8X9X USER'S MANUAL

ORDER NO.: 272457-001
DECEMBER 1993

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

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

8X9X USER'S MANUAL

CONTENTS

PAGE

MCS® 96 8X9X Architectural Overview

1

intel®

October 1990

MCS®-96 8X9X Architectural Overview

Order Number: 270250-005

MCS®-96 8X9X ARCHITECTURAL OVERVIEW

CONTENTS

CONTENTS

This overview is written about the 8X9XBH, 8X9XJF, and 8X98 devices. These devices are generically referred to as the 8X9X. All information in this overview refers to the 8X9XBH, the 8X9XJF, and the 8X98 unless otherwise noted.

The 8X9X can be separated into several sections for the purpose of describing its operation. There is a 16-bit CPU, a programmable High Speed I/O Unit, an analog to digital converter, a serial port, and a Pulse Width Modulated (PWM) output for digital to analog conversion. In addition to these functional units, there are some sections which support overall operation of the chip such as the clock generator. The CPU and the programmable I/O make the 8X9X very different from any other microcontroller. Let us first examine the CPU.

1.0 CPU OPERATION

The major components of the CPU on the 8X9X are the Register File and the RALU. Communication with the outside world is done through either the Special Function Registers (SFRs) or the Memory Controller. The RALU (Register/Arithmetic Logic Unit) does not use an accumulator, it operates directly on the 256-byte register space made up of the Register File and the SFRs. Efficient I/O operations are possible by directly controlling the I/O through the SFRs. The main benefits of this structure are the ability to quickly change context, the absence of accumulator bottleneck, and fast throughput and I/O times.

1.1 CPU Buses

A "Control Unit" and two buses connect the Register File and RALU. Figure 1 shows the CPU with its

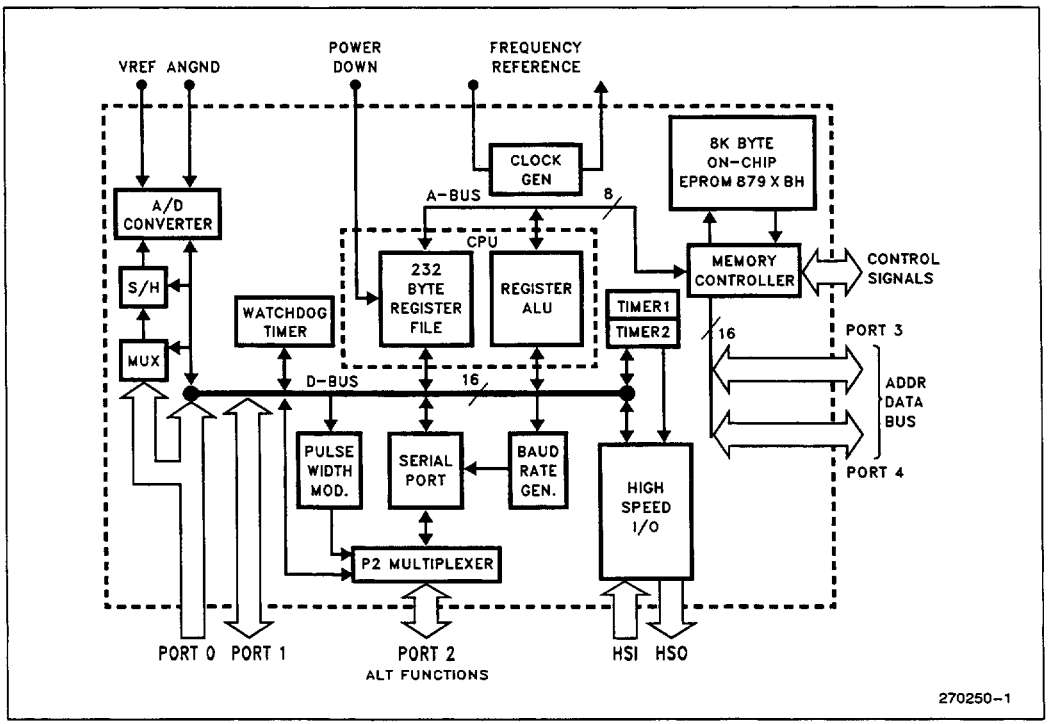


Figure 1. Block Diagram

major bus connections. The two buses are the "A-Bus" which is 8-bits wide, and the "D-Bus" which is 16-bits wide. The D-Bus transfers data only between the RALU and the Register File or Special Function Registers (SFRs). The A-Bus is used as the address bus for the above transfers or as a multiplexed address/data bus connecting to the "Memory Controller". Any accesses of either the internal ROM or external memory are done through the Memory Controller.

Within the memory controller is a slave program counter (Slave PC) which keeps track of the PC in the CPU. By having most program fetches from memory referenced to the slave PC, the processor saves time as addresses seldom have to be sent to the memory controller. If the address jumps sequence then the slave PC is loaded with a new value and processing continues. Data fetches from memory are also done through the memory controller, but the slave PC is bypassed for this operation.

1.2 CPU Register File

The Register File contains 232 bytes of RAM which can be accessed as bytes, words, or double-words. Since each of these locations can be used by the RALU, there are essentially 232 "accumulators". The first word in the Register File is reserved for use as the stack pointer so it can not be used for data when stack manipulations are taking place. Addresses for accessing the Register File and SFRs are temporarily stored in two 8-bit address registers by the CPU hardware.

1.3 RALU Control

Instructions to the RALU are taken from the A-Bus and stored temporarily in the instruction register. The Control Unit decodes the instructions and generates the correct sequence of signals to have the RALU perform the desired function. Figure 1 shows the instruction register and the control unit.

1.4 RALU

Most calculations performed by the 8X9X take place in the RALU. The RALU, shown in Figure 2, contains a 17-bit ALU, the Program Status Word (PSW), the Program Counter (PC), a loop counter, and three temporary registers. All of the registers are 16-bits or 17-bits (16+ sign extension) wide. Some of the registers have the ability to perform simple operations to offload the ALU.

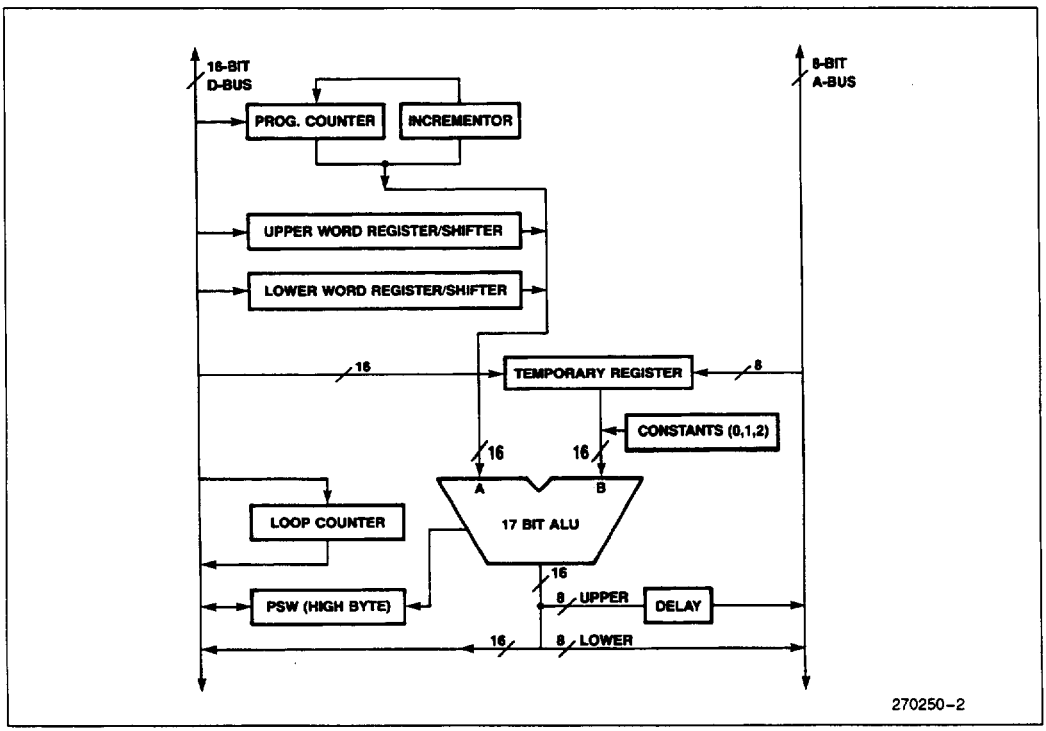


Figure 2. RALU Block Diagram

A separate incrementor is used for the PC; however, jumps must be handled through the ALU. Two of the temporary registers have their own shift logic. These registers are used for the operations which require logical shifts, including Normalize, Multiply, and Divide. The "Lower Word" register is used only when double-word quantities are being shifted, the "Upper Word" register is used whenever a shift is performed or as a temporary register for many instructions. Repetitive shifts are counted by the 5-bit "Loop Counter".

A temporary register is used to store the second operand of two operand instructions. This includes the multiplier during multiplications and the divisor during divisions. To perform subtractions, the output of this register can be complemented before being placed into the "B" input of the ALU.

The DELAY shown in Figure 2 is used to convert the 16-bit bus into an 8-bit bus. This is required as all addresses and instructions are carried on the 8-bit A-Bus. Several constants, such as 0, 1 and 2 are stored in the RALU for use in speeding up certain calculations. These come in handy when the RALU needs to make a 2's complement number or perform an increment or decrement instruction.

1.5 Internal Timing

The 8X9X requires an input clock frequency of between 6.0 MHz and 12 MHz to function. This frequency can be applied directly to XTAL1. Alternatively, since XTAL1 and XTAL2 are inputs and outputs of an inverter, it is also possible to use a crystal to generate the clock. A block diagram of the oscillator section is shown in Figure 3. Details of the circuit and suggestions for its use can be found in Section 1 of the Hardware Design chapter.

The crystal or external oscillator frequency is divided by 3 to generate the three internal timing phases as shown in Figure 4. Each of the internal phases repeat every 3 oscillator periods: 3 oscillator periods are referred to as one "state time", the basic time measurement for 8X9X operations. Most internal operations are synchronized to either Phase A, B or C, each of which have a 33% duty cycle. Phase A is represented externally by CLKOUT, a signal available on the 68-pin device. Phases B and C are not available externally. The relationships of XTAL1, CLKOUT, and Phases A, B, and C are shown in Figure 4. It should be noted that propagation delays have not been taken into account in this diagram. Details on these and other timing relationships can be found in the Hardware Design chapter.

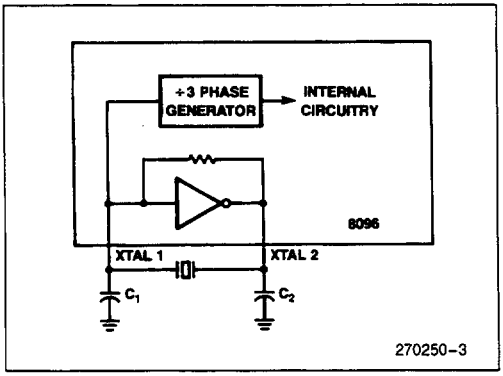


Figure 3. Block Diagram of Oscillator

The $\overline{\text{RESET}}$ line can be used to start the 8X9X at an exact time to provide for synchronization of test equipment and multiple chip systems. Use of this feature is fully explained under RESET, Section 13.

