WITH
                        ;INTERRUPT PROCESSING
;
;
```

The complete assembled program listing is printed in Appendix A. The resulting code consists of 67 program statements, not counting declarations and comments, which assemble into 150 bytes of object code. Each pass through the service routine requires (coincidently) 67 usec, plus 32 usec once per second for the electrical test. If executed every 4 msec as suggested this software would typically reduce the throughput of the background program by less than 2%.

Once a microcomputer has been designed into a system, new features suddenly become virtually free. Software could make the emergency blinkers flash alternately or at a rate faster than the turn signals. Turn signals could override the emergency blinkers. Adding more bulbs would allow multiple tail light sequencing and syncopation — true flash factor, so to speak.

Design Example #5 - Complex Control Functions

Finally, we'll mix byte and bit operations to extend the use of 8051 into extremely complex applications.

Programmers can arbitrarily assign I/O pins to input and output functions only if the total does not exceed 32, which is insufficient for applications with a very large number of input variables. One way to expand the number of inputs is with a technique similar to multiplexed-keyboard scanning.

01489A

Figure 19 shows a block diagram for a moderately complex programmable industrial controller with the following characteristics:

- 64 input variable sensors;
- 12 output signals;
- Combinational and sequential logic computations;
- Remote operation with communications to a host processor via a high-speed full-duplex serial link;
- Two prioritized external interrupts;
- Internal real-time and time-of-day clocks.

While many microprocessors could be programmed to provide these capabilities with assorted peripheral support chips, an 8051 microcomputer needs **no** other integrated circuits!

The 64 input sensors are logically arranged as an 8x8 matrix. The pins of Port 1 sequentially enable each column of the sensor matrix; as each is enabled Port 0 reads in the state of each sensor in that column. An eight-byte block in bit-addressable RAM remembers the data as it is read in so that after each complete scan cycle there is an internal map of the current state of all sensors. Logic functions can then directly address the elements of the bit map.

The computer's serial port is configured as a nine-bit UART, transferring data at 17,000 bytes-per-second. The ninth bit may distinguish between address and data bytes.

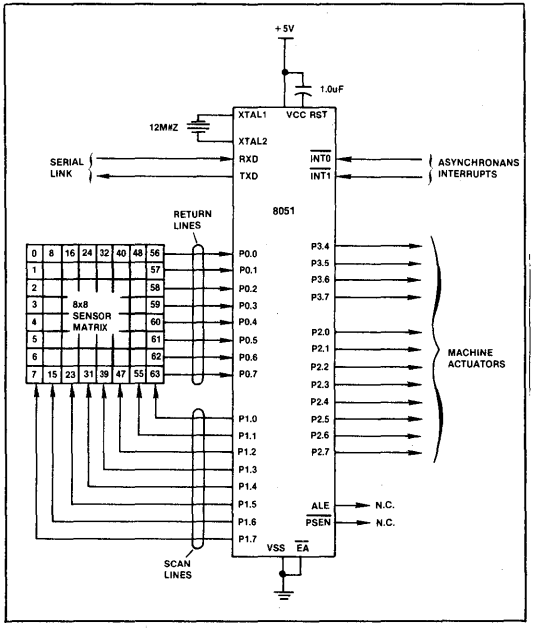


Figure 19. Block diagram of 64-input machine controller.

The 8051 serial port can be configured to detect bytes with the address bit set, automatically ignoring all others. Pins INT0 and INT1 are interrupts configured respectively as high-priority, falling-edge triggered and low-priority, low-level triggered. The remaining 12 I/O pins output TTL-level control signals to 12 actuators.

There are several ways to implement the sensor matrix circuitry, all logically similar. Figure 20.a shows one possibility. Each of the 64 sensors consists of a pair of simple switch contacts in series with a diode to permit multiple contact closures throughout the matrix.

The scan lines from Port 1 provide eight un-encoded active-high scan signals for enabling columns of the matrix. The return lines on rows where a contact is closed are pulled high and read as logic ones. Open return lines are pulled to ground by one of the 40 kohm resistors and are read as zeroes. (The resistor values must be chosen to ensure all return lines are pulled above the 2.0 V logic threshold, even in the worst-case, where all contacts in an enabled column are closed.) Since P0 is provided open-collector outputs and high-impedance MOS inputs its input loading may be considered negligible.

The circuits in Figures 20.b—20.d are variations on this theme. When input signals must be electrically isolated from the computer circuitry as in noisy industrial environments, phototransistors can replace the switch/diode pairs **and** provide optical isolation as in Figure 20.b. Additional opto-isolators could also be used on the control output and special signal lines.

The other circuits assume that input signals are already at TTL levels. Figure 20.c uses octal three-state buffers enabled by active-low scan signals to gate eight signals onto Port 0. Port 0 is available for memory expansion or peripheral chip interfacing between sensor matrix scans. Eight-to-one multiplexers in Figure 20.d select one of eight inputs for each return line as determined by encoded address bits output on three pins of Port 1. (Five more output pins are thus freed for more control functions.) Each output can drive at least one standard TTL or up to 10 low-power TTL loads without additional buffering.

Going back to the original matrix circuit, Figure 21 shows the method used to scan the sensor matrix. Two complete bit maps are maintained in the bit-addressable region of the RAM: one for the current state and one for the previous state read for each sensor. If the need arises, the program could then sense input transitions and/or debounce contact closures by comparing each bit with its earlier value.

01469A

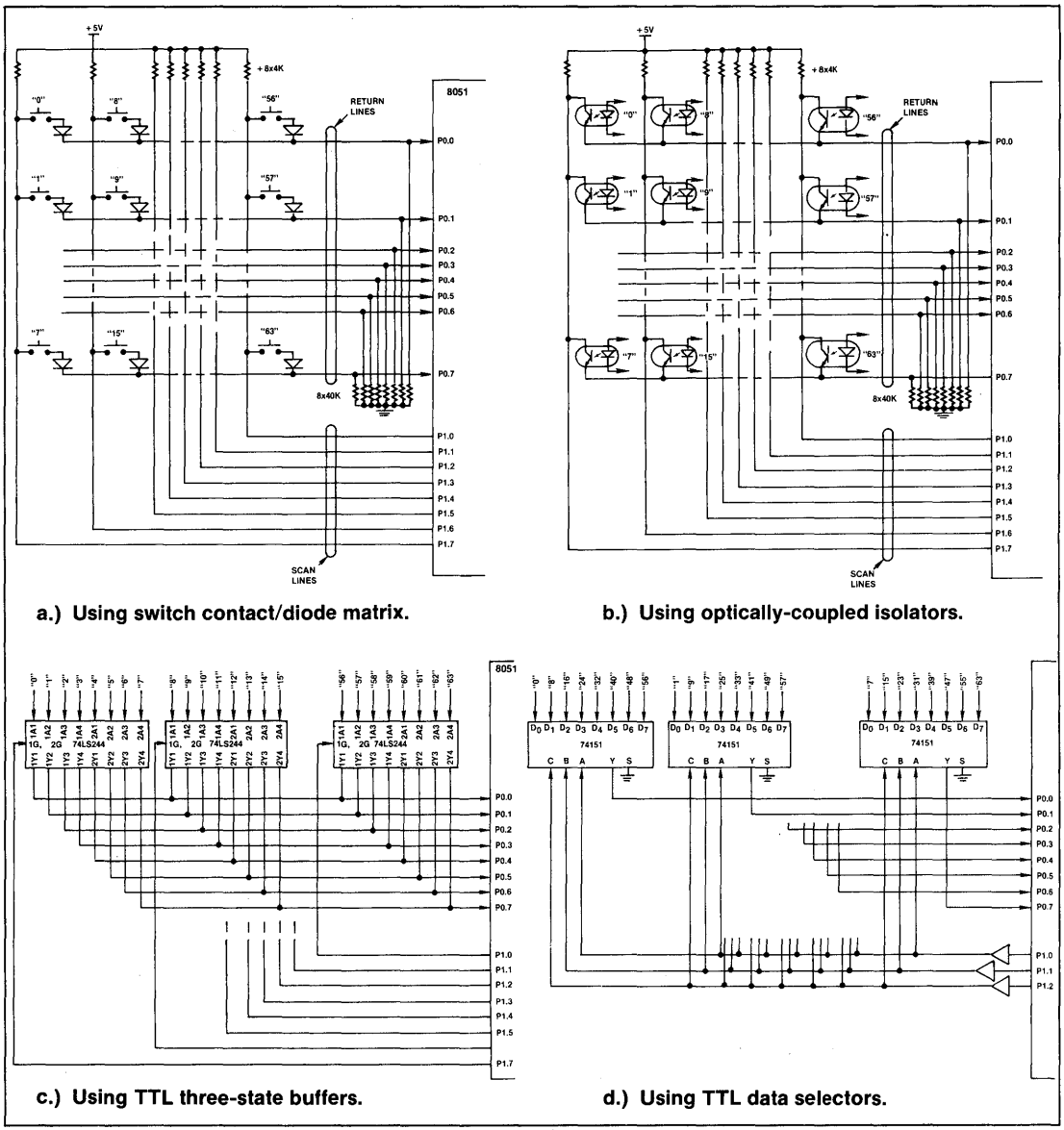


Figure 20. Sensor Matrix Implementation Methods.

The code in Example 3 implements the scanning algorithm for the circuits in Figure 20.a. Each column is enabled by setting a single bit in a field of zeroes. The bit maps are positive logic; ones represent contacts that are closed or isolators turned on.

Example 3.