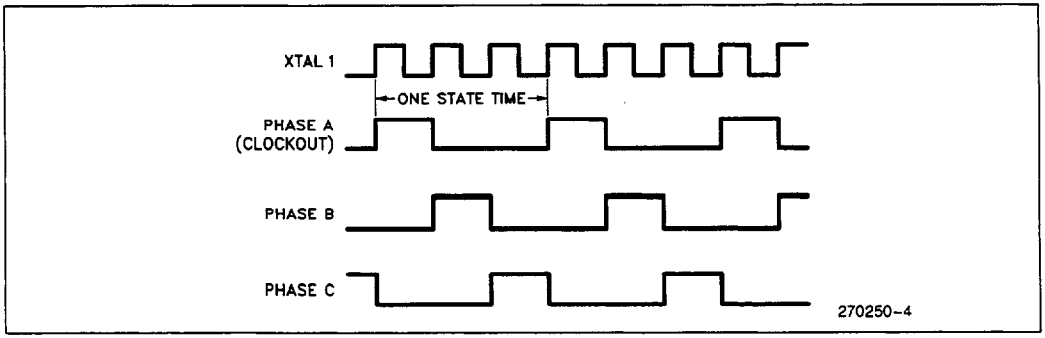


Figure 4. Internal Timings Relative to XTAL 1

2.0 MEMORY SPACE

The addressable memory space on the 8X9X consists of 64K bytes, most of which is available to the user for program or data memory. Locations which have special purposes are 0000H through 00FFH, 0100H through 01FFH (8X9XJF only), and 1FFEH through 2080H. All other locations can be used for either program or data storage or for memory mapped peripherals. A memory map is shown in Figure 5.

2.1 Register File

Locations 00H through 0FFH contain the Register File and Special Function Registers, (SFRs). No code can be executed from this internal RAM section. If an attempt to execute instructions from locations 000H through 0FFH is made, the instructions will be fetched from *external* memory. This section of external memory is reserved for use by Intel development tools. Execution of a nonmaskable interrupt (NMI) will force a

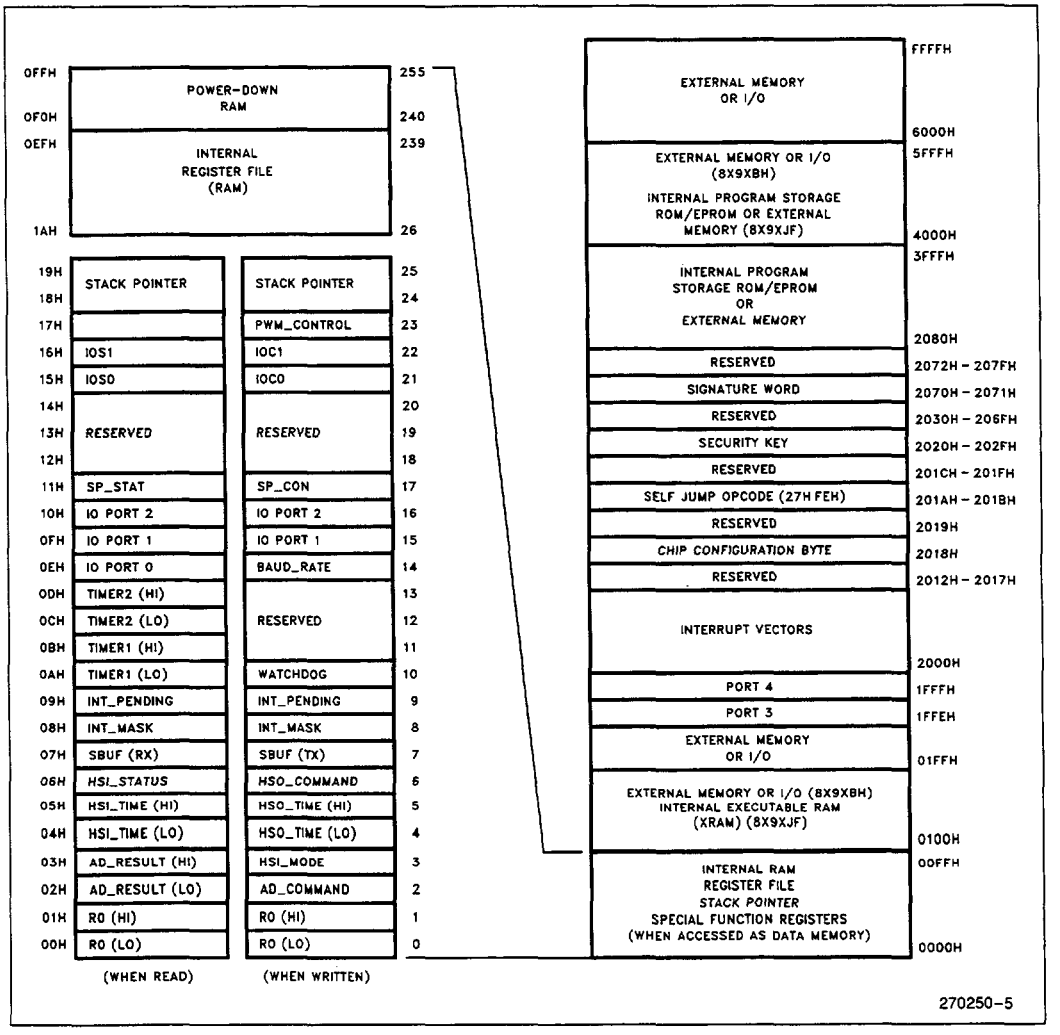


Figure 5. Memory Map

call to external location 0000H, therefore, the NMI and TRAP interrupt are also reserved for Intel development tools.

The RALU can operate on any of the 256 internal register locations. Locations 00H through 17H are used to access the SFRs. Locations 18H and 19H contain the stack pointer. These are not SFRs, and may be used as standard RAM if stack operations are not being performed. The stack pointer must be initialized by the user program and can point anywhere in the 64K memory space. The stack builds down. There are no restrictions on the use of the remaining 230 locations except that code cannot be executed from them.

2.2 Special Function Registers

All of the I/O on the 8X9X is controlled through the SFRs. Many of these registers serve two functions; one if they are read from, the other if they are written to. Figure 5 shows the locations and names of these registers. A summary of the capabilities of each of these registers is shown in Figure 6, with complete descriptions reserved for later sections.

There are several restrictions on using special function registers.

Neither the source or destination addresses of the Multiply and Divide instructions can be a writable special function register.

These registers may not be used as base or index registers for indirect or indexed instructions.

These registers can only be accessed as bytes unless otherwise specified in Figure 6. Note that some of these registers can only be accessed as words, and not as bytes.

Within the SFR space are several registers labeled "RESERVED". These registers are reserved for future expansion and test purposes. Operations should not be performed with these registers as reads from them and writes to them may produce unexpected results. For example, in some versions of the 8X9X writing to location 0CH will set both timers to 0FFFXH. This may not be the case in future products, so it should not be used as a feature.

2.3 Power Down

The upper 16 RAM locations (0F0H through 0FFH) receive their power from the V_{PD} pin. If it is desired to keep the memory in these locations alive during a power down situation, one need only keep voltage on the V_{PD} pin. The current required to keep the RAM alive is approximately 1 milliamp (refer to the data sheet for the exact specification). Both V_{CC} and V_{PD} must have power applied for normal operation. If V_{PD} is not applied the power down RAM will not function properly, even if V_{CC} is applied.

To place the 8X9X into a power down mode, the $\overline{\text{RESET}}$ pin is pulled low. Two state times later the device will be in reset. This is necessary to prevent the device from writing into RAM as the power goes down. The power may now be removed from the V_{CC} pin, the V_{PD} pin must remain within specifications. The 8X9X can remain in this state for any amount of time and the 16 RAM bytes will retain their values.

To bring the 8X9X out of power down, $\overline{\text{RESET}}$ is held low while V_{CC} is applied. Two state times after the oscillator has stabilized, the $\overline{\text{RESET}}$ pin can be pulled high. The 8X9X will begin to execute code at location 02080H 10 state times after $\overline{\text{RESET}}$ is pulled high. Figure 7 shows a timing diagram of the power down sequence. To ensure that the 2 state time minimum reset time (synchronous with CLKOUT) is met, it is recommended that 10 XTAL1 cycles be used. Suggestions for actual hardware connections are given in the Hardware Design Chapter. Reset is discussed in Section 13.

To determine if a reset is a return from power down or a complete cold start a "key" can be written into power-down RAM while the device is running. This key can be checked on reset to determine which type of reset has occurred. In this way the validity of the power-down RAM can be verified. The length of this key determines the probability that this procedure will work, however, there is always a statistical chance that the RAM will power up with a replica of the key.

Under most circumstances, the power-fail indicator which is used to initiate a power-down condition must come from the unfiltered, unregulated section of the power supply. The power supply must have sufficient storage capacity to operate the 8X9X until it has completed its reset operation.

Register	Description	Section
R0	Zero Register — Always reads as a zero, useful for a base when indexing and as a constant for calculations and compares.	3
AD_RESULT	A/D Result Hi/Low — Low and high order Results of the A/D converter (byte read only)	8
AD_COMMAND	A/D Command Register — Controls the A/D	8
HSI_MODE	HSI Mode Register — Sets the mode of the High Speed Input unit.	6
HSI_TIME	HSI Time Hi/Lo — Contains the time at which the High Speed Input unit was triggered. (word read only)	6
HSO_TIME	HSO Time Hi/Lo — Sets the time or count for the High Speed Output to execute the command in the Command Register. (word write only)	7
HSO_COMMAND	HSO Command Register — Determines what will happen at the time loaded into the HSO Time registers.	7
HSI_STATUS	HSI Status Registers — Indicates which HSI pins were detected at the time in the HSI Time registers and the current state of the pins.	6
SBUF (TX)	Transmit buffer for the serial port, holds contents to be outputted.	9
SBUF (RX)	Receive buffer for the serial port, holds the byte just received by the serial port.	9
INT_MASK	Interrupt Mask Register — Enables or disables the individual interrupts.	4
INT_PENDING	Interrupt Pending Register — Indicates that an interrupt signal has occurred on one of the sources and has not been serviced.	4
WATCHDOG	Watchdog Timer Register — Written to periodically to hold off automatic reset every 64K state times.	12
TIMER1	Timer 1 Hi/Lo — Timer 1 high and low bytes. (word read only)	5
TIMER2	Timer 2 Hi/Lo — Timer 2 high and low bytes. (word read only)	5
IOPORT0	Port 0 Register — Levels on pins of port 0.	10
BAUD_RATE	Register which determines the baud rate, this register is loaded sequentially.	9
IOPORT1	Port 1 Register — Used to read or write to Port 1.	10
IOPORT2	Port 2 Register — Used to read or write to Port 2.	10
SP_STAT	Serial Port Status — Indicates the status of the serial port.	9
SP_CON	Serial Port Control — Used to set the mode of the serial port.	9
IOS0	I/O Status Register 0 — Contains information on the HSO status	11
IOS1	I/O Status Register 1 — Contains information on the status of the timers and of the HSI.	11
IOC0	I/O Control Register 0 — Controls alternate functions of HSI pins, Timer 2 reset sources and Timer 2 clock sources.	11
IOC1	I/O Control Register 1 — Controls alternate functions of Port 2 pins, timer interrupts and HSI interrupts.	11
PWM_CONTROL	Pulse Width Modulation Control Register — Sets the duration of the PWM pulse.	8

Figure 6. SFR Summary

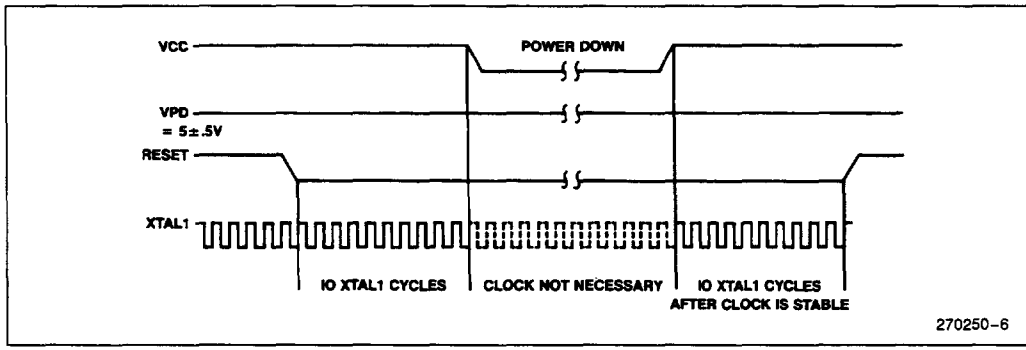


Figure 7. Power Down Timing

2.4 Reserved Memory Spaces

A listing of locations with special significance is shown in Figure 8. The locations marked "Reserved" are reserved by Intel for use in testing or future products. All reserved locations except 2019H must be filled with Hex value 0FFH to insure compatibility with future devices. Location 2019H must be filled with 20H.