```
INPUT_SCAN:     ; SUBROUTINE TO READ
                  CURRENT STATE
                ; OF 64 SENSORS AND
                  SAVE IN RAM 20H-27H.
    MOV  R0,#20H    ; INITIALIZE
                    ; POINTERS
    MOV  R1,#28H    ; FOR BIT MAP
                    ; BASES.
```

01489A

```
        MOV  A,#80H        ; SET FIRST BIT IN
                             ACC.
SCAN:   MOV  P1.A          ; OUTPUT TO SCAN
                             LINES.
        RR   A             ; SHIFT TO ENABLE
                             NEXT COLUMN
                             NEXT.
        MOV  R2,A          ; REMEMBER CUR-
                             RENT SCAN
                             POSITION.
        MOV  A,P0          ; READ RETURN
                             LINES.
        XCH  A,@R0         ; SWITCH WITH
                             PREVIOUS MAP
                             BITS.
        MOV  @R1,A         ; SAVE PREVIOUS
                             STATE AS WELL.
        INC  R0            ; BUMP POINTERS.
        INC  R1
        MOV  A,R2          ; RELOAD SCAN LINE
                             MASK
        JNB  ACC.7,SCAN    ; LOOP UNTIL ALL
                             EIGHT COLUMNS
                             READ.
        RET
```

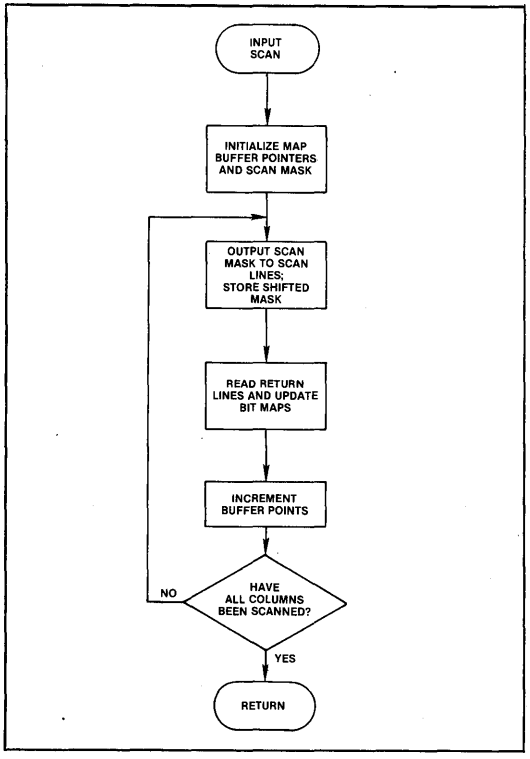


Figure 21. Flowchart for reading in sensor matrix.

What happens after the sensors have been scanned depends on the individual application. Rather than inventing some artificial design problem, software corresponding to commonplace logic elements will be discussed.

Combinatorial Output Variables. An output variable which is a simple (or not so simple) combinational function of several input variables is computed in the spirit of Design Example 3. All 64 inputs are represented in the bit maps; in fact, the sensor numbers in Figure 20 correspond to the absolute bit addresses in RAM! The code in Example 4 activates an actuator connected to P2.2 when sensors 12, 23, and 34 are closed and sensors 45 and 56 are open.

Example 4.

Simple Combinatorial Output Variables.

```
; SET P2.2 = (12) (23) (34) (/45) (/56)
        MOV  C,12
        ANL  C,23
        ANL  C,34
        ANL  C,/45
        ANL  C,/56
        MOV  P2.2,C
```

Intermediate Variables. The examination of a typical relay-logic ladder diagram will show that many of the rungs control *not* outputs but rather relays whose contacts figure into the computation of other functions. In effect, these relays indicate the state of intermediate variables of a computation.

The MCS-51™ solution can use any directly addressable bit for the storage of such intermediate variables. Even when all 128 bits of the RAM array are dedicated (to input bit maps in this example), the accumulator, PSW, and B register provide 18 additional flags for intermediate variables.

For example, suppose switches 0 through 3 control a safety interlock system. Closing any of them should deactivate certain outputs. Figure 22 is a ladder diagram for this situation. The interlock function could be recomputed for every output affected, or it may be computed once and saved (as implied by the diagram). As the program proceeds this bit can qualify each output.

Example 5. Incorporating Override signal into actuator outputs.

```
        CALL INPUT_SCAN
        MOV  C,0
        ORL  C,1
        ORL  C,2
        ORL  C,3
        MOV  F0,C
;       ....    .....
```

01489A-28

```
;       COMPUTE FUNCTION 0
;
        ANL   C,/F0
        MOV   P1.0,C
;       ....    .....
;       COMPUTE FUNCTION 1
;
        ANL   C,/F0
        MOV   P1.1,C
;       ....    .....
;       COMPUTE FUNCTION 2
;
        ANL   C,/F0
        MOV   P1.2,C
;       ....    .....
```

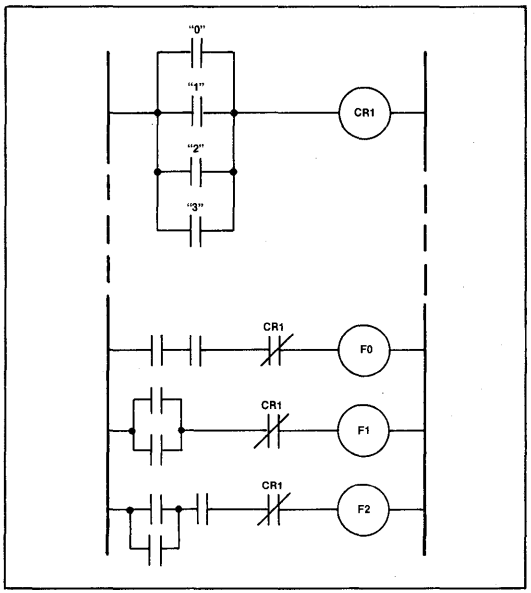


Figure 22. Ladder diagram for output override circuitry.

Latching Relays. A latching relay can be forced into either the ON or OFF state by two corresponding input signals, where it will remain until forced onto the opposite state—analogous to a TTL Set/Reset flip-flop. The relay is used as an intermediate variable for other calculations. In the previous example, the emergency condition could be remembered and remain active until an "emergency cleared" button is pressed.

Any flag or addressable bit may represent a latching relay with a few lines of code (see Example 6).

Example 6. Simulating a latching relay.

```
;L_SET   SET FLAG 0 IF C=1
L_SET:    ORL   C,F0
          MOV   F0,C
;

;L_RSET RESET FLAG 0 IF C=1
L_RSET:  CPS   C
          ANL   C,F0
          MOV   F0,C
;
```

Time Delay Relays. A time delay relay does not respond to an input signal until it has been present (or absent) for some predefined time. For example, a ballast or load resistor may be switched in series with a D.C. motor when it is first turned on, and shunted from the circuit after one second. This sort of time delay may be simulated by an interrupt routine driven by one of the two 8051 timer/counters. The procedure followed by the routine depends heavily on the details of the exact function needed; time-outs or time delays with resettable or non-resettable inputs are possible. If the interrupt routine is executed every 10 milliseconds the code in Example 7 will clear an intermediate variable set by the background program after it has been active for two seconds.

Example 7. Code to clear USRFLG after a fixed time delay.

```
          JNB   USR_FLG,NXTTST
          DJNZ DLAY_COUNT,NXTTST
          CLR   USR_FLG
          MOV   DLAY_COUNT,#200
NXTTST:  ;..     .....
```

Serial Interface to Remote Processor. When it detects emergency conditions represented by certain input combinations (such as the earlier Emergency Override), the controller could shut down the machine immediately and/or alert the host processor via the serial port. Code bytes indicating the nature of the problem could be transmitted to a central computer. In fact, at 17,000 bytes-per-second, the entire contents of both bit maps could be sent to the host processor for further analysis in less than a millisecond! If the host decides that conditions warrant, it could alert other remote processors in the system that a problem exists and specify which shut-down sequence each should initiate. For more information on using the serial port, consult the MCS-51™ User's Manual.

Response Timing.
One difference between relay and programmed industrial controllers (when each is considered as a "black box") is their respective reaction times to input changes. As reflected by a ladder diagram, relay systems contain a

01489A

large number of "rungs" operating in parallel. A change in input conditions will begin propagating through the system immediately, possibly affecting the output state within milliseconds.

Software, on the other hand, operates sequentially. A change in input states will not be detected until the next time an input scan is performed, and will not affect the outputs until that section of the program is reached. For that reason the raw speed of computing the logical functions is of extreme importance.

Here the Boolean processor pays off. *Every instruction mentioned in this Note* completes in one or two microseconds—the *minimum* instruction execution time for many other microcontrollers! A ladder diagram containing a hundred rungs, with an average of four contacts per rung can be replaced by approximately five hundred lines of software. A complete pass through the entire matrix scanning routine and all computations would require about a millisecond; less than the time it takes for most relays to change state.

A programmed controller which simulates each Boolean function with a subroutine would be less efficient by at least an order of magnitude. Extra software is needed for the simulation routines, and each step takes longer to execute for three reasons: several byte-wide logical instructions are executed per user program step (rather than one Boolean operation); most of those instructions take longer to execute with microprocessors performing multiple off-chip accesses; and calling and returning from the various subroutines requires overhead for stack operations.

In fact, the speed of the Boolean Processor solution is likely to be much faster than the system requires. The CPU might use the time left over to compute feedback parameters, collect and analyze execution statistics, perform system diagnostics, and so forth.

Additional functions and uses.

With the building-block basics mentioned above many more operations may be synthesized by short instruction sequences.

Exclusive-OR. There are no common mechanical devices or relays analogous to the Exclusive-OR operation, so this instruction was omitted from the Boolean Processor. However, the Exclusive-OR or Exclusive-NOR operation may be performed in two instructions by conditionally complementing the carry or a Boolean variable based on the state of any other testable bit.