Locations 1FFEH and 1FFFH are reserved for Ports 3 and 4 respectively. This is to allow easy reconstruction of these ports if external memory is used in the system. An example of reconstructing the I/O ports is given in section 7 of the Hardware Design chapter. If ports 3 and 4 are not going to be reconstructed, these locations can be treated as any other external memory location.

The 9 interrupt vectors are stored in locations 2000H through 2011H. The 9th vector is used by Intel development systems, as explained in Section 4.

Locations 2012H through 2017H are reserved for future use. Location 2018H is the Chip Configuration byte which will be discussed in the next section. The Jump-To-Self opcodes at locations 201AH and 201BH are provided for EPROM programming as detailed in the Hardware Design chapter. Locations 2020H through 202FH are the security key used with the ROM Lock feature which will be discussed in the next section. All unspecified addresses in locations 2000H through 207FH, including those marked Reserved, should be considered reserved for use by Intel.

Resetting the 8X9X causes instructions to be fetched starting from location 2080H. This location was chosen to allow a system to have up to 8K of RAM continuous with the register file. Further information on reset can be found in Section 13.

0000H-	0017H	Register Mapped I/O (SFRs)
0018H-	0019H	Stack Pointer
1FFEH-	1FFFH	Ports 3 and 4
2000H-	2011H	Interrupt Vectors
2012H-	2017H	Reserved
2018H		Chip Configuration Byte
2019H		Reserved
201AH-	201BH	"Jump to Self" Opcode (27H FEH)
201CH-	201FH	Reserved
2020H-	202FH	Security Key
2030H-	207FH	Reserved
2080H		Reset Location

Figure 8. Registers with Special Significance

2.5 Internal ROM and EPROM

When a ROM device is ordered, or an EPROM device is programmed, the internal memory locations 2080H through 3FFFH on the 8X9XBH and 8X98 and locations 2080H through 5FFFH on the 8X9XJF are user specified, as are the interrupt vectors, Chip Configuration Register and Security Key in locations 2000H through 202FH.

Instruction and data fetches from the internal ROM or EPROM occur only if the device has a ROM or EPROM, $\overline{EA}$ is tied high, and the address is between 2000H and 3FFFH on the 8X9XBH and 8X98 and between 2000H and 5FFFH on the 8X9XJF. At all other times data is accessed from either the internal RAM space or external memory and instructions are fetched from external memory. The $\overline{EA}$ pin is latched on $\overline{RESET}$ rising. Information on programming EPROMs can be found in Section 10 of the Hardware Design chapter.

Do not execute code out of the last three locations of internal ROM/EPROM.

2.6 Internal Executable RAM (XRAM)—8X9XJF only

Locations 0100H through 01FFH (8X9XJF only) contain the internal executable RAM (XRAM) space. Instruction fetches will be performed in this region if the program counter points to the addresses 0100H through 01FFH. Data accesses can also be performed from this region.

The XRAM is accessed and executed from as if it were external RAM that is contained on chip. No external bus signals will be generated when accessing the XRAM.

The XRAM is not part of the Register File. 8-bit direct addressing can not be used on this address space.

2.7 Memory Controller

The RALU talks to the memory (except for the locations in the register file and SFR space) through the memory controller which is connected to the RALU by the A-Bus and several control lines. Since the A-Bus is eight bits wide, the memory controller uses a Slave Program Counter to avoid having to always get the instruction location from the RALU. This slave PC is incremented after each fetch. When a jump or call occurs, the slave PC must be loaded from the A-Bus before instruction fetches can continue.

In addition to holding a slave PC, the memory controller contains a 4 byte queue to help speed execution. This queue is transparent to the RALU and to the user unless wait states are forced during external bus cycles. The instruction execution times shown in Section 14.8 show the normal execution times with no wait states added and the 16-bit bus selected. Reloading the slave PC and fetching the first byte of the new instruction stream takes 4 state times. This is reflected in the jump taken/not-taken times shown in the table.

2.8 System Bus

There are several operating modes on the 8X9X. The standard bus mode uses a 16-bit multiplexed address/data bus. Other bus modes include an 8-bit mode and a mode in which the bus size can dynamically be switched between 8-bits and 16-bits. In addition, there are several options available on the type of control signals used by the bus.

In the standard mode, external memory is addressed through lines AD0 through AD15 which form a 16-bit multiplexed (address/data) data bus. These lines share pins with I/O Ports 3 and 4. The falling edge of the Address Latch Enable (ALE) line is used to provide a clock to a transparent latch (74LS373) in order to de-

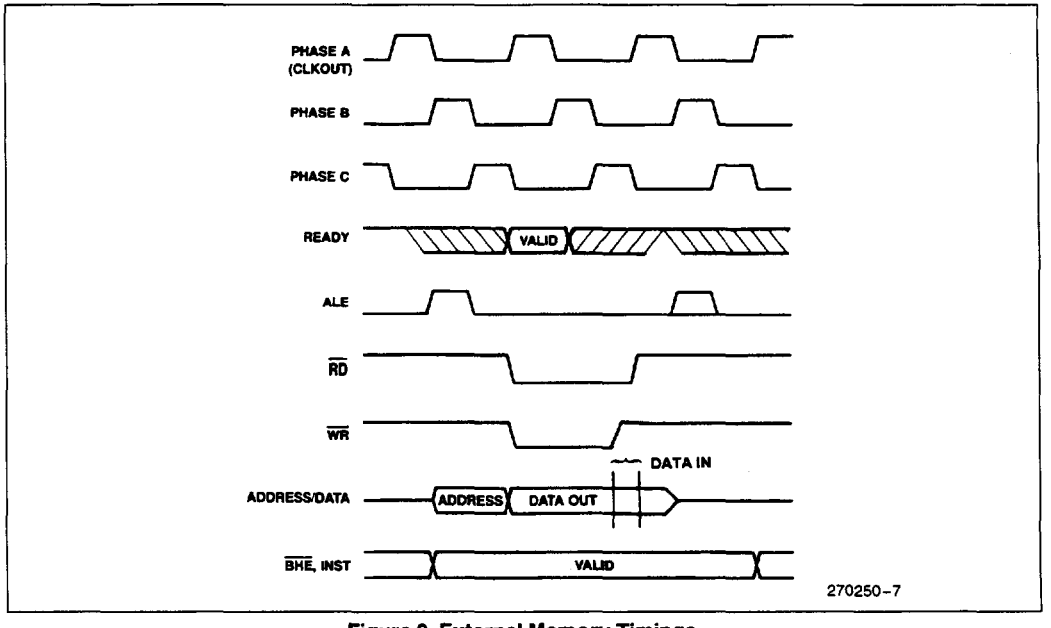


Figure 9. External Memory Timings

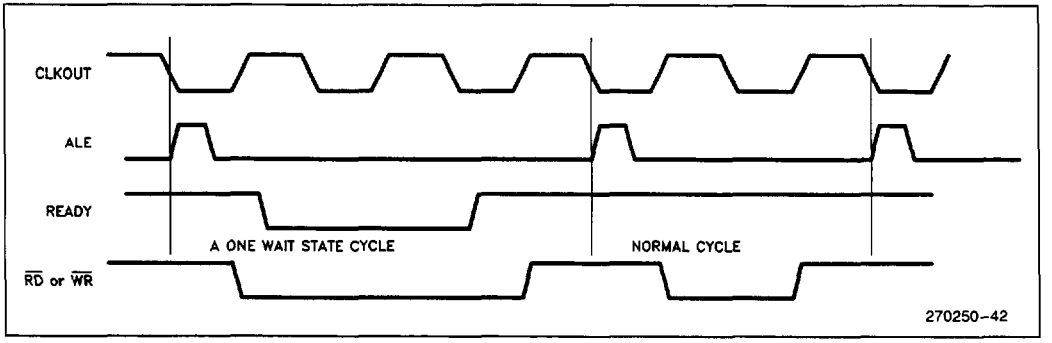


Figure 9A.

multiplex the bus. A typical circuit and the required timings are shown in Section 7 of the Hardware Design chapter. Since the 8X9X's external memory can be addressed as either bytes or words, the decoding is controlled with two lines, Bus High Enable ($\overline{BHE}$) and Address/Data Line 0 (AD0).

To avoid confusion during the explanation of the memory system it is reasonable to give names to the demultiplexed address/data signals. The address signals will be called MA0 through MA15 (Memory Address), and the data signals will be called MD0 through MD15 (Memory Data).

When $\overline{BHE}$ is active (low), the memory connected to the high byte of the data bus should be selected. When MA0 is low the memory connected to the low byte of the data bus should be selected. In this way accesses to a 16-bit wide memory can be to the low (even) byte only (MA0 = 0, $\overline{BHE}$ = 1), to the high (odd) byte only (MA0 = 1, $\overline{BHE}$ = 0), or to both bytes (MA0 = 0, $\overline{BHE}$ = 0). When a memory block is being used only for reads, $\overline{BHE}$ and MA0 need not be decoded.

TIMINGS

Figure 9 shows the idealized waveforms related to the following description of external memory manipulations. For exact timing specifications please refer to the latest data sheet. When an external memory fetch begins, the address latch enable (ALE) line rises, the address is put on AD0–AD15 and $\overline{BHE}$ is set to the required state. ALE then falls, the address is taken off the pins, and the $\overline{RD}$ (Read) signal goes low. When $\overline{RD}$ falls, external memory should present its data to the 8X9X.

READ

The data from the external memory must be on the bus and stable for a minimum of the specified set-up time before the rising edge of $\overline{RD}$. The rising edge of $\overline{RD}$ latches the information into the 8X9X. If the read is for data, the INST pin will be low when the address is valid, if it is for an instruction the INST pin will be high during this time. The 48-lead device does not have the INST pin. The INST pin will be low for the Chip Configuration Byte and Interrupt Vector fetches.

WRITE

Writing to external memory requires timings that are similar to those required when reading from it. The main difference is that the write ($\overline{WR}$) signal is used instead of the $\overline{RD}$ signal. The timings are the same until the falling edge of the $\overline{WR}$ line. At this point the 8X9X removes the address and places the data on the bus. When the $\overline{WR}$ line goes high the data should be latched to the external memory. In systems which can write to byte locations, the AD0 and $\overline{BHE}$ lines must be used to decode $\overline{WR}$ into WRite to Low byte ($\overline{WRL}$) and WRite to High byte ($\overline{WRH}$) signals. INST is always low during a write, as instructions cannot be written. The exact timing specifications for memory accesses can be found in the data sheet.

READY

A ready line is available on the 8X9X to extend the width of the RD and WR pulses in order to allow access of slow memories or for DMA purposes. If the READY line is low by the specified time after ALE falls, the 8X9X will hold the bus lines to their values at the falling edge of CLKOUT. When the READY line rises the bus cycle will continue with the next falling edge of CLKOUT. (See Figure 9A.)

Since the bus is synchronized to CLKOUT, it can be held only for an integral number of state times. If more than TYLYH nanoseconds are added the processor will act unpredictably.

There are several set-up and hold times associated with the READY signal. If these timings are not met, the device may not respond with the proper number of wait states.

For falling edges of READY, sampling is done internally on the falling edge of Phase A. Since Phase A generates CLKOUT, (after some propagation delay) the sample will be taken prior to CLKOUT falling. The timing specification for this is given as TLLYV, the time between when ALE falls and READY must be valid. If READY changes between TLLYV max and the falling edge of CLKOUT (TLLYH MIN on 48-lead devices) it would be possible to have the READY signal transitioning as it is being sampled.

This situation could cause a metastable condition which could make the device operate unpredictably.

For the rising edge of READY, sampling is done internally on the rising edge of Phase A. The rising edge logic is fully synchronized, so it is not possible to cause a metastable condition once the device is in a valid not-ready condition. To cause one wait state to occur the rising edge of READY must occur before TLLYH MAX after ALE falls. If the signal is brought up after this time two wait states may occur. If two wait states are desired, READY should be brought high within the TLLYH specification + 3 Tosc. Additional wait states can be caused by adding additional state times to the READY low time. The maximum amount of time that a device may be held not-ready is specified as TYLYH.

The 8X9X has the ability to internally limit the number of wait states to 1, 2, or 3 as determined by the value in the Chip Configuration Register, (CCR). Using the CCR for ready timing is discussed at the end of this section. If a ready limit is set, the TLLYH MAX specification is not used.

OPERATING MODES

The 8X9X supports a variety of options to simplify memory systems, interfacing requirements and ready control. Bus flexibility is provided by allowing selection of bus control signal definitions and runtime selection of the external bus width. In addition, several ready control modes are available to simplify the external hardware requirements for accessing slow devices. The Chip Configuration Register (CCR) is used to store the operating mode information.

CHIP CONFIGURATION REGISTER (CCR)

Configuration information is stored in the Chip Configuration Register (CCR). Four of the bits in the register specify the bus control mode and ready control mode. Two bits also govern the level of ROM/EPROM protection and one bit is NANDed with the BUSWIDTH pin every bus cycle to determine the bus size. The CCR bit map is shown in Figure 10. The functions associated with each bit are described in this section.

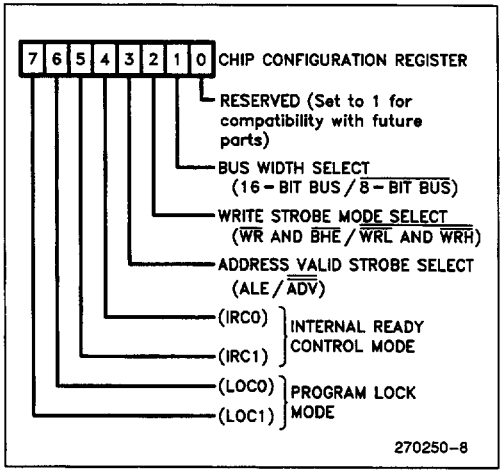


Figure 10. Chip Configuration Register

The CCR is loaded on reset with the Chip Configuration Byte, located at address 2018H. The CCR register is a non-memory mapped location that can only be written to during the reset sequence; once it is loaded it cannot be changed until the next reset occurs. The 8X9X will correctly read this location in every bus mode.

If the $\overline{EA}$ pin is set to a logical 0, the access to 2018H comes from external memory. If $\overline{EA}$ is a logical 1, the access comes from internal ROM/EPROM. If $\overline{EA}$ is +12.75V, the CCR is loaded with a byte from a separate non-memory-mapped location called PCCB (Programming CCB). The Programming mode is described in Section 10 of the Hardware Design chapter.

BUS WIDTH

The 8X9XBH and 8X9XJF external bus width can be run-time configured to operate as a standard 16-bit multiplexed address/data bus, or as an 8051 style 16-bit address/8-bit data bus. The 8X98 external bus must be configured as a 16-bit address/8-bit data bus.

During 16-bit bus cycles, Ports 3 and 4 contain the address multiplexed with data using ALE to latch the address. In 8-bit bus cycles, Port 3 is multiplexed address/data while Port 4 is address bits 8 through 15. The address bits on Port 4 are valid throughout an 8-bit bus cycle. Figure 11 shows the two options.

The bus width can be changed each bus cycle on the 8X9XBH and the 8X9XJF and is controlled using bit 1 of the CCR with the BUSWIDTH pin. If either CCR.1 or BUSWIDTH is a 0, external accesses will be over a 16-bit address/8-bit data bus. If both CCR.1 and BUSWIDTH are 1s, external accesses will be over a 16-bit address/16-bit data bus. Internal accesses are always 16-bits wide. The BUSWIDTH pin is not available on the 8X98. CCR.1 must be a 0 on the 8X98.

The bus width can be changed every external bus cycle if a 1 was loaded into CCR bit 1 at reset. If this is the case, changing the value of the BUSWIDTH pin at run-time will dynamically select the bus width. For example, the user could feed the INST line into the BUSWIDTH pin, thus causing instruction accesses to be word wide from EPROMs while data accesses are byte wide to and from RAMs. A second example would be to place an inverted version of Address bit 15 on the BUSWIDTH pin. This would make half of external memory word wide, while half is byte wide.

Since BUSWIDTH is sampled after address decoding has had time to occur, even more complex memory maps could be constructed. See the timing specifications for an exact description of BUSWIDTH timings. The bus width will be determined by bit 1 of the CCR alone on 48-pin devices since they do not have a BUSWIDTH pin.

When using an 8-bit bus, some performance degradation is to be expected. On the 8X9X , instruction execution times with an 8-bit bus will slow down if any of three conditions occur. First, word writes to external memory will cause the executing instruction to take two extra state times to complete. Second, word reads from external memory will cause a one state time extension of instruction execution time. Finally, if the prefetch queue is empty when an instruction fetch is requested, instruction execution is lengthened by one state time for each byte that must be externally acquired (worst case is the number of bytes in the instruction minus one.)

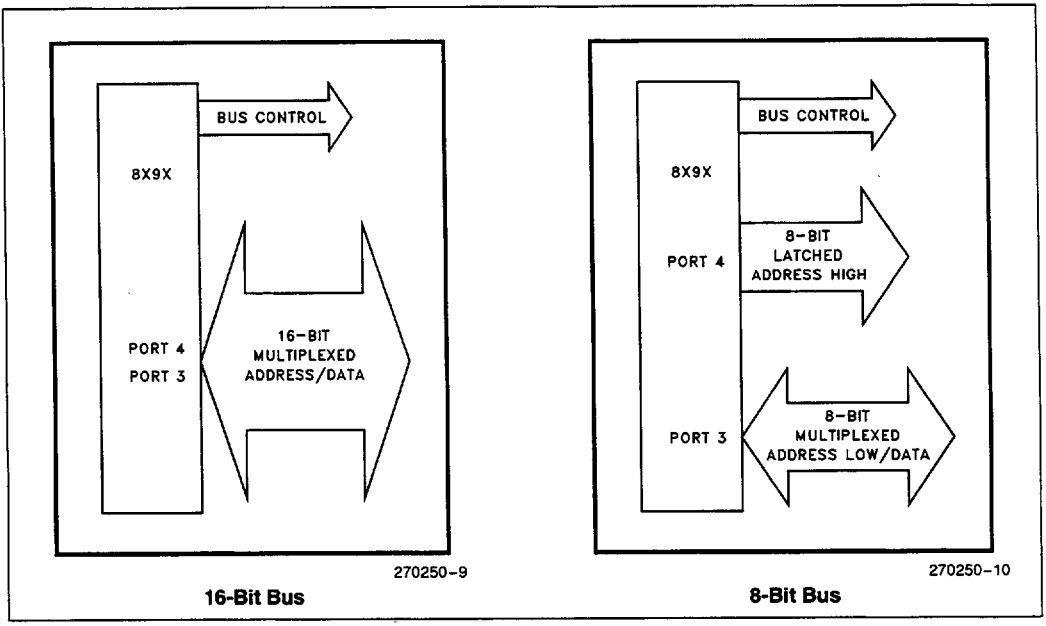


Figure 11. Bus Width Options

BUS CONTROL

Using the CCR, the 8X9X can be made to provide bus control signals of several types. Three control lines have dual functions designed to reduce external hardware. Bits 2 and 3 of the CCR specify the functions performed by these control lines. Figures 12–15 show the signals which can be modified by changing bits in the CCR, all other lines will operate as shown in Figure 9.

Standard Bus Control

If CCR bits 2 and 3 are 1s, then the standard 8X9X control signals $\overline{WR}$, $\overline{BHE}$ and ALE are provided (Figure 12). $\overline{WR}$ will come out for every write. $\overline{BHE}$ will be valid throughout the bus cycle and can be combined with $\overline{WR}$ and address line 0 to form $\overline{WRL}$ and $\overline{WRH}$. ALE will rise as the address starts to come out, and will fall to provide the signal to externally latch the address.

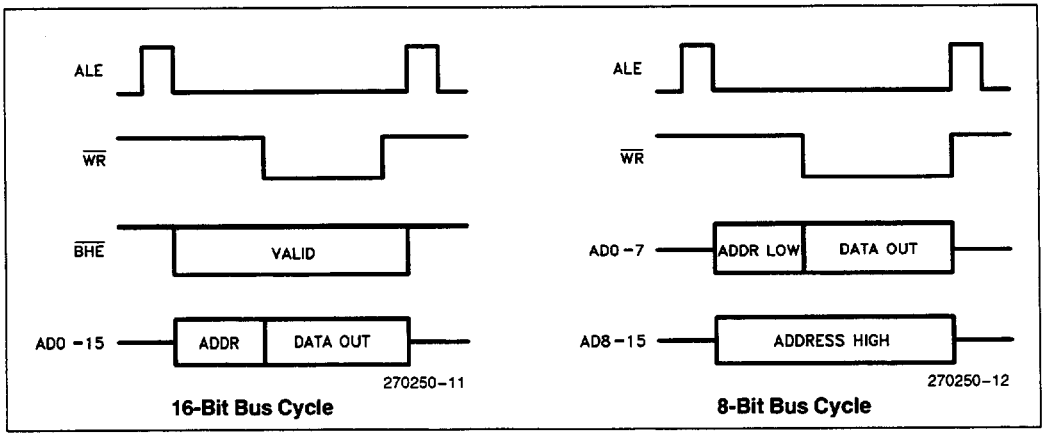


Figure 12. Standard Bus Control

Write Strobe Mode

The Write Strobe Mode eliminates the necessity to externally decode for odd or even byte writes. If CCR bit 2 is a 0, and the bus is in a 16-bit cycle, $\overline{WRL}$ and $\overline{WRH}$ signals are provided in place of $\overline{WR}$ and $\overline{BHE}$ (Figure 13). $\overline{WRL}$ will go low for all byte writes to an even address and all word writes. $\overline{WRH}$ will go low for all byte writes to an odd address and all word writes.

Write Strobe Mode is particularly well suited to memory systems latching data on the falling edge of WRITE.

$\overline{WRL}$ is provided for all 8-bit bus write cycles.

Address Valid Strobe Mode

If CCR bit 3 is a 0, then an Address Valid strobe is provided in the place of ALE (Figure 14). When the address valid mode is selected, $\overline{ADV}$ will go low after an external address is set up. It will stay low until the end of the bus cycle, where it will go inactive high. This can be used by ROM devices to provide a chip select for a single external RAM device in a minimum chip count system.

Address Valid with Write Strobe

If both CCR bits 2 and 3 are 0s, both the Address Valid strobe and the Write Strobes will be provided for bus control. Figure 15 shows these signals.