```
; EXCLUSIVE-OR FUNCTION IMPOSED ON CARRY
; USING F0 IS INPUT VARIABLE.
XOR_F0: JNB   F0,XORCNT  ;("JB" FOR X-NOR)
        CPL   C
XORCNT: ...   .....
```

XCH. The contents of the carry and some other bit may be exchanged (switched) by using the accumulator as temporary storage. Bits can be moved into and out of the accumulator simultaneously using the Rotate-through-carry instructions, though this would alter the accumulator data.

```
; EXCHANGE CARRY WITH USRFLG
XCHBIT: RLC   A
        MOV   C,USR_FLG
        RRC   A
        MOV   USR_FLG,C
        RLC   A
```

Extended Bit Addressing. The 8051 can directly address 144 general-purpose bits for all instructions in Figure 3.b. Similar operations may be extended to any bit anywhere on the chip with some loss of efficiency.

The logical operations AND, OR, and Exclusive-OR are performed on byte variables using six different addressing modes, one of which lets the source be an immediate mask, and the destination any directly addressable byte. Any bit may thus be set, cleared, or complemented with a three-byte, two-cycle instruction if the mask has all bits but one set or cleared.

Byte variables, registers, and indirectly addressed RAM may be moved to a bit addressable register (usually the accumulator) in one instruction. Once transferred, the bits may be tested with a conditional jump, allowing any bit to be polled in 3 microseconds—still much faster than most architectures—or used for logical calculations. (This technique can also simulate additional bit addressing modes with byte operations.)

Parity of bytes or bits. The parity of the current accumulator contents is always available in the PSW, from whence it may be moved to the carry and further processed. Error-correcting Hamming codes and similar applications require computing parity on groups of isolated bits. This can be done by conditionally complementing the carry flag based on those bits or by gathering the bits into the accumulator (as shown in the DES example) and then testing the parallel parity flag.

Multiple byte shift and CRC codes.

Though the 8051 serial port can accommodate eight- or nine-bit data transmissions, some protocols involve much

01489A-30

longer bit streams. The algorithms presented in Design Example 2 can be extended quite readily to 16 or more bits by using multi-byte input and output buffers.

Many mass data storage peripherals and serial communications protocols include Cyclic Redundancy (CRC) codes to verify data integrity. The function is generally computed serially by hardware using shift registers and Exclusive-OR gates, but it can be done with software. As each bit is received into the carry, appropriate bits in the multi-byte data buffer are conditionally complemented based on the incoming data bit. When finished, the CRC register contents may be checked for zero by ORing the two bytes in the accumulator.

4. SUMMARY

A truly unique facet of the Intel MCS-51™ microcomputer family design is the collection of features optimized for the one-bit operations so often desired in real-world, real-time control applications. Included are 17 special instructions, a Boolean accumulator, implicit and direct addressing modes, program and mass data storage, and many I/O options. These are the world's first single-chip microcomputers able to efficiently manipulate, operate on, and transfer either bytes or individual bits as data.

This Application Note has detailed the information needed by a microcomputer system designer to make full use of these capabilities. Five design examples were used to contrast the solutions allowed by the 8051 and those required by previous architectures. Depending on the individual application, the 8051 solution will be easier to design, more reliable to implement, debug, and verify, use less program memory, and run up to an order of magnitude faster than the same function implemented on previous digital computer architectures.

Combining byte- and bit-handling capabilities in a single microcomputer has a strong synergistic effect: the power of the result exceeds the power of byte- and bit-processors laboring individually. Virtually all user applications will benefit in some ways from this duality. Data intensive applications will use bit addressing for test pin monitoring or program control flags; control applications will use byte manipulation for parallel I/O expansion or arithmetic calculations.

It is hoped that these design examples give the reader an appreciation of these unique features and suggest ways to exploit them in his or her own application.

01489A

```
ISIS-II MCS-51 MACRO ASSEMBLER V1.0
OBJECT MODULE PLACED IN :F0:AP70.HEX
ASSEMBLER INVOKED BY:   :f1:asm51 ap70.src date(328)
LOC  OBJ             LINE     SOURCE

                        1     $XREF TITLE(AP-70 APPENDIX)
                        2     ;*******************************************************
                        3     ;
                        4     ;       THE FOLLOWING PROGRAM USES THE BOOLEAN INSTRUCTION SET
                        5     ;       OF THE INTEL 8051 MICROCOMPUTER TO PERFORM A NUMBER OF
                        6     ;       AUTOMOTIVE DASHBOARD CONTROL FUNCTIONS RELATING TO
                        7     ;       TURN SIGNAL CONTROL, EMERGENCY BLINKERS, BRAKE LIGHT
                        8     ;       CONTROL, AND PARKING LIGHT OPERATION.
                        9     ;       THE ALGORITHMS AND HARDWARE ARE DESCRIBED IN DESIGN
                       10     ;       EXAMPLE #4 OF INTEL APPLICATION NOTE AP-70,
                       11     ;               "USING THE INTEL MCS-51(TM)
                       12     ;               BOOLEAN PROCESSING CAPABILITIES"
                       13     ;
                       14     ;*******************************************************
                       15     ;
                       16     ;       INPUT PIN DECLARATIONS:
                       17     ;        (ALL INPUTS ARE POSITIVE-TRUE LOGIC.
                       18     ;       INPUTS ARE HIGH WHEN RESPECTIVE SWITCH CONTACT IS CLOSED.)
                       19     ;
0090                   20     BRAKE   BIT     P1.0    ; BRAKE PEDAL DEPRESSED
0091                   21     EMERG   BIT     P1.1    ; EMERGENCY BLINKER ACTIVATED
0092                   22     PARK    BIT     P1.2    ; PARKING LIGHTS ON
0093                   23     L_TURN  BIT     P1.3    ; TURN LEVER DOWN
0094                   24     R_TURN  BIT     P1.4    ; TURN LEVER UP
                       25     ;
                       26     ;       OUTPUT PIN DECLARATIONS:
                       27     ;       (ALL OUTPUTS ARE POSITIVE TRUE LOGIC.
                       28     ;       BULB IS TURNED ON WHEN OUTPUT PIN IS HIGH.)
                       29     ;
0095                   30     L_FRNT  BIT     P1.5    ; FRONT LEFT-TURN INDICATOR
0096                   31     R_FRNT  BIT     P1.6    ; FRONT RIGHT-TURN INDICATOR
0097                   32     L_DASH  BIT     P1.7    ; DASHBOARD LEFT-TURN INDICATOR
00A0                   33     R_DASH  BIT     P2.0    ; DASHBOARD RIGHT-TURN INDICATOR
00A1                   34     L_REAR  BIT     P2.1    ; REAR LEFT-TURN INDICATOR
00A2                   35     R_REAR  BIT     P2.2    ; REAR RIGHT-TURN INDICATOR
                       36     ;
00A3                   37     S_FAIL  BIT     P2.3    ; ELECTRICAL SYSTEM FAULT INDICATOR
                       38     ;
                       39     ;       INTERNAL VARIABLE DEFINITIONS:
                       40     ;
0020                   41     SUB_DIV DATA    20H             ; INTERRUPT RATE SUBDIVIDER
0000                   42     HI_FREQ BIT     SUB_DIV.0       ; HIGH-FREQUENCY OSCILLATOR BIT
0007                   43     LO_FREQ BIT     SUB_DIV.7       ; LOW-FREQUENCY OSCILLATOR BIT
                       44     ;       ...     .....
00D1                   45     DIM     BIT     PSW.1           ; PARKING LIGHTS ON FLAG
                       46     ;
                       47     ;=======================================================
                       48 +1  $EJECT
```

```
LOC  OBJ              LINE     SOURCE

                        49             ORG     0000H           ; RESET VECTOR
0000 020040             50             LJMP    INIT
                        51     ;
000B                    52             ORG     000BH           ; TIMER 0 SERVICE VECTOR
000B 758CF0             53             MOV     TH0,#-16        ; HIGH TIMER BYTE ADJUSTED TO CONTROL INT. RATE
000E C0D0               54             PUSH    PSW             ; EXECUTE CODE TO SAVE ANY REGISTERS USED BELOW
0010 0154               55             AJMP    UPDATE          ;      (CONTINUE WITH REST OF ROUTINE)
                        56     ;
0040                    57             ORG     0040H
0040 758A00             58     INIT:   MOV     TL0,#0          ; ZERO LOADED INTO LOW-ORDER BYTE AND
0043 758CF0             59             MOV     TH0,#-16        ; -16 IN HIGH-ORDER BYTE GIVES 4 MSEC PERIOD
0046 758961             60             MOV     TMOD,#01100001B ; 8-BIT AUTO RELOAD COUNTER MODE FOR TIMER 1,
                        61                                     ;  16-BIT TIMER MODE FOR TIMER 0 SELECTED
0049 7520F4             62             MOV     SUB_DIV,#244    ; SUBDIVIDE INTERRUPT RATE BY 244 FOR 1 HZ
004C D2A9               63             SETB    ET0             ; USE TIMER 0 OVERFLOWS TO INTERRUPT PROGRAM
004E D2AF               64             SETB    EA              ; CONFIGURE IE TO GLOBALLY ENABLE INTERRUPTS
0050 D28C               65             SETB    TR0             ; KEEP INSTRUCTION CYCLE COUNT UNTIL OVERFLOW
0052 80FE               66             SJMP    $               ; START BACKGROUND PROGRAM EXECUTION
                        67     ;
                        68     ;
0054 D52038             69     UPDATE: DJNZ    SUB_DIV,TOSERV  ; EXECUTE SYSTEM TEST ONLY ONCE PER SECOND
0057 7520F4             70             MOV     SUB_DIV,#244    ; GET VALUE FOR NEXT ONE SECOND DELAY AND
                        71     ;                               ; GO THROUGH ELECTRICAL SYSTEM TEST CODE:
005A 4390E0             72             ORL     P1,#11100000B   ; SET CONTROL OUTPUTS HIGH
005D 43A007             73             ORL     P2,#00000111B
0060 C295               74             CLR     L_FRNT          ; FLOAT DRIVE COLLECTOR
0062 20B428             75             JB      T0,FAULT        ; T0 SHOULD BE PULLED LOW
0065 D295               76             SETB    L_FRNT          ; PULL COLLECTOR BACK DOWN
0067 C297               77             CLR     L_DASH          ; REPEAT SEQUENCE FOR L_DASH,
0069 20B421             78             JB      T0,FAULT
006C D297               79             SETB    L_DASH
006E C2A1               80             CLR     L_REAR          ;                L_REAR,
0070 20B41A             81             JB      T0,FAULT
0073 D2A1               82             SETB    L_REAR
0075 C296               83             CLR     R_FRNT          ;                R_FRNT,
0077 20B413             84             JB      T0,FAULT
007A D296               85             SETB    R_FRNT
007C C2A0               86             CLR     R_DASH          ;                R_DASH,
007E 20B40C             87             JB      T0,FAULT
0081 D2A0               88             SETB    R_DASH
0083 C2A2               89             CLR     R_REAR          ;                AND R_REAR.
0085 20B405             90             JB      T0,FAULT
0088 D2A2               91             SETB    R_REAR
                        92     ;
                        93     ;       WITH ALL COLLECTORS GROUNDED, T0 SHOULD BE HIGH
                        94     ;       IF SO, CONTINUE WITH INTERRUPT ROUTINE.
                        95     ;
008A 20B402             96             JB      T0,TOSERV
008D B2A3               97     FAULT:  CPL     S_FAIL          ; ELECTRICAL FAILURE PROCESSING ROUTINE
                        98                                     ; (TOGGLE INDICATOR ONCE PER SECOND)
                        99 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     100      ;       CONTINUE WITH INTERRUPT PROCESSING:
                     101      ;
                     102      ;   1)  COMPUTE LOW BULB INTENSITY WHEN PARKING LIGHTS ARE ON.
                     103      ;
008F A201            104      TOSERV: MOV     C,SUB_DIV.1         ; START WITH 50 PERCENT,
0091 8200            105              ANL     C,SUB_DIV.0         ; MASK DOWN TO 25 PERCENT,
0093 7202            106              ORL     C,SUB_DIV.2         ; BUILD BACK TO 62.5 PERCENT,
0095 8292            107              ANL     C,PARK              ; GATE WITH PARKING LIGHT SWITCH,
0097 92D1            108              MOV     DIM,C               ; AND SAVE IN TEMP. VARIABLE.
                     109      ;
                     110      ;   2)  COMPUTE AND OUTPUT LEFT-HAND DASHBOARD INDICATOR.
                     111      ;
0099 A293            112              MOV     C,L_TURN            ; SET CARRY IF TURN
009B 7291            113              ORL     C,EMERG             ; OR EMERGENCY SELECTED.
009D 8207            114              ANL     C,LO_FREQ           ; IF SO, GATE IN 1 HZ SIGNAL
009F 9297            115              MOV     L_DASH,C            ; AND OUTPUT TO DASHBOARD.
                     116      ;
                     117      ;   3)  COMPUTE AND OUTPUT LEFT-HAND FRONT TURN SIGNAL.
                     118      ;
00A1 92D5            119              MOV     F0,C                ; SAVE FUNCTION SO FAR.
00A3 72D1            120              ORL     C,DIM               ; ADD IN PARKING LIGHT FUNCTION
00A5 9295            121              MOV     L_FRNT,C            ; AND OUTPUT TO TURN SIGNAL.
                     122      ;
                     123      ;   4)  COMPUTE AND OUTPUT LEFT-HAND REAR TURN SIGNAL.
                     124      ;
00A7 A290            125              MOV     C,BRAKE             ; GATE BRAKE PEDAL SWITCH
00A9 B093            126              ANL     C,/L_TURN           ; WITH TURN LEVER.
00AB 72D5            127              ORL     C,F0                ; INCLUDE TEMP. VARIABLE FROM DASH
00AD 72D1            128              ORL     C,DIM               ; AND PARKING LIGHT FUNCTION
00AF 92A1            129              MOV     L_REAR,C            ; AND OUTPUT TO TURN SIGNAL.
                     130      ;
                     131      ;   5)  REPEAT ALL OF ABOVE FOR RIGHT-HAND COUNTERPARTS.
                     132      ;
00B1 A294            133              MOV     C,R_TURN            ; SET CARRY IF TURN
00B3 7291            134              ORL     C,EMERG             ; OR EMERGENCY SELECTED.
00B5 8207            135              ANL     C,LO_FREQ           ; IF SO, GATE IN 1 HZ SIGNAL
00B7 92A0            136              MOV     R_DASH,C            ; AND OUTPUT TO DASHBOARD.
00B9 92D5            137              MOV     F0,C                ; SAVE FUNCTION SO FAR.
00BB 72D1            138              ORL     C,DIM               ; ADD IN PARKING LIGHT FUNCTION
00BD 9296            139              MOV     R_FRNT,C            ; AND OUTPUT TO TURN SIGNAL.
00BF A290            140              MOV     C,BRAKE             ; GATE BRAKE PEDAL SWITCH
00C1 B094            141              ANL     C,/R_TURN           ; WITH TURN LEVER.
00C3 72D5            142              ORL     C,F0                ; INCLUDE TEMP. VARIABLE FROM DASH
00C5 72D1            143              ORL     C,DIM               ; AND PARKING LIGHT FUNCTION
00C7 92A2            144              MOV     R_REAR,C            ; AND OUTPUT TO TURN SIGNAL.
                     145      ;
                     146      ;       RESTORE STATUS REGISTER AND RETURN.
                     147      ;
00C9 D0D0            148              POP     PSW                 ; RESTORE PSW
00CB 32              149              RETI                        ; AND RETURN FROM INTERRUPT ROUTINE
                     150      ;
                     151              END
```

```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------

NAME      TYPE      VALUE AND REFERENCES

BRAKE .   N BSEG    0090H  20# 125 140
DIM . .   N BSEG    00D1H  45# 108 120 128 138 143
EA. . .   N BSEG    00AFH  64
EMERG .   N BSEG    0091H  21# 113 134
ETO . .   N BSEG    00A9H  63
FO. . .   N BSEG    00D5H  119 127 137 142
FAULT .   L CSEG    008DH  75 78 81 84 87 90 97#
HI_FREQ   N BSEG    0000H  42#
INIT. .   L CSEG    0040H  50 58#
L_DASH.   N BSEG    0097H  32# 77 79 115
L_FRNT.   N BSEG    0095H  30# 74 76 121
L_REAR.   N BSEG    00A1H  34# 80 82 129
L_TURN.   N BSEG    0093H  23# 112 126
LO_FREQ   N BSEG    0007H  43# 114 135
P1. . .   N DSEG    0090H  20 21 22 23 24 30 31 32 72
P2. . .   N DSEG    00A0H  33 34 35 37 73
PARK. .   N BSEG    0092H  22# 107
PSW . .   N DSEG    00D0H  45 54 148
R_DASH.   N BSEG    00A0H  33# 86 88 136
R_FRNT.   N BSEG    0096H  31# 83 85 139
R_REAR.   N BSEG    00A2H  35# 89 91 144
R_TURN.   N BSEG    0094H  24# 133 141
S_FAIL.   N BSEG    00A3H  37# 97
SUB_DIV   N DSEG    0020H  41# 42 43 62 69 70 104 105 106
TO. . .   N BSEG    00B4H  75 78 81 84 87 90 96
TOSERV.   L CSEG    008FH  69 96 104#
THO . .   N DSEG    00BCH  53 59
TLO . .   N DSEG    00BAH  58
TMOD. .   N DSEG    0089H  60
TRO . .   N BSEG    00BCH  65
UPDATE.   L CSEG    0054H  55 69#

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

U.S. AND CANADIAN SALES OFFICES

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080
TWX: 910-338-0026
TELEX: 34-6372

ALABAMA

Intel Corp.
303 Williams Avenue, S.W.
Suite 1422
Huntsville 35801
Tel: (205) 533-9353

Pen-Tech Associates, Inc.
Holiday Office Center
3322 Memorial Pkwy., S.W.
Huntsville 35801
Tel: (205) 881-9298

ARIZONA

Intel Corp.
10210 N. 25th Avenue, Suite 11
Phoenix 85021
Tel: (602) 997-9695

BFA
4426 North Saddle Bag Trail
Scottsdale 85251
Tel: (602) 994-5400

CALIFORNIA

Intel Corp.
7670 Opportunity Rd.
Suite 135
San Diego 92111
Tel: (714) 268-3563

Intel Corp.*
2000 East 4th Street
Suite 100
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114

Intel Corp.*
15335 Morrison
Suite 345
Sherman Oaks 91403
Tel: (213) 986-9510
TWX: 910-495-2045

Intel Corp.*
3375 Scott Blvd.
Santa Clara 95051
Tel: (408) 987-8086
TWX: 910-339-9279
910-338-0255

Earle Associates, Inc.
4617 Ruffner Street
Suite 202
San Diego 92111
Tel: (714) 278-5441

Mac-l
2576 Shattuck Ave.
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-l
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Mac-l
558 Valley Way
Calaveras Business Park
Milpitas 95035
Tel: (408) 946-8885

Mac-l
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Mac-l
1321 Centinela Avenue
Suite 1
Santa Monica 90404
Tel: (213) 829-4797

Mac-l
20121 Ventura Blvd., Suite 240E
Woodland Hills 91364
Tel: (213) 347-5900

COLORADO

Intel Corp.*
650 S. Cherry Street
Suite 720
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289