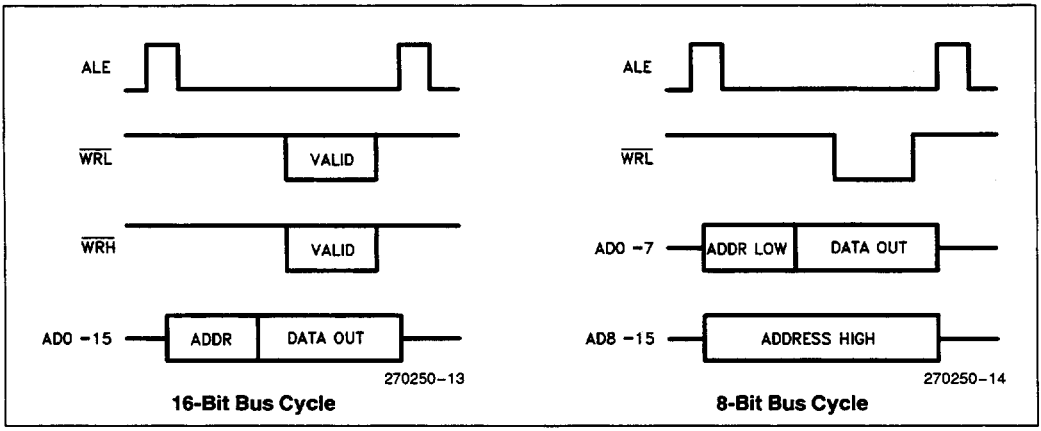


Figure 13. Write Strobe Mode

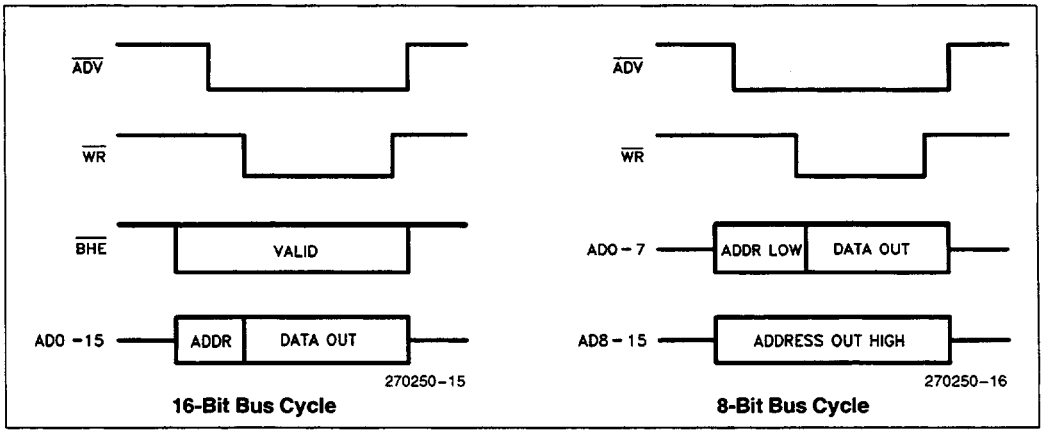


Figure 14. Address Valid Strobe Mode

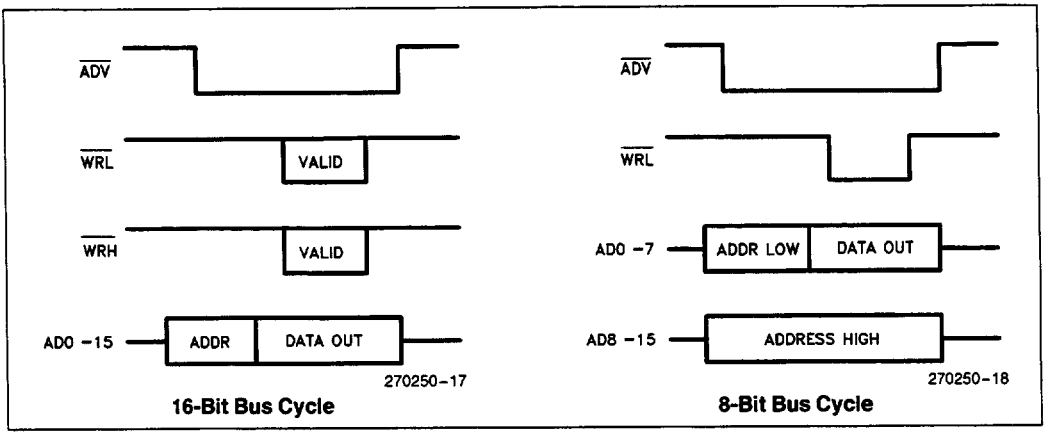


Figure 15. Write Strobe with Address Valid Strobe

READY CONTROL

To simplify ready control, four modes of internal ready control logic have been provided. The modes are chosen by properly configuring bits 4 and 5 of the CCR.

The internal ready control logic can be used to limit the number of wait states that slow devices can insert into the bus cycle. When the READY pin is pulled low, wait states will be inserted into the bus cycle until the READY pin goes high, or the number of wait states equals the number specified by CCR bits 4 and 5, whichever comes first. Table 1 shows the number of wait states that can be selected. Internal Ready control can be disabled by loading 11 into bits 4 and 5 of the CCR.

Table 1. Internal Ready Control

IRC1	IRC0	Description
0	0	Limit to 1 Wait State
0	1	Limit to 2 Wait States
1	0	Limit to 3 Wait States
1	1	Disable Internal Ready Control

This feature provides for simple ready control. For example, every slow memory chip select line could be ORed together and be connected to the READY pin with CCR bits 4 and 5 programmed to give the desired number of wait states to the slow devices.

ROM/EPROM LOCK

Four modes of program memory lock are available on the 8X9X devices. CCR bits 6 and 7 (LOC0, LOC1) select whether internal program memory can be read (or written in EPROM devices) by a program executing from external memory. The modes are shown in Table 2. Internal ROM/EPROM addresses 2020H through 3FFFH on the 8X9XBH and the 8X98 and addresses 2020H through 5FFFH on the 8X9XJF are protected from reads. 2000H through 3FFFH on the 8X9XBH and the 8X98 and 2000H through 5FFFH on the 8X9XJF are protected from writes, as set by the CCR.

Table 2. Program Lock Modes

LOC1	LOC0	Protection
0	0	Read and Write Protected
0	1	Read Protected
1	0	Write Protected
1	1	No Protection

Only code executing from internal memory can read protected internal memory, while a write protected memory can not be written to, even from internal execution. As a result of 8X9X prefetching of instructions, however, accesses to protected memory are not allowed for instructions located above 3FFAH on the 8X9XBH and the 8X98 and above 5FFAH on the 8X9XJF. This is because the lock protection mechanism is gated off of the Memory Controller's slave program counter and not the CPU program counter. If the bus controller receives a request to perform a read of protected memory, the read sequence occurs with indeterminate data being returned to the CPU. Note that the interrupt vectors and the CCR are not protected.

To provide verification and testing when the program lock feature is enabled, the 8X9X verifies the security key before programming or test modes are allowed to read from protected memory. Before protected memory can be read, the chip reads external memory locations 4020H through 402FH and compares the values

found to the internal security key located from 2020H through 202FH. Only when the values exactly match will accesses to protected memory be allowed. The details of ROM/EPROM accessing are discussed in Section 10 of the Hardware Design chapter.

3.0 SOFTWARE OVERVIEW

This section provides information on writing programs to execute in the 8X9X. Additional information can be found in the following documents:

MCS®-96 MACRO ASSEMBLER USER'S GUIDE
Order Number 186 ASM 96 (Intel Systems)
Order Number D86 ASM 96NL (DOS Systems)

C-96 USER'S GUIDE
Order Number D86 C96NL (DOS Systems)

PL/M-96 USER'S GUIDE
Order Number 186 PLM 96 (Intel Systems)
Order Number D86 PLM 96NL (DOS Systems)

Throughout this section, short sections of code are used to illustrate the operation of the device. For these sections it has been assumed that a set of temporary registers have been predeclared. The names of these registers have been chosen as follows:

AX, BX, CX, and DX are 16-bit registers.

AL is the low byte of AX, AH is the high byte.

BL is the low byte of BX

CL is the low byte of CX

DL is the low byte of DX

These are the same as the names for the general data registers used in the 8086 (80186). It is important to note, however, that in the 8X9X, these are not dedicated registers but merely the symbolic names assigned by the programmer to an eight byte region within the onboard register file.

3.1 Operand Types

The MCS®-96 architecture provides support for a variety of data types which are likely to be useful in a control application. In the discussion of these operand types that follows, the names adopted by the PLM-96 programming language will be used where appropriate. To avoid confusion, the name of an operand type will be capitalized. A "BYTE" is an unsigned eight bit variable; a "byte" is an eight bit unit of data of any type.

BYTES

BYTES are unsigned 8-bit variables which can take on the values between 0 and 255. Arithmetic and relational operators can be applied to BYTE operands but the result must be interpreted in modulo 256 arithmetic. Logical operations on BYTES are applied bitwise. Bits within BYTES are labeled from 0 to 7, with 0 being the least significant bit. There are no alignment restrictions for BYTES, so they may be placed anywhere in the MCS-96 address space.

WORDS

WORDS are unsigned 16-bit variables which can take on the values between 0 and 65535. Arithmetic and relational operators can be applied to WORD operands but the result must be interpreted modulo 65536. Logical operations on WORDS are applied bitwise. Bits within words are labeled from 0 to 15 with 0 being the least significant bit. WORDS must be aligned at even byte boundaries in the MCS-96 address space. The least significant byte of the WORD is in the even byte address and the most significant byte is in the next higher (odd) address. The address of a word is the address of its least significant byte. Word operations to odd addresses are not guaranteed to operate in a consistent manner.

SHORT-INTEGERS

SHORT-INTEGERS are 8-bit signed variables which can take on the values between −128 and +127. Arithmetic operations which generate results outside of the range of a SHORT-INTEGER will set the overflow indicators in the program status word. The actual numeric result returned will be the same as the equivalent operation on BYTE variables. There are no alignment restrictions on SHORT-INTEGERS so they may be placed anywhere in the MCS-96 address space.

INTEGERS

INTEGERS are 16-bit signed variables which can take on the values between −32,768 and 32,767. Arithmetic operations which generate results outside of the range of an INTEGER will set the overflow indicators in the program status word. The actual numeric result returned will be the same as the equivalent operation on WORD variables. INTEGERS conform to the same alignment and addressing rules as do WORDS.

BITS

BITS are single-bit operands which can take on the Boolean values of true and false. In addition to the normal support for bits as components of BYTE and WORD operands, the 8X9X provides for the direct testing of any bit in the internal register file. The MCS-96 architecture requires that bits be addressed as components of BYTES or WORDS, it does not support the direct addressing of bits that can occur in the MCS-51 architecture.

DOUBLE-WORDS

DOUBLE-WORDS are unsigned 32-bit variables which can take on the values between 0 and 4,294,967,295. The MCS-96 architecture provides direct support for this operand type only for shifts and as the dividend in a 32 by 16 divide and the product of a 16 by 16 multiply. For these operations a DOUBLE-WORD variable must reside in the on-board register file of the 8096 and be aligned at an address which is evenly divisible by 4. A DOUBLE-WORD operand is addressed by the address of its least significant byte. DOUBLE-WORD operations which are not directly supported can be easily implemented with two WORD operations. For consistency with Intel provided software the user should adopt the conventions for addressing DOUBLE-WORD operands which are discussed in Section 3.5.

LONG-INTEGERS

LONG-INTEGERS are 32-bit signed variables which can take on the values between −2,147,483,648 and 2,147,483,647. The MCS-96 architecture provides direct support for this data type only for shifts and as the dividend in a 32 by 16 divide and the product of a 16 by 16 multiply.

LONG-INTEGERS can also be normalized. For these operations a LONG-INTEGER variable must reside in the onboard register file of the 8X9X and be aligned at an address which is evenly divisible by 4. A LONG-INTEGER is addressed by the address of its least significant byte.

LONG-INTEGER operations which are not directly supported can be easily implemented with two INTEGER operations. For consistency with Intel provided software, the user should adopt the conventions for addressing LONG operands which are discussed in Section 3.5.

3.2 Operand Addressing

Operands are accessed within the address space of the 8X9X with one of six basic addressing modes. Some of the details of how these addressing modes work are hidden by the assembly language. If the programmer is to take full advantage of the architecture, it is important that these details be understood. This section will describe the addressing modes as they are handled by the hardware. At the end of this section the addressing modes will be described as they are seen through the assembly language. The six basic address modes which will be described are termed register-direct, indirect, indirect with auto-increment, immediate, short-indexed, and long-indexed. Several other useful addressing operations can be achieved by combining these basic addressing modes with specific registers such as the ZERO register or the stack pointer.

REGISTER-DIRECT REFERENCES

The register-direct mode is used to directly access a register from the 256 byte on-board register file. The register is selected by an 8-bit field within the instruction and register address and must conform to the alignment rules for the operand type. Depending on the instruction, up to three registers can take part in the calculation.

Examples

```
ADD     AX,BX,CX      ; AX:=BX+CX
MUL     AX,BX         ; AX:=AX*BX
INCB    CL            ; CL:=CL+1
```

INDIRECT REFERENCES

The indirect mode is used to access an operand by placing its address in a WORD variable in the register file. The calculated address must conform to the alignment rules for the operand type. Note that the indirect address can refer to an operand anywhere within the address space of the 8X9X, including the register file. The register which contains the indirect address is selected by an eight bit field within the instruction. An instruction can contain only one indirect reference and the remaining operands of the instruction (if any) must be register-direct references.

Examples

```
LD      AX,[AX]      ; AX:=MEM_WORD(AX)
ADDB    AL,BL,[CX]   ; AL:=BL+MEM_BYTE(CX)
POP     [AX]         ; MEM_WORD(AX):=MEM_WORD(SP) ; SP:=SP+2
```

INDIRECT WITH AUTO-INCREMENT REFERENCES

This addressing mode is the same as the indirect mode except that the WORD variable which contains the indirect address is incremented *after* it is used to address the operand. If the instruction operates on BYTES or SHORT-INTEGERS the indirect address variable will be incremented by one, if the instruction operates on WORDS or INTEGERS the indirect address variable will be incremented by two.

Examples

```
LD    AX,[BX]+     ; AX:=MEM_WORD(BX) ; BX:=BX+2
ADDB  AL,BL,[CX]+  ; AL:=BL+MEM_BYTE(CX) ; CX:=CX+1
PUSH  [AX]+        ; SP:=SP-2;
                   ;   MEM_WORD(SP):=MEM_WORD(AX)
                   ;   AX:=AX+2
```

IMMEDIATE REFERENCES

This addressing mode allows an operand to be taken directly from a field in the instruction. For operations on BYTE or SHORT-INTEGER operands this field is eight bits wide, for operations on WORD or INTEGER operands the field is 16 bits wide. An instruction can contain only one immediate reference and the remaining operand(s) must be register-direct references.

Examples

```
ADD  AX,#340  ; AX:=AX+340
PUSH #1234H   ; SP:=SP-2; MEM_WORD(SP):=1234H
DIVB AX,#10   ; AL:=AX/10; AH:=AX MOD 10
```

SHORT-INDEXED REFERENCES

In this addressing mode an eight bit field in the instruction selects a WORD variable in the register file which is assumed to contain an address. A second eight bit field in the instruction stream is sign-extended and summed with the WORD variable to form the address of the operand which will take part in the calculation. Since the eight bit field is sign-extended, the effective address can be up to 128 bytes before the address in the WORD variable and up to 127 bytes after it. An instruction can contain only one short-indexed reference and the remaining operand(s) must be register-direct references.

Examples

```
LD    AX,12[BX]     ; AX:=MEM_WORD(BX+12)
MULB  AX,BL,3[CX]   ; AX:=BL*MEM_BYTE(CX+3)
```

LONG-INDEXED REFERENCES

This addressing mode is like the short-indexed mode except that a *16-bit* field is taken from the instruction and added to the WORD variable to form the address of the operand. No sign extension is necessary. An instruction can contain only one long-indexed reference and the remaining operand(s) must be register-direct references.

Examples

```
AND   AX,BX,TABLE[CX]     ; AX:=BX AND MEM_WORD(TABLE+CX)
ST    AX,TABLE[BX]        ; MEM_WORD(TABLE+BX):=AX
ADDB  AL,BL,LOOKUP[CX]    ; AL:=BL+MEM_BYTE(LOOKUP+CX)
```

ZERO REGISTER ADDRESSING

The first two bytes in the register file are fixed at zero by the 8096 hardware. In addition to providing a fixed source of the constant zero for calculations and comparisons, this register can be used as the WORD variable in a long-indexed reference. This combination of register selection and address mode allows any location in memory to be addressed directly.

Examples

```
ADD  AX,1234[0]   ; AX:=AX+MEM_WORD(1234)
POP  5678[0]      ; MEM_WORD(5678):=MEM_WORD(SP)
                  ;  SP:=SP+2
```

STACK POINTER REGISTER ADDRESSING

The system stack pointer in the 8X9X can be accessed as register 18H of the internal register file. In addition to providing for convenient manipulation of the stack pointer, this also facilitates the accessing of operands in the stack. The top of the stack, for example, can be accessed by using the stack pointer as the WORD variable in an indirect reference. In a similar fashion, the stack pointer can be used in the short-indexed mode to access data within the stack.

Examples

```
PUSH  [SP]       ; DUPLICATE TOP_OF_STACK
LD    AX,2[SP]   ; AX:=NEXT_TO_TOP
```

ASSEMBLY LANGUAGE ADDRESSING MODES

The 8X9X assembly language simplifies the choice of addressing modes to be used in several respects:

Direct Addressing. The assembly language will choose between register-direct addressing and long-indexed with the ZERO register depending on where the operand is in memory. The user can simply refer to an operand by its symbolic name; if the operand is in the register file, a register-direct reference will be used, if the operand is elsewhere in memory, a long-indexed reference will be generated.

Indexed Addressing. The assembly language will choose between short and long indexing depending on the value of the index expression. If the value can be expressed in eight bits then short indexing will be used, if it cannot be expressed in eight bits then long indexing will be used.

The use of these features of the assembly language simplifies the programming task and should be used wherever possible.

3.3 Program Status Word

The program status word (PSW) is a collection of Boolean flags which retain information concerning the state of the user's program. The format of the PSW is shown in Figure 16. The information in the PSW can be broken down into two basic categories; interrupt control and condition flags. The PSW can be saved in the system stack with a single operation (PUSHF) and restored in a like manner (POPF).

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Z	N	V	VT	C	—	I	ST	<Interrupt Mask Reg>							

Figure 16. PSW Register

INTERRUPT FLAGS

The lower eight bits of the PSW are used to individually mask the various sources of interrupt to the 8096. A logical '1' in these bit positions enables the servicing of the corresponding interrupt. These mask bits can be accessed as an eight bit byte (INT_MASK—address 8) in the on-board register file. Bit 9 in the PSW is the global interrupt disable. If this bit is cleared then all interrupts will be locked out except for the Non Maskable Interrupt (NMI). Note that the various interrupts are collected in the INT_PENDING register even if they are locked out. Execution of the corresponding service routines will procede according to their priority when they become enabled. Further information on the interrupt structure of the 8X9X can be found in Section 4.

CONDITION FLAGS

The remaining bits in the PSW are set as side effects of instruction execution and can be tested by the conditional jump instructions.

Z. The Z (Zero) flag is set to indicate that the operation generated a result equal to zero. For the add-with-carry (ADDC) and subtract-with-borrow (SUBC) operations the Z flag is cleared if the result is non-zero but is never set. These two instructions are normally used in conjunction with the ADD and SUB instructions to perform multiple precision arithmetic. The operation of the Z flag for these instructions leaves it indicating the proper result for the entire multiple precision calculation.

N. The N (Negative) flag is set to indicate that the operation generated a negative result. Note that the N flag will be set to the algebraically correct state even if the calculation overflows.

V. The V (overflow) flag is set to indicate that the operation generated a result which is outside the range that can be expressed in the destination data type. For the SHL, SHLB and SHLL instructions, the V flag will be set if the most significant bit of the operand changes at any time during the shift.

VT. The VT (oVerflow Trap) flag is set whenever the V flag is set but can only be cleared by an instruction which explicitly operates on it such as the CLRVT or JVT instructions. The operation of the VT flag allows for the testing for a possible overflow condition at the end of a sequence of related arithmetic operations. This is normally more efficient than testing the V flag after each instruction.

C. The C (Carry) flag is set to indicate the state of the arithmetic carry from the most significant bit of the ALU for an arithmetic operation or the state of the last bit shifted out of the operand for a shift. Arithmetic Borrow after a subtract operation is the complement of the C flag (i.e. if the operation generated a borrow then C = 0).

ST. The ST (STicky bit) flag is set to indicate that during a right shift a 1 has been shifted first into the C flag and then been shifted out. The ST flag is undefined after a multiply operation. The ST flag can be used along with the C flag to control rounding after a right shift. Consider multiplying two eight bit quantities and then scaling the result down to 12 bits:

```
MULUB   AX,CL,DL    ;AX:=CL*DL
SHR     AX,#4       ;Shift right 4 places
```

If the C flag is set after the shift, it indicates that the bits shifted off the end of the operand were greater-than or equal-to one half the least significant bit (LSB) of the result. If the C flag is clear after the shift, it indicates that the bits shifted off the end of the operand were less than half the LSB of the result. Without the ST flag, the rounding decision must be made on the basis of this information alone. (Normally the result would be rounded up if the C flag is set.) The ST flag allows a finer resolution in the rounding decision:

C ST	Value of the Bits Shifted Off
0 0	Value = 0
0 1	0 < Value < ½ LSB
1 0	Value = ½ LSB
1 1	Value > ½ LSB

Figure 17. Rounding Alternatives

Imprecise rounding can be a major source of error in a numerical calculation; use of the ST flag improves the options available to the programmer.

3.4 Instruction Set

The MCS-96 instruction set contains a full set of arithmetic and logical operations for the 8-bit data types BYTE and SHORT INTEGER and for the 16-bit data types WORD and INTEGER. The DOUBLE-WORD and LONG data types (32 bits) are supported for the products of 16 by 16 multiplies and the dividends of 32

by 16 divides and for shift operations. The remaining operations on 32-bit variables can be implemented by combinations of 16-bit operations. As an example the sequence:

```
ADD     AX,CX
ADDC    BX,DX
```

performs a 32-bit addition, and the sequence

```
SUB     AX,CX
SUBC    BX,DX
```

performs a 32-bit subtraction. Operations on REAL (i.e. floating point) variables are not supported directly by the hardware but are supported by the floating point library for the 8X9X (FPAL-96) which implements a single precision subset of the proposed IEEE standard for floating point operations. The performance of this software is significantly improved by the 8X9X NORML instruction which normalizes a 32-bit variable and by the existence of the ST flag in the PSW.

In addition to the operations on the various data types, the 8X9X supports conversions between these types. LDBZE (load byte zero extended) converts a BYTE to a WORD and LDBSE (load byte sign extended) converts a SHORT-INTEGER into an INTEGER. WORDS can be converted to DOUBLE-WORDS by simply clearing the upper WORD of the DOUBLE-WORD (CLR) and INTEGERS can be converted to LONGS with the EXT (sign extend) instruction.

The MCS-96 instructions for addition, subtraction, and comparison do not distinguish between unsigned words and signed integers. Conditional jumps are provided to allow the user to treat the results of these operations as either signed or unsigned quantities. As an example, the CMPB (compare byte) instruction is used to compare both signed and unsigned eight bit quantities. A JH (jump if higher) could be used following the compare if unsigned operands were involved or a JGT (jump if greater-than) if signed operands were involved.

Table 3 summarizes the operation of each of the instructions. Complete descriptions of each instruction and its timings can be found in the Instruction Set chapter. A summary of instruction opcodes and timing is included in the quick reference section at the end of this chapter.