Westek Data Products, Inc.
25921 Fern Gulch
P.O. Box 1355
Evergreen 80439
Tel: (303) 674-5255

Westek Data Products, Inc.
1322 Arapahoe
Boulder 80302
Tel: (303) 449-2620

Westek Data Products, Inc.
1228 W. Hinsdale Dr.
Littleton 80120
Tel: (303) 797-0482

CONNECTICUT

Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA

Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
5151 Adanson Street, Suite 203
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

Pen-Tech Associates, Inc.
201 S.E. 15th Terrace, Suite K
Deerfield Beach 33441
Tel: (305) 421-4989

Pen-Tech Associates, Inc.
111 So. Maitland Ave., Suite 202
P.O. Box 1475
Maitland 32751
Tel: (305) 645-3444

GEORGIA

Pen Tech Associates, Inc.
Cherokee Center, Suite 21
627 Cherokee Street
Marietta 30060
Tel: (404) 424-1931

ILLINOIS

Intel Corp.*
2550 Golf Road, Suite 815
Rolling Meadows 60008
Tel: (312) 981-7200
TWX: 910-651-5881

Technical Representatives
1502 North Lee Street
Bloomington 61701
Tel: (309) 829-8080

INDIANA

Intel Corp.
9101 Wesleyan Road
Suite 204
Indianapolis 46268
Tel: (317) 299-0623

IOWA

Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52405
Tel: (319) 393-5510

KANSAS

Intel Corp.
9393 W. 110th St., Ste. 265
Overland Park 66210
Tel: (913) 642-8080

Technical Representatives, Inc.
8245 Nieman Road, Suite 100
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

Technical Representatives, Inc.
360 N. Rock Road
Suite 4
Wichita 67206
Tel: (316) 681-0242

MARYLAND

Intel Corp.*
7257 Parkway Drive
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

Mesa Inc.
11900 Parklawn Drive
Rockville 20852
Tel: Washington (301) 881-8430
Baltimore (301) 792-0021

MASSACHUSETTS

Intel Corp.*
27 Industrial Ave.
Chelmsford 01824
Tel: (617) 667-8126
TWX: 710-343-6333

EMC Corp.
381 Elliot Street
Newton 02164
Tel: (617) 244-4740
TWX: 922531

MICHIGAN

Intel Corp.*
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 810-244-4915

Lowry & Associates, Inc.
135 W. North Street
Suite 4
Brighton 48116
Tel: (313) 227-7067

Lowry & Associates, Inc.
3902 Costa NE
Grand Rapids 49505
Tel: (616) 363-9839

MINNESOTA

Intel Corp.
7401 Metro Blvd.
Suite 355
Edina 55435
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI

Intel Corp.
502 Earth City Plaza
Suite 121
Earth City 63045
Tel: (314) 291-1990

Technical Representatives, Inc.
320 Brookes Drive, Suite 104
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY

Intel Corp.*
Raritan Plaza
2nd Floor
Raritan Center
Edison 08817
Tel: (201) 225-3000
TWX: 710-480-6238

NEW MEXICO

BFA Corporation
P.O. Box 1237
Las Cruces 88001
Tel: (505) 523-0601
TWX: 910-983-0543

BFA Corporation
3705 Westerfield, N.E.
Albuquerque 87111
Tel: (505) 292-1212
TWX: 910-989-1157

NEW YORK

Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
80 Washington St.
Poughkeepsie 12601
Tel: (914) 473-2303
TWX: 510-248-0060

Intel Corp.*
2255 Lyell Avenue
Lower Floor East Suite
Rochester 14606
Tel: (716) 254-6120
TWX: 510-253-7391

Measurement Technology, Inc.
159 Northern Boulevard
Great Neck 11021
Tel: (516) 482-3500

T-Squared
4054 Newcourt Avenue
Syracuse 13206
Tel: (315) 463-8592
TWX: 710-541-0554

T-Squared
2 E. Main
Victor 14564
Tel: (716) 924-9101
TWX: 510-254-8542

NORTH CAROLINA

Intel Corp.
154 Huffman Mill Rd.
Burlington 27215
Tel: (919) 584-3631

Pen-Tech Associates, Inc.
1202 Eastchester Dr.
Highpoint 27260
Tel: (919) 883-9125

OHIO

Intel Corp.*
6500 Poe Avenue
Dayton 45415
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
Chagrin-Brainard Bldg., No. 210
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736
TWX: 810-427-9298

OREGON

Intel Corp.
10700 S.W. Beaverton
Hillsdale Highway
Suite 324
Beaverton 97005
Tel: (503) 641-8086
TWX: 910-467-8741

PENNSYLVANIA

Intel Corp.*
275 Commerce Dr.
200 Office Center
Suite 300
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

Q.E.D. Electronics
300 N. York Road
Hatboro 19040
Tel: (215) 674-9600

TEXAS

Intel Corp.*
2925 L.B.J. Freeway
Suite 175
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5617

Intel Corp.*
6420 Richmond Ave.
Suite 280
Houston 77057
Tel: (713) 784-3400
TWX: 910-881-2490

Industrial Digital Systems Corp.
5925 Sovereign
Suite 101
Houston 77036
Tel: (713) 988-9421

Intel Corp.
313 E. Anderson Lane
Suite 314
Austin 78752
Tel: (512) 454-3628

WASHINGTON

Intel Corp.
Suite 114, Bldg. 3
1603 116th Ave. N.E.
Bellevue 98005
Tel: (206) 453-8086
TWX: 910-443-3002

WISCONSIN

Intel Corp.
150 S. Sunnyslope Rd.
Brookfield 53005
Tel: (414) 784-9060

CANADA

Intel Semiconductor Corp.*
Suite 233, Bell Mews
39 Highway 7, Bells Corners
Ottawa, Ontario K2H 8R2
Tel: (613) 829-9714
TELEX: 053-4115

Intel Semiconductor Corp.
50 Galaxy Blvd.
Unit 12
Rexdale, Ontario
M9W 4Y5
Tel: (416) 675-2105
TELEX: 06983574

Multilek, Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 226-2365
TELEX: 053-4585

Multilek, Inc.
Toronto
Tel: (416) 245-4622

Multilek, Inc.
Montreal
Tel: (514) 481-1350

*Field Application Location

U.S. AND CANADIAN DISTRIBUTORS

ALABAMA

†Hamilton/Avnet Electronics
4812 Commercial Drive N.W.
Huntsville 35805
Tel: (205) 837-7210

†Pioneer/Huntsville
1207 Putman Drive NW
Huntsville 35805
Tel: (205) 837-9033
TWX: 810-726-2197

ARIZONA

†Hamilton/Avnet Electronics
505 S. Madison Drive
Tempe 85281
Tel: (602) 275-7851

†Wyle Distribution Group
8155 N. 24th Avenue
Phoenix 85021
Tel: (602) 249-2232
TWX: 910-951-4282

CALIFORNIA

Arrow Electronics, Inc.
Electronics Distribution Division
9511 Ridge Haven Court
San Diego 92123
Tel: (714) 565-4800

Arrow Electronics, Inc.
Electronics Distribution Division
720 Palomar Avenue
Sunnyvale, California 94086
Tel: (408) 739-3011

†Avnet Electronics
350 McCormick Avenue
Costa Mesa 92626
Tel: (714) 754-6111
TWX: 910-595-1928

Hamilton/Avnet Electronics
1175 Bordeaux Dr.
Sunnyvale 94086
Tel: (408) 743-3355
TWX: 910-339-9332

†Hamilton/Avnet Electronics
4545 Viewridge Ave.
San Diego 92123
Tel: (714) 571-7510
TWX: 910-335-1216

†Hamilton/Avnet Electronics
10950 W. Washington Blvd.
Culver City 90230
Tel: (213) 558-2809
TWX: 910-340-6364 or 7073

†Hamilton Electro Sales
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4100

†Avnet Electronics
4942 Rosecrans Ave.
Hawthorne 90250
Tel: (213) 970-0956

†Wyle Distribution Group
124 Maryland Street
El Segundo 90245
Tel: (213) 322-3826
TWX: 910-348-7140 or 7111

†Wyle Distribution Group
9525 Chesapeake Dr.
San Diego 92123
Tel: (714) 565-9171
TWX: 910-335-1590

†Wyle Distribution Group
3000 Bowers Avenue
Santa Clara 95052
Tel: (408) 727-2500
TWX: 910-338-0451 or 0296

Hamilton/Avnet Electronics
3170 Pullman
Costa Mesa 92626
Tel: (714) 641-1850

COLORADO

Arrow Electronics, Inc.
Electronics Distribution Division
2121 South Hudson Street
Denver 80222
Tel: (303) 758-2100

†Wyle Distribution Group
6777 E. 50th Avenue
Commerce City 80022
Tel: (303) 287-9611
TWX: 910-931-0510

†Hamilton/Avnet Electronics
8765 E. Orchard Road
Suite 708
Englewood 80111
Tel: (303) 740-1000
TWX: 910-931-0510

CONNECTICUT

†Arrow Electronics
12 Beaumont Road
Wallingford 06492
Tel: (203) 265-7741
TWX: 710-476-0162

†Hamilton/Avnet Electronics
643 Danbury Road
Georgetown 06829
Tel: (203) 762-0361

†Harvey Electronics
112 Main Street
Norwalk 06851
Tel: (203) 853-1515
TWX: 710-468-3373

FLORIDA

†Arrow Electronics
1001 N.W. 62nd Street
Suite 108
Ft. Lauderdale 33309
Tel: (305) 776-7790

†Arrow Electronics
115 Palm Bay Road, NW
Suite 10, Bldg. 200
Palm Bay 32905
Tel: (305) 725-1480
TWX: 510-959-6337

†Hamilton/Avnet Electronics
6800 Northwest 20th Ave.
Ft. Lauderdale 33309
Tel: (305) 971-2900
TWX: 510-955-3097