Table 3. Instruction Summary

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	D ← D + A	↕	↕	↕	↕	↑	—	
ADD/ADDB	3	D ← B + A	↕	↕	↕	↕	↑	—	
ADDC/ADDCB	2	D ← D + A + C	↓	↕	↕	↕	↑	—	
SUB/SUBB	2	D ← D − A	↕	↕	↕	↕	↑	—	
SUB/SUBB	3	D ← B − A	↕	↕	↕	↕	↑	—	
SUBC/SUBCB	2	D ← D − A + C − 1	↓	↕	↕	↕	↑	—	
CMP/CMPB	2	D − A	↕	↕	↕	↕	↑	—	
MUL/MULU	2	D, D + 2 ← D * A	—	—	—	—	—	?	2
MUL/MULU	3	D, D + 2 ← B * A	—	—	—	—	—	?	2
MULB/MULUB	2	D, D + 1 ← D * A	—	—	—	—	—	?	3
MULB/MULUB	3	D, D + 1 ← B * A	—	—	—	—	—	?	3
DIVU	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	↕	↑	—	2
DIVUB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	↕	↑	—	3
DIV	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	?	↑	—	
DIVB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	?	↑	—	
AND/ANDB	2	D ← D and A	↕	↕	0	0	—	—	
AND/ANDB	3	D ← B and A	↕	↕	0	0	—	—	
OR/ORB	2	D ← D or A	↕	↕	0	0	—	—	
XOR/XORB	2	D ← D (excl. or) A	↕	↕	0	0	—	—	
LD/LDB	2	D ← A	—	—	—	—	—	—	
ST/STB	2	A ← D	—	—	—	—	—	—	
LDBSE	2	D ← A; D + 1 ← SIGN(A)	—	—	—	—	—	—	3, 4
LDBZE	2	D ← A; D + 1 ← 0	—	—	—	—	—	—	3, 4
PUSH	1	SP ← SP − 2; (SP) ← A	—	—	—	—	—	—	
POP	1	A ← (SP); SP ← SP + 2	—	—	—	—	—	—	
PUSHF	0	SP ← SP − 2; (SP) ← PSW; PSW ← 0000H I ← 0	0	0	0	0	0	0	
POPF	0	PSW ← (SP); SP ← SP + 2; I ← ↕	↕	↕	↕	↕	↕	↕	
SJMP	1	PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LJMP	1	PC ← PC + 16-bit offset	—	—	—	—	—	—	5
BR [indirect]	1	PC ← (A)	—	—	—	—	—	—	
SCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 16-bit offset	—	—	—	—	—	—	5
RET	0	PC ← (SP); SP ← SP + 2	—	—	—	—	—	—	
J (conditional)	1	PC ← PC + 8-bit offset (if taken)	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Table 3. Instruction Summary (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D − 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D − 1	↕	↕	↕	↕	↑	—	
NEG/NEGB	1	D ← 0 − D	↕	↕	↕	↕	↑	—	
INC/INCB	1	D ← D + 1	↕	↕	↕	↕	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	↕	↕	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign(D)	↕	↕	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	↕	↕	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb — — — — — lsb ← 0	↕	?	↕	↕	↑	—	7
SHR/SHRB/SHRL	2	0 → msb — — — — — lsb → C	↕	?	↕	0	—	↕	7
SHRA/SHRAB/SHRAL	2	msb → msb — — — — — lsb → C	↕	↕	↕	0	—	↕	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left shift till msb = 1; D ← shift count	↕	?	0	—	—	—	7
TRAP	0	SP ← SP − 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

3.5 Software Standards and Conventions

For a software project of any size it is a good idea to modularize the program and to establish standards which control the communication between these modules. The nature of these standards will vary with the needs of the final application. A common component of all of these standards, however, must be the mechanism for passing parameters to procedures and returning results from procedures. In the absence of some overriding consideration which prevents their use, it is suggested that the user conform to the conventions adopted by the PLM-96 programming language for procedure linkage. It is a very usable standard for both the assembly language and PLM-96 environment and it offers compatibility between these environments. Another advantage is that it allows the user access to the same floating point arithmetics library that PLM-96 uses to operate on REAL variables.

REGISTER UTILIZATION

The MCS-96 architecture provides a 256 byte register file. Some of these registers are used to control register-mapped I/O devices and for other special functions such as the ZERO register and the stack pointer. The remaining bytes in the register file, some 230 of them, are available for allocation by the programmer. If these registers are to be used effectively, some overall strategy for their allocation must be adopted. PLM-96 adopts the simple and effective strategy of allocating the eight bytes between addresses 1CH and 23H as temporary storage. The starting address of this region is called PLMREG. The remaining area in the register file is treated as a segment of memory which is allocated as required.

ADDRESSING 32-BIT OPERANDS

These operands are formed from two adjacent 16-bit words in memory. The least significant word of the double word is always in lower address, even when the data is in the stack (which means that the most significant word must be pushed into the stack first). A double word is addressed by the address of its least significant byte. Note that the hardware supports some operations on double words (e.g. normalize and divide). For these operations the double word must be in the internal register file and must have an address which is evenly divisible by four.

SUBROUTINE LINKAGE

Parameters are passed to subroutines in the stack. Parameters are pushed into the stack in the order that they are encountered in the scanning of the source text. Eight-bit parameters (BYTES or SHORT-INTEGERS) are pushed into the stack with the high order byte undefined. Thirty-two bit parameters (LONG-INTEGERS, DOUBLE-WORDS, and REALS) are pushed into the stack as two 16-bit values; the most significant half of the parameter is pushed into the stack first.

As an example, consider the following PLM-96 procedure:

```
example_procedure: PROCEDURE
(param1,param2,param3);
  DECLARE param1 BYTE,
          param2 DWORD,
          param3 WORD;
```

When this procedure is entered at run time the stack will contain the parameters in the following order:

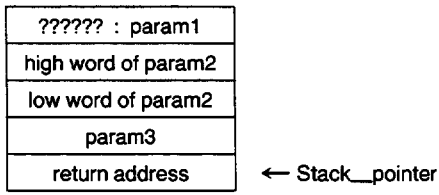


Figure 18. Stack Image

If a procedure returns a value to the calling code (as opposed to modifying more global variables) then the result is returned in the variable PLMREG. PLMREG is viewed as either an 8-, 16- or 32-bit variable depending on the type of the procedure.

The standard calling convention adopted by PLM-96 has several key features:

a) Procedures can always assume that the eight bytes of register file memory starting at PLMREG can be used as temporaries within the body of the procedure.

b) Code which calls a procedure must assume that the eight bytes of register file memory starting at PLMREG are modified by the procedure.

c) The Program Status Word (PSW—see Section 3.3) is not saved and restored by procedures so the calling code must assume that the condition flags (Z, N, V, VT, C, and ST) are modified by the procedure.

d) Function results from procedures are always returned in the variable PLMREG.

PLM-96 allows the definition of INTERRUPT procedures which are executed when a predefined interrupt occurs. These procedures do not conform to the rules of a normal procedure. Parameters cannot be passed to these procedures and they cannot return results. Since they can execute essentially at any time (hence the term interrupt), these procedures must save the PSW and PLMREG when they are entered and restore these values before they exit.

4.0 INTERRUPT STRUCTURE

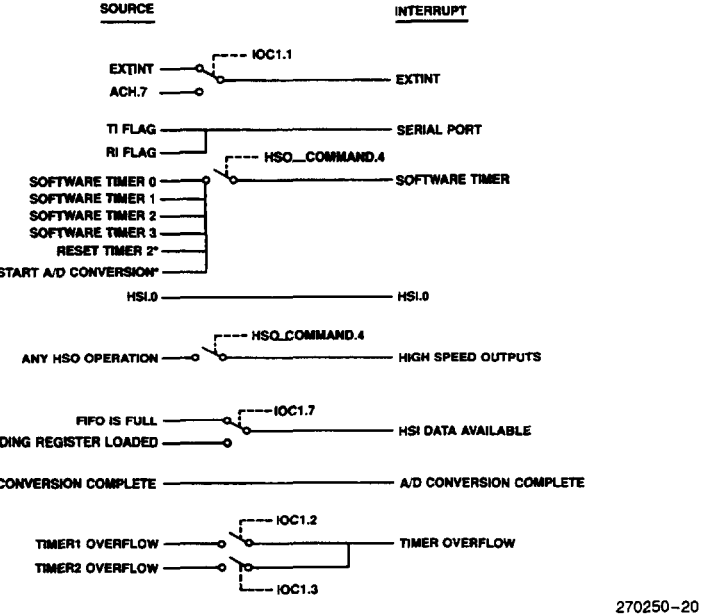
There are 21 sources of interrupts on the 8X9X. These sources are gathered into 8 interrupt types as indicated in Figure 19. The I/O control registers which control some of the sources are indicated in the figure. Each of the eight types of interrupts has its own interrupt vector as listed in Figure 20. In addition to the 8 standard interrupts, there is a TRAP instruction which acts as a software generated interrupt. This instruction is not currently supported by the MCS-96 Assembler and is reserved for use in Intel development systems.

The programmer must initialize the interrupt vector table with the starting address of the appropriate interrupt service routine. It is suggested that any unused interrupts be vectored to an error handling routine. The error routine should contain recovery code that will not further corrupt an already erroneous situation. In a debug environment, it may be desirable to have the routine lock into a jump to self loop which would be easily traceable with emulation tools. More sophisticated routines may be appropriate for production code recoveries.

Three registers control the operation of the interrupt system: Interrupt Pending, Interrupt Mask, and the PSW which contains a global disable bit. A block diagram of the system is shown in Figure 21. The transition detector looks for 0 to 1 transitions on any of the sources. External sources have a maximum transition speed of one edge every state time. If this is exceeded the interrupt may not be detected.

Vector	Vector Location (High Byte)	Vector Location (Low Byte)	Priority
Software Trap	2011H	2010H	Not Applicable
Extint	200FH	200EH	7 (Highest)
Serial Port	200DH	200CH	6
Software Timers	200BH	200AH	5
HSI.0	2009H	2008H	4
High Speed Outputs	2007H	2006H	3
HSI Data Available	2005H	2004H	2
A/D Conversion Complete	2003H	2002H	1
Timer Overflow	2001H	2000H	0 (Lowest)

Figure 20. Interrupt Vector Locations



NOTE:
*Only when initiated by the HSO unit.