†Pioneer/Orlando
6220 S. Orange Blossom Trail
Suite 412
Orlando 32809
Tel: (305) 859-3600
TWX: 810-850-0177

Hamilton/Avnet Electronics
3197 Tech. Drive North
St. Petersburg 33702
Tel: (813) 576-3930
TWX: 810-863-0374

GEORGIA

Arrow Electronics
2979 Pacific Drive
Norcross 30071
Tel: (404) 449-8252
TWX: 810-757-4213

†Hamilton/Avnet Electronics
6700 I-85 Access Road, No. 11
Suite 1E
Norcross 30071
Tel: (404) 448-0800

ILLINOIS

Arrow Electronics
492 Lunt Avenue
P.O. Box 94248
Schaumburg 60193
Tel: (312) 893-9420
TWX: 910-222-1807

†Hamilton/Avnet Electronics
3901 No. 25th Avenue
Schiller Park 60176
Tel: (312) 678-6310
TWX: 910-227-0060

Pioneer/Chicago
1551 Carmen Drive
Elk Grove 60007
Tel: (312) 437-9680
TWX: 910-222-1834

INDIANA

†Pioneer/Indiana
6408 Castleplace Drive
Indianapolis 46250
Tel: (317) 849-7300
TWX: 810-260-1794

KANSAS

†Hamilton/Avnet Electronics
9219 Quivira Road
Overland Park 66215
Tel: (913) 888-8900

†Component Specialties, Inc.
8369 Nieman Road
Lenexa 66214
Tel: (913) 492-3555

MARYLAND

Arrow Electronics, Inc.
Electronics Distribution Division
4801 Benson Avenue
Baltimore 21227
Tel: (301) 247-5200
(202) 737-1700

†Hamilton/Avnet Electronics
7235 Standard Drive
Hanover 21076
Tel: (301) 796-5684
TWX: 710-862-1861

†Pioneer/Washington
9100 Gaither Road
Gaithersburg 20760
Tel: (301) 948-0710
TWX: 710-828-0545

MASSACHUSETTS

†Hamilton/Avnet Electronics
50 Tower Office Park
Woburn 01801
Tel: (617) 273-7500

†Arrow Electronics
96D Commerce Way
Woburn 01801
Tel: (617) 933-8130

Harvey/Boston
44 Hartwell Ave.
Lexington 02173
Tel: (617) 861-9200

MICHIGAN

†Arrow Electronics
3810 Varsity Drive
Ann Arbor 48104
Tel: (313) 971-8220
TWX: 810-223-6020

†Pioneer/Michigan
13485 Stamford
Livonia 48150
Tel: (313) 525-1800
TWX: 810-242-3271

†Hamilton/Avnet Electronics
32487 Schoolcraft Road
Livonia 48150
Tel: (313) 522-4700
TWX: 810-242-8775

MINNESOTA

†Industrial Components
5229 Edina Industrial Blvd.
Minneapolis 55435
Tel: (612) 831-2666
TWX: 910-576-3153

†Arrow Electronics
5230 W. 73rd Street
Edina 55435
Tel: (612) 830-1800
TWX: 910-576-2726

†Hamilton/Avnet Electronics
7449 Cahill Road
Edina 55435
Tel: (612) 941-3801
TWX: 910-576-2720

MISSOURI

†Hamilton/Avnet Electronics
396 Brookes Lane
Hazelwood 63042
Tel: (314) 731-1144
TWX: 910-762-0606

NEW HAMPSHIRE

†Arrow Electronics
1 Perimeter Road
Manchester 03103
Tel: (603) 668-6968

NEW JERSEY

†Arrow Electronics
Pleasant Valley Avenue
Moorestown 08057
Tel: (609) 235-1900
TWX: 710-897-0829

†Arrow Electronics
285 Midland Avenue
Saddle Brook 07662
Tel: (201) 797-5800
TWX: 710-988-2206

Hamilton/Avnet Electronics
10 Industrial Road
Fairfield 07006
Tel: (201) 575-3390
TWX: 710-734-4338

†Harvey Electronics
45 Route 46
Pinebrook 07058
Tel: (201) 575-3510
TWX: 710-734-4382

†Hamilton/Avnet Electronics
1 Keystone Ave.
Bldg. 36
Cherry Hill 08034
Tel: (609) 424-0100
TWX: 710-897-1405

NEW MEXICO

†Alliance Electronics Inc.
11721 Central Ave. N.E.
Albuquerque 87123
Tel: (505) 292-3360
TWX: 910-989-1151

Arrow Electronics, Inc.
Electronics Distribution Division
2460 Alamo Avenue, Southeast
Albuquerque 87106
Tel: (505) 243-4566

†Hamilton/Avnet Electronics
2524 Baylor Drive, S.E.
Albuquerque 87106
Tel: (505) 765-1500

†Microcomputer System Technical Demonstrator Centers

U.S. AND CANADIAN DISTRIBUTORS

NEW YORK

Harvey Electronics
P.O. Box 1208
Binghampton 13902
Tel: (607) 748-8211
TWX: 510-252-0893

Arrow Electronics
900 Broad Hollow Road
Farmingdale 11735
Tel: (516) 694-6800
TWX: 510-224-6494

†Arrow Electronics
3000 South Winton Road
Rochester 14623
Tel: (716) 275-0300
TWX: 910-338-0026

†Hamilton/Avnet Electronics
167 Clay Road
Rochester 14623
Tel: (716) 475-9130
TWX: 910-340-6364

†Arrow Electronics
7705 Maltlage Drive
Liverpool 13088
Tel: (315) 652-1000
TWX: 710-545-0230

Arrow Electronics
20 Oser Avenue
Hauppauge 11787
Tel: (516) 231-1000
TWX: 510-224-6494

†Hamilton/Avnet Electronics
16 Corporate Circle
E. Syracuse 13057
Tel: (315) 437-2641

†Hamilton/Avnet Electronics
5 Hub Drive
Melville, Long Island 11746
Tel: (516) 454-6000
TWX: 510-252-0893

†Harvey Electronics
60 Crossways Park West
Woodbury 11797
Tel: (516) 921-8700
TWX: 510-221-2184

Harvey/Rochester
840 Airport Park
Fairport 14450
Tel: (716) 381-7070

NORTH CAROLINA

Pioneer/Carolina
106 Industrial Ave.
Greensboro 27406
Tel: (919) 273-4441
TWX: 510-925-1114

†Hamilton/Avnet Electronics
2803 Industrial Drive
Raleigh 27609
Tel: (919) 829-8030

Arrow Electronics
938 Burke Street
Winston-Salem 27102
Tel: (919) 725-8711
TWX: 510-922-4765

OHIO

Arrow Electronics
7620 McEwen Road
Centerville 45459
Tel: (513) 435-5563
TWX: 810-459-1611

Arrow Electronics
6238 Cochran Rd.
Solon 44139
Tel: (216) 248-3990

†Hamilton/Avnet Electronics
954 Senate Drive
Dayton 45459
Tel: (513) 433-0610
TWX: 910-340-2531

†Pioneer/Dayton
1900 Troy Street
Dayton 45404
Tel: (513) 236-9900
TWX: 810-459-1622

Arrow Electronics
10 Knollcrest Dr.
Cincinnati 45237
Tel: (513) 761-5432
TWX: 810-461-2670

†Pioneer/Cleveland
4800 E. 131st Street
Cleveland 44105
Tel: (216) 587-3600
TWX: 810-422-2210

†Hamilton/Avnet Electronics
4588 Emery Industrial Parkway
Warrensville Heights 44128
Tel: (216) 831-3500

OKLAHOMA

†Components Specialties, Inc.
7920 E. 40th Street
Tulsa 74145
Tel: (918) 664-2820
TWX: 910-845-2215

OREGON

†Hamilton/Avnet Electronics
6024 SW Jean Rd.
Bldg. C, Suite 10
Lake Oswego 97034
Tel: (503) 635-8157

†Almac/Stroum Electronics
8022 S.W. Nimbus, Bldg. 7
Beaverton 97005
Tel: (503) 641-9070

PENNSYLVANIA

Pioneer/Pittsburgh
560 Alpha Drive
Pittsburgh 15238
Tel: (412) 782-2300
TWX: 710-795-3122

Pioneer/Delaware Valley
141 Gibraltar Road
Horsham 19044
Tel: (215) 674-4000
TWX: 510-665-6778

†Arrow/ Electronics
4297 Greensburg Pike
Suite 3114
Pittsburgh 15221
Tel: (412) 351-4000

TENNESSEE

†Arrow Electronics
P.O. Box 129
W. Andrew Johnson Hwy.
Talbott 37677
Tel: (615) 587-2137

TEXAS

Arrow Electronics
13715 Gamma Road
Dallas 75234
Tel: (214) 386-7500
TWX: 910-861-5495

Arrow Electronics, Inc.
Electronics Distribution Division
10700 Corporate Drive, Suite 100
Stafford 77477
Tel: (713) 491-4100

Component Specialties Inc.
8222 Jamestown Drive
Suite 115
Austin 78757
Tel: (512) 837-8922
TWX: 910-874-1320

Hamilton/Avnet Electronics
10508A Boyer Blvd.
Austin 78757
Tel: (512) 837-8911

†Hamilton/Avnet Electronics
4445 Sigma Road
Dallas 75240
Tel: (214) 661-8661
TWX: 910-860-5371

†Hamilton/Avnet Electronics
3939 Ann Arbor Drive
Houston 77063
Tel: (713) 780-1771

†Component Specialties, Inc.
10907 Shady Trail, Suite 101
Dallas 75220
Tel: (214) 357-6511
TWX: 910-861-4999

†Component Specialties, Inc.
8585 Commerce Park Drive, Suite 590
Houston 77036
Tel: (713) 771-7237
TWX: 910-881-2422

UTAH

†Hamilton/Avnet Electronics
1585 West 2110 South
Salt Lake City 84119
Tel: (801) 972-2800

WASHINGTON

Arrow Electronics, Inc.
Electronics Distribution Division
1059 Andover Park East
Tukwila 98188
Tel: (206) 575-0907

†Hamilton/Avnet Electronics
14212 N.E. 21st Street
Bellevue 98005
Tel: (206) 746-8750

†Almac/Stroum Electronics
5811 Sixth Ave. South
Seattle 98108
Tel: (206) 763-2300
TWX: 910-444-2067

†Wyle Distribution Group
1750 132nd Avenue NE
Bellevue 98005
Tel: (206) 453-8300
TWX: 910-443-2526

WISCONSIN

†Arrow Electronics
430 W. Rawson Avenue
Oak Creek 53154
Tel: (414) 764-6600
TWX: 910-338-0026

†Hamilton/Avnet Electronics
2975 Moorland Road
New Berlin 53151
Tel: (414) 784-4510
TWX: 910-262-1182

CANADA

ALBERTA

Zentronics
9224 27th Avenue
Edmonton T6N 1B2

†L.A. Varah Ltd.
4742 14th Street N.E.
Calgary T2E 6L7
Tel: (403) 230-1235
TWX: 018-258-97

Zentronics
3651 21st N.E.
Calgary T2E 6T5
Tel: (403) 230-1422

BRITISH COLUMBIA

†L.A. Varah Ltd.
2077 Alberta Street
Vancouver V5Y 1C4
Tel: (604) 873-3211
TWX: 610-929-1068

Zentronics
550 Cambie St.
Vancouver V6B 2N7
Tel: (604) 688-2533
TWX: 04-5077-89

MANITOBA

L.A. Varah
1-1832 King Edward Street
Winnipeg R2R 0N1
Tel: (204) 633-6190
TWX: 07-55-365

Zentronics
590 Berry St.
Winnipeg R3H 0S1
Tel: (204) 775-8661

ONTARIO

†L.A. Varah, Ltd.
505 Kenora Avenue
Hamilton L8E 3P2
Tel: (416) 561-9311
TWX: 061-8349

†Hamilton/Avnet Electronics
3688 Nashua Drive, Units G & H
Mississauga L4V 1M5
Tel: (416) 677-7432
TWX: 610-492-8860

†Hamilton/Avnet Electronics
1735 Courtwood Crescent
Ottawa K2C 3J2
Tel: (613) 226-1700

†Zentronics
141 Catherine Street
Ottawa K2P 1C3
Tel: (613) 238-6411
TWX: 053-3636

†Zentronics
1355 Meyerside Drive
Mississauga, Ontario L5T 1C9
Tel: (416) 676-9000
Telex: 06-983-657

QUEBEC

†Hamilton/Avnet Electronics
2670 Sabourin Street
St. Laurent H4S 1M2
Tel: (514) 331-6443
TWX: 610-421-3731

Zentronics
5010 Pare Street
Montreal H4P 1P3
Tel: (514) 735-5361
TWX: 05-827-535

†Microcomputer System Technical Demonstrator Centers

INTERNATIONAL SALES AND MARKETING OFFICES

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA

Micro Sistemas S.A.
9 De Julio 561
Cordoba
Tel: 54-51-32-880
TELEX: 51837 BICCO

AUSTRALIA

A.J.F. Systems & Components Pty. Ltd.
310 Queen Street
Melbourne
Victoria 3000
Tel:
TELEX:

Warburton Franki
Corporate Headquarters
372 Eastern Valley Way
Chatswood, New South Wales 2067
Tel: 407-3261
TELEX: AA 21299

AUSTRIA

Bacher Elektronische Geraete GmbH
Rotenmulgasse 26
A 1120 Vienna
Tel: (0222) 83 63 96
TELEX: (01) 1532

Rekirsch Elektronik Geraete GmbH
Lichtensteinstrasse 97
A1000 Vienna
Tel: (222) 347646
TELEX: 74759

BELGIUM

Inelco Belgium S.A.
Ave. des Croix de Guerre 94
B1120 Brussels
Tel: (02) 216 01 60
TELEX: 25441

BRAZIL

Icotron S.A.
0511-Av. Mutinga 3650
6 Andar
Pirituba-Sao Paulo
Tel: 261-0211
TELEX: (011) 222 ICO BR

CHILE

DIN
Av. Vic. Mc kenna 204
Casilla 6055
Santiago
Tel: 227 564
TELEX: 3520003

CHINA

C.M. Technologies
525 University Avenue
Suite A-40
Palo Alto, CA 94301

COLOMBIA

International Computer Machines
Adpo. Aereo 19403
Bogota 1
Tel: 232-6635
TELEX: 43439

CYPRUS

Cyprus Eltrom Electronics
P.O. Box 5393
Nicosia
Tel: 21-27982

DENMARK

STL-Lyngso Komponent A/S
Ostmarken 4
DK-2860 Soborg
Tel: (01) 67 00 77
TELEX: 22990

Scandinavian Semiconductor
Supply A/S
Nannasgade 18
DK-2200 Copenhagen
Tel: (01) 83 50 90
TELEX: 19037

FINLAND

Oy Fintronic AB
Melkonkatu 24 A
SF-00210
Helsinki 21
Tel: 0-692 6022
TELEX: 124 224 Ftron SF

FRANCE

Celdis S.A.*
53, Rue Charles Frerot
F-94250 Gentilly
Tel: (1) 581 00 20
TELEX: 200 485

Feutrier
Rue des Trois Glorieuses
F-42270 St. Priest-en-Jarez
Tel: (77) 74 67 33
TELEX: 300 0 21

Metrologie*
La Tour d'Asnieres
4, Avenue Laurent Cely
92606-Asnieres
Tel: 791 44 44
TELEX: 611 448

Tekelec Airtronic*
Cite des Bruyeres
Rue Carle Vernet
F-92310 Sevres
Tel: (1) 534 75 35
TELEX: 204552

GERMANY

Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
D-8000 Munich 80
Tel: (089) 434061
TELEX: 522561

Jermyn GmbH
Postfach 1180
D-6077 Camberg
Tel: (06434) 231
TELEX: 484426

Kontron Elektronik GmbH
Breslauerstrasse 2
8057 Eching B
D-8000 Munich
Tel: (89) 319.011
TELEX: 522122

Neye Enatechnik GmbH
Schillerstrasse 14
D-2085 Quickborn-Hamburg
Tel: (04106) 6121
TELEX: 02-13590

GREECE

American Technical Enterprises
P.O. Box 156
Athens
Tel: 30-1-8811271
30-1-8219470

HONG KONG

Schmidt & Co.
28/F Wing on Center
Connaught Road
Hong Kong
Tel: 5-455-644
TELEX: 74766 Schmc Hx

INDIA

Micronic Devices
104/109C, Nirmal Industrial Estate
Sion (E)
Bombay 400022, India
Tel: 486-170
TELEX: 011-5947 MDEV IN

ISRAEL

Eastronics Ltd.*
11 Rozanis Street
P.O. Box 39300
Tel Aviv 61390
Tel: 475151
TELEX: 33638

ITALY

Eledra 3S S.P.A.*
Viale Elvezia, 18
I 20154 Milan
Tel: (02) 34.93.041-31.85.441
TELEX: 332332

JAPAN

Asahi Electronics Co. Ltd.
KMM Bldg. Room 407
2-14-1 Asano, Kokura
Kita-Ku, Kitokyushu City 802
Tel: (093) 511-6471
TELEX: AECKY 7126-16

Hamilton-Avnet Electronics Japan Ltd.
YU and YOU Bldg. 1-4 Horidome-Cho
Nihonbashi
Tel: (03) 662-9911
TELEX: 2523774

Nippon Micro Computer Co. Ltd.
Mutsumi Bldg. 4-5-21 Kojimachi
Chlyoda-ku, Tokyo 102
Tel: (03) 230-0041

Ryoyo Electric Corp.
Konwa Bldg.
1-12-22, Tsukiji, 1-Chome
Chuo-Ku, Tokyo 104
Tel: (03) 543-7711

Tokyo Electron Ltd.
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel: (045) 471-8811
TELEX: 781-4473

KOREA

Koram Digital
Room 411 Ahil Bldg.
49-4 2-GA Hoehyun-Dong
Chung-Ku Seoul
Tel: 23-8123
TELEX: K23542 HANSINT

Leewood International, Inc.
C.P.O. Box 4046
112-25, Sokong-Dong
Chung-Ku, Seoul 100
Tel: 28-5927
CABLE: "LEEWOOD" Seoul

NETHERLANDS

Inelco Nether. Comp. Sys. BV
Turfstekerstraat 63
Aalsmeer 1431 D
Tel: (2977) 28855
TELEX: 14693

Koning & Hartman
Koperwerf 30
2544 EN Den Haag
Tel: (70) 210.101
TELEX: 31528

NEW ZEALAND

W. K. McLean Ltd.
P.O. Box 18-065
Glenn Innes, Auckland, 6
Tel: 587-037
TELEX: NZ2763 KOSFY

NORWAY

Nordisk Elektronik (Norge) A/S
Postoffice Box 122
Smedsvingen 4
1364 Hvalstad
Tel: 02 78 62 10
TELEX: 17546

PORTUGAL

Ditram
Componentes E Electronica LDA
Av. Miguel Bombarda, 133
Lisboa 1
Tel: (19) 545313
TELEX: 14347 GESPIC

SINGAPORE

General Engineers Associates
Blk 3, 1003-1008, 10th Floor
P.S.A. Multi-Storey Complex
Telok Blangah/Pasir Panjang
Singapore 5
Tel: 271-3163
TELEX: RS23987 GENERCO

SOUTH AFRICA

Electronic Building Elements
Pine Square
18th Street
Hazelwood, Pretoria 0001
Tel: 789 221
TELEX: 30181SA

SPAIN

Interface
Av. Generalisimo 51 9°
E-Madrid 16
Tel: 456 3151

ITT SESA
Miguel Angel 16
Madrid 10
Tel: (1) 4190957
TELEX: 27707/27461

SWEDEN

AB Gosta Backstrom
Box 12009
10221 Stockholm
Tel: (08) 541 080
TELEX: 10135

Nordisk Electronik AB
Box 27301
S-10254 Stockholm
Tel: (08) 635040
TELEX: 10547

SWITZERLAND

Industrade AG
Gemsenstrasse 2
Postcheck 80 - 21190
CH-8021 Zurich
Tel: (01) 60 22 30
TELEX: 56788

TAIWAN

Taiwan Automation Co.*
3d Floor #75, Section 4
Nanking East Road
Taipei
Tel: 771-0940
TELEX: 11942 TAIAUTO

TURKEY

Turkelek Electronics
Apapurk Boulevard 169
Ankara
Tel: 189483

UNITED KINGDOM

Comway Microsystems Ltd.
Market Street
68-Bracknell, Berkshire
Tel: (344) 51654
TELEX: 847201

G.E.C. Semiconductors Ltd.
East Lane
North Wembley
Middlesex HA9 7PP
Tel: (01) 904-9303/908-4111
TELEX: 28817

Jermyn Industries
Vestry Estate
Sevenoaks, Kent
Tel: (0732) 501.44
TELEX: 95142

Rapid Recall, Ltd.
6 Soho Mills Ind. Park
Wooburn Green
Bucks, England
Tel: (6285) 24961
TELEX: 849439

Sintrom Electronics Ltd.*
Arkwright Road 2
Reading, Berkshire RG2 0LS
Tel: (0734) 85464
TELEX: 847395

VENEZUELA

Componentes y Circuitos
Electronicos TTLCA C.A.
Apartado 3223
Caracas 101
Tel: 718-100
TELEX: 21795 TELETIPOS

*Field Application Location

INTERNATIONAL SALES AND MARKETING OFFICES

INTEL® MARKETING OFFICES

AUSTRALIA

Intel Australia
Suite 2, Level 15, North Point
100 Miller Street
North Sydney, NSW, 2060
Tel: 450-847
TELEX: AA 20097

BELGIUM

Intel Corporation S.A.
Rue du Moulin a Papier 51
Boite 1
B-1160 Brussels
Tel: (02) 660 30 10
TELEX: 24814

DENMARK

Intel Denmark A/S*
Lyngbyvej 32 2nd Floor
DK-2100 Copenhagen East
Tel: (01) 18 20 00
TELEX: 19567

FINLAND

Intel Scandinavia
Sentnerikuja 3
SF - 00400 Helsinki 40
Tel: (0) 558531
TELEX: 123 332

FRANCE

Intel Corporation, S.A.R.L.*
5 Place de la Balance
Silic 223
94528 Rungis Cedex
Tel: (01) 687 22 21
TELEX: 270475

GERMANY

Intel Semiconductor GmbH*
Seidlstrasse 27
8000 Muenchen 2
Tel: (089) 53 891
TELEX: 523 177

Intel Semiconductor GmbH
Mainzer Strasse 75
6200 Wiesbaden 1
Tel: (06121) 700874
TELEX: 04186183

Intel Semiconductor GmbH
Wernerstrasse 67
P.O. Box 1460
7012 Fellbach
Tel: (0711) 580082
TELEX: 7254826

Intel Semiconductor GmbH
Hindenburgstrasse 28/29
3000 Hannover 1
Tel: (0511) 852051
TELEX: 923625

HONG KONG

Intel Trading Corporation
99-105 Des Voeux Rd., Central
18F, Unit B
Hong Kong
Tel: 5-450-847
TELEX: 63869

ISRAEL

Intel Semiconductor Ltd.*
P.O. Box 2404
Haifa
Tel: 972/452 4261
TELEX: 92246511

ITALY

Intel Corporation Italia, S.p.A.
Corso Sempione 39
I-20145 Milano
Tel: 2/34.93287
TELEX: 311271

JAPAN

Intel Japan K.K.*
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagaya-ku
Tokyo 154
Tel: (03) 426-9261
TELEX: 781-28426

NETHERLANDS

Intel Semiconductor B.V.
Cometongebouw
Westblaak 106
3012 Km Rotterdam
Tel: (10) 149122
TELEX: 22283

NORWAY

Intel Norway A/S
P.O. Box 92
Hvamveien 4
N-2013
Skjetten
Tel: (2) 742 420
TELEX: 18018

SWEDEN

Intel Sweden A.B.*
Box 20092
Alpvagen 17
S-16120 Bromma
Tel: (08) 98 53 90
TELEX: 12261

SWITZERLAND

Intel Semiconductor A.G.
Forchstrasse 95
CH 8032 Zurich
Tel: 1-55 45 02
TELEX: 557 89 ich ch

UNITED KINGDOM

Intel Corporation (U.K.) Ltd.
Broadfield House
4 Between Towns Road
Cowley, Oxford OX4 3NB
Tel: (0865) 77 14 31
TELEX: 837203

Intel Corporation (U.K.) Ltd.*
5 Hospital Street
Nantwich, Cheshire CW5 5RE
Tel: (0270) 62 65 60
TELEX: 36620

Intel Corporation (U.K.) Ltd.
Dorcan House
Eldine Drive
Swindon, Wiltshire SN3 3TU
Tel: (0793) 26101
TELEX: 444447 INT SWN

*Field Application Location

U.S. AND CANADIAN SERVICE OFFICES

CALIFORNIA

Intel Corp.
1601 Old Bayshore Hwy.
Suite 345
Burlingame 94010
Tel: (415) 692-4762
TWX: 910-375-3310

Intel Corp.
2000 E. 4th Street
Suite 100
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114

Intel Corp.
7670 Opportunity Road
San Diego 92111
Tel: (714) 268-3563

Intel Corp.
3375 Scott Blvd.
Santa Clara 95051
Tel: (408) 987-8086

Intel Corp.
15335 Morrison
Sherman Oaks 91403
Tel: (213) 986-9510

COLORADO

Intel Corp.
650 South Cherry
Suite 720
Denver, 80222
Tel: (303) 321-8086
TWX: 910-931-2289

FLORIDA

Intel Corp.
1001 N.W. 62nd Street
Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

ILLINOIS

Intel Corp.
2550 Golf Road
Suite 815
Rolling Meadows 60008
Tel: (312) 981-7230
TWX: 910-253-1825

KANSAS

Intel Corp.
9393 W. 110th Street
Suite 265
Overland Park 66210
Tel: (913) 642-8080

MARYLAND

Intel Corp.
7257 Parkway Drive
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

MASSACHUSETTS

Intel Corp.
27 Industrial Avenue
Chelmsford 01824
Tel: (617) 667-8126
TWX: 710-343-6333

MICHIGAN

Intel Corp.
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 810-244-4915

MINNESOTA

Intel Corp.
7401 Metro Blvd.
Suite 355
Edina 55435
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI

Intel Corp.
502 Earth City Plaza
Suite 121
Earth City 63045
Tel: (314) 291-1990

NEW JERSEY

Intel Corp.
2450 Lemoine Avenue
Ft. Lee 07024
Tel: (201) 947-6267
TWX: 710-991-8593

OHIO

Intel Corp.
Chagrin-Brainard Bldg. #210
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736
TWX: 810-427-9298

Intel Corp.
6500 Poe Avenue
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528

OREGON

Intel Corp.
10700 S.W. Beaverton-Hillsdale Hwy.
Bldg. 2, Suite 324
Beaverton 97005
Tel: (503) 641-8086
TWX: 910-467-8741

PENNSYLVANIA

Intel Corp.
275 Commerce Drive
200 Office Center
Suite 300
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-651-2077

TEXAS

Intel Corp.
313 E. Anderson Lane
Suite 314
Austin 78752
Tel: (512) 454-3628
TWX: 910-874-1347

Intel Corp.
2925 L.B.J. Freeway
Suite 175
Dallas 75234
Tel: (214) 241-2820
TWX: 910-860-5617

Intel Corp.
6420 Richmond Avenue
Suite 280
Houston 77057
Tel: (713) 784-1300
TWX: 910-881-2490

VIRGINIA

Intel Corp.
7700 Leesburg Pike
Suite 412
Falls Church 22043
Tel: (703) 734-9707
TWX: 710-931-0625

WASHINGTON

Intel Corp.
1603 116th Ave. N.E.
Suite 114
Bellevue 98005
Tel: (206) 232-7823
TWX: 910-443-3002

WISCONSIN

Intel Corp.
150 S. Sunnyslope Road
Suite 148
Brookfield 53005
Tel: (414) 784-9060

CANADA

Intel Corp.
50 Galaxy Blvd.
Unit 12
Rexdale, Ontario
M9W4Y5
Tel: (416) 675-2105
Telex: 069-83574

Intel Corp.
39 Hwy. 7, Bells Corners
Ottawa, Ontario
K2H 8R2
Tel: (613) 829-9714
Telex: 053-4115

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Intel Corporation
Rue du Moulin à Papier 51, Boite 1,
B-1160 Brussels, Belgium

Intel Corporation
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagayu-ku
Tokyo 154, Japan

Printed in U.S.A. /C-1/181/15K/CP

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Intel Corporation
Rue du Moulin à Papier 51, Boite 1,
B-1160 Brussels, Belgium

Intel Corporation
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagayu-ku
Tokyo 154, Japan

Printed in U.S.A./C-1/181/15K/CP