Figure 19. All Possible Interrupt Sources

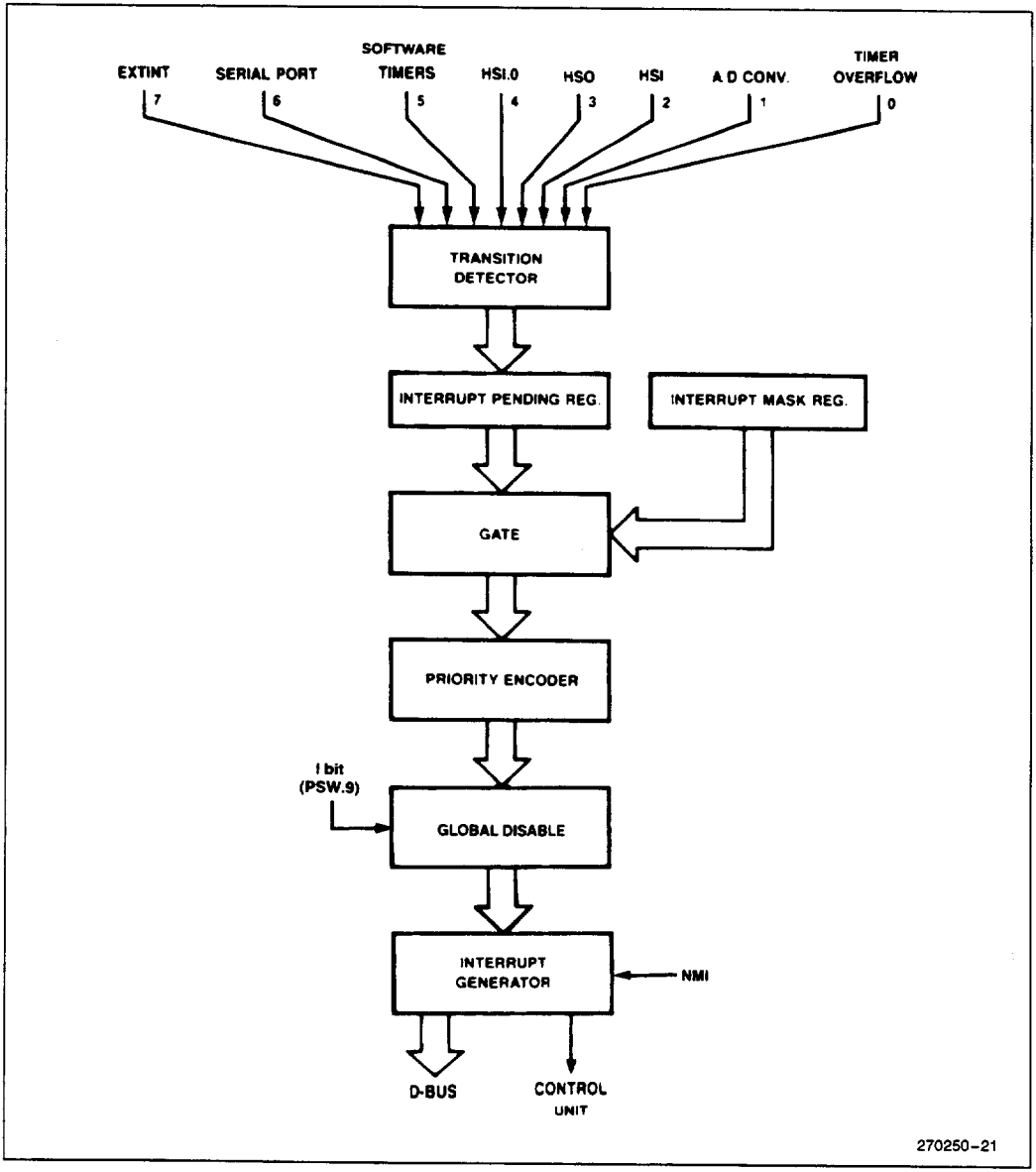


Figure 21. Block Diagram of Interrupt System

4.1 Interrupt Control

Interrupt Pending Register

When the hardware detects one of the eight interrupts it sets the corresponding bit in the pending interrupt register (INT_PENDING-09H). When the interrupt vector is taken, the pending bit is cleared. This register, the format of which is shown in Figure 22, can be read or modified as a byte register. It can be read to determine which of the interrupts are pending at any given time or modified to either clear pending interrupts or generate interrupts under software control. Any software which modifies the INT_PENDING register should ensure that the entire operation is indivisible. The easiest way to do this is to use the logical instructions in the two or three operand format, for example:

```
ANDB   INT_PENDING,#11111101B
           ; Clears the A/D Interrupt
ORB    INT_PENDING,#00000010B
           ; Sets the A/D Interrupt
```

Caution must be used when writing to the pending register to clear interrupts. If the interrupt has already been acknowledged when the bit is cleared, a 4 state time "partial" interrupt cycle will occur. This is because the 8X9X will have to fetch the next instruction of the normal instruction flow, instead of proceeding with the interrupt processing as it was going to. The effect on the program will be essentially that of an extra NOP. This can be prevented by clearing the bits using a 2 operand immediate logical, as the 8X9X holds off acknowledging interrupts during these "read/modify/write" instructions.

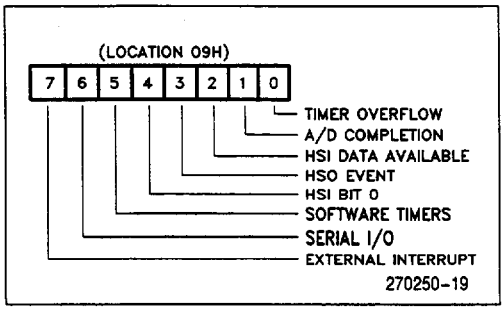


Figure 22. Interrupt Pending Register

Interrupt Mask Register

Individual interrupts can be enabled or disabled by setting or clearing bits in the interrupt mask register (INT_MASK-08H). The format of this register is the same as that of the Interrupt Pending Register shown in Figure 22.

The INT_MASK register can be read or written as byte register. A one in any bit position will enable the corresponding interrupt source and a zero will disable the source. The hardware will save any interrupts that occur by setting bits in the pending register, even if the interrupt mask bit is cleared. The INT_MASK register also can be accessed as the lower eight bits of the PSW so the PUSHF and POPF instructions save and restore the INT_MASK register as well as the global interrupt lockout and the arithmetic flags.

GLOBAL DISABLE

The processing of all interrupts can be disabled by clearing the I bit in the PSW. Setting the I bit will enable interrupts that have mask register bits which are set. The I bit is controlled by the EI (Enable Interrupts) and DI (Disable Interrupts) instructions. Note that the I bit only controls the actual servicing of interrupts. Interrupts that occur during periods of lockout will be held in the pending register and serviced on a prioritized basis when the lockout period ends.

4.2 Interrupt Priorities

The priority encoder looks at all of the interrupts which are both pending and enabled, and selects the one with the highest priority. The priorities are shown in Figure 20 (7 is highest, 0 is lowest). The interrupt generator then forces a call to the location in the indicated vector location. This location would be the starting location of the Interrupt Service Routine (ISR).

This priority selection controls the order in which pending interrupts are passed to the software via interrupt calls. The software can then implement its own priority structure by controlling the mask register (INT_MASK). To see how this is done, consider the case of a serial I/O service routine which must run at a priority level which is lower than the HSI data available interrupt but higher than any other source. The "preamble" and exit code for this interrupt service routine would look like this:

```
serial_io_isr:
 PUSHF          ; Save the PSW
                 (Includes INT_MASK)
 LDB    INT_MASK,#00000100B
 EI             ; Enable interrupts again
 ;  \
 ;   \
 ;    \
 ;     }  Service the interrupt
 ;    /
 ;   /
 ;  /
 POPF           ; Restore the PSW
 RET
```

Note that location 200CH in the interrupt vector table would have to be loaded with the value of the label serial_io_isr and the interrupt be enabled for this routine to execute.

There is an interesting chain of instruction side-effects which makes this (or any other) 8X9X interrupt service routine execute properly:

a) After the hardware decides to process an interrupt, it generates and executes a special interrupt-call instruction, which pushes the current program counter onto the stack and then loads the program counter with the contents of the vector table entry corresponding to the interrupt. The hardware will not allow another interrupt to be serviced immediately following the interrupt-call. This guarantees that once the interrupt-call starts, the first instruction of the interrupt service routine will execute.

b) The PUSHF instruction, which is now guaranteed to execute, saves the PSW in the stack and then clears the PSW. The PSW contains, in addition to the arithmetic flags, the INT_MASK register and the global disable flag (I). The hardware will not allow an interrupt following a PUSHF instruction and, by the time the LD instruction starts, all of the interrupt enable flags will be cleared. Now there is guaranteed execution of the LD INT_MASK instruction.

c) The LD INT_MASK instruction enables those interrupts that the programmer chooses to allow to interrupt the serial I/O interrupt service routine. In this example only the HSI data available interrupt will be allowed to do this but any interrupt or combination of interrupts could be enabled at this point, even the serial interrupt. It is the loading of the INT_MASK register which allows the software to establish its own priorities for interrupt servicing independently from those that the hardware enforces.

d) The EI instruction reenables the processing of interrupts.

e) The actual interrupt service routine executes within the priority structure established by the software.

f) At the end of the service routine the POPF instruction restores the PSW to its state when the interrupt-call occurred. The hardware will not allow interrupts to be processed following a POPF instruction so the execution of the last instruction (RET) is guaranteed before further interrupts can occur. The reason that this RET instruction must be protected in this fashion is that it is quite likely that the POPF instruction will reenable an interrupt which is already pending. If this interrupt were serviced before the RET instruction, then the return address to the code that was executing when the original interrupt occurred would be left on the stack. While this does not present a problem to the program flow, it could result in a stack overflow if interrupts are occurring at a high frequency. The POPF instruction also pops the INT_MASK register (part of the PSW), so any changes made to this register during a routine which ends with a POPF will be lost.

Notice that the "preamble" and exit code for the interrupt service routine does not include any code for saving or restoring registers. This is because it has been assumed that the interrupt service routine has been allocated its own private set of registers from the on-board register file. The availability of some 230 bytes of register storage makes this quite practical.

4.3 Critical Regions

Interrupt service routines must share some data with other routines. Whenever the programmer is coding those sections of code which access these shared pieces of data, great care must be taken to ensure that the integrity of the data is maintained. Consider clearing a bit in the interrupt pending register as part of a non-interrupt routine:

```
LDB     AL,INT_PENDING
ANDB    AL,#bit_mask
STB     AL,INT_PENDING
```

This code works if no other routines are operating concurrently, but will cause occasional but serious problems if used in a concurrent environment. (All programs which make use of interrupts must be considered to be part of a concurrent environment.) To demonstrate this problem, assume that the INT_PENDING register contains 00001111B and bit 3 (HSO event interrupt pending) is to be reset. The code does work for this data pattern but what happens if an HSI interrupt occurs somewhere between the LDB and the STB instructions? Before the LDB instruction INT_PENDING contains 00001111B and after the LDB instruction so does AL. If the HSI interrupt service routine executes at this point then INT_PENDING will change to 00001011B. The ANDB changes AL to 00000111B and the STB changes INT_PENDING to 00000111B. It should be 00000011B. This code sequence has manged to generate a false HSI interrupt The same basic process can generate an amazing assortment of problems and headaches. These problems can be avoided by assuring mutual exclusion which basically means that if more than one routine can change a variable, then the programmer must ensure exclusive access to the variable during the entire operation on the variable.

In many cases the instruction set of the 8X9X allows the variable to be modified with a single instruction. The code in the above example can be implemented with a single instruction.

```
ANDB    INT_PENDING,#bit_mask
```

Instructions are indivisible so mutual exclusion is ensured in this case. Changes to the INT_PENDING register must be made as a single instruction, since bits can be changed in this register even if interrupts are disabled. Depending on system configurations, several other SFRs might also need to be changed in a single instruction for the same reason.

When variables must be modified without interruption, and a single instruction can not be used, the programmer must create what is termed a critical region in which it is safe to modify the variable. One way to do this is to simply disable interrupts with a DI instruction, perform the modification, and then re-enable interrupts with an EI instruction. The problem with this approach is that it leaves the interrupts enabled even if they were not enabled at the start. A better solution is to enter the critical region with a PUSHF instruction which saves the PSW and also clears the interrupt enable flags. The region can then be terminated with a POPF instruction which returns the interrupt enable to the state it was in before the code sequence. It should be noted that some system configurations might require more protection to form a critical region. An example is a system in which more than one processor has access to a common resource such as memory or external I/O devices.

4.4 Interrupt Timing

Interrupts are not always acknowledged immediately. If the interrupt signal does not occur prior to 4 state-times before the end of an instruction, the interrupt will not be acknowledged until after the next instruction has been executed. This is because an instruction is fetched and prepared for execution a few state times before it is actually executed.

There are 6 instructions which always inhibit interrupts from being acknowledged until after the next instruction has been executed. These instructions are:

EI, DI — Enable and Disable Interrupts

POPF, PUSHF — Pop and Push Flags

SIGND — Prefix to perform signed multiply and divide (Note that this is not an ASM-96 Mnemonic, but is used for signed multiply and divide)

SOFTWARE TRAP — Software interrupt

When an interrupt is acknowledged, the interrupt pending bit is cleared, and a call is forced to the location indicated by the specified interrupt vector. This call occurs after the completion of the instruction in process, except as noted above. The procedure of getting the vector and forcing the call requires 21 state times. If the stack is in external RAM an additional 3 state times are required.

The maximum number of state times required from the time an interrupt is generated (not acknowledged) until the 8X9X begins executing code at the desired location is the time of the longest instruction, NORML (Normalize — 42 state times), plus the 4 state times prior to the end of the previous instruction, plus the response time (21 to 24 state times). Therefore, the maximum response time is 70 (42 + 4 + 24) state times. This does not include the 12 state times required for PUSHF if it is used as the first instruction in the interrupt routine or additional latency caused by having the interrupt masked or disabled. Refer to Figure 22A, Interrupt Response Time, to visualize an example of a worst case scenario.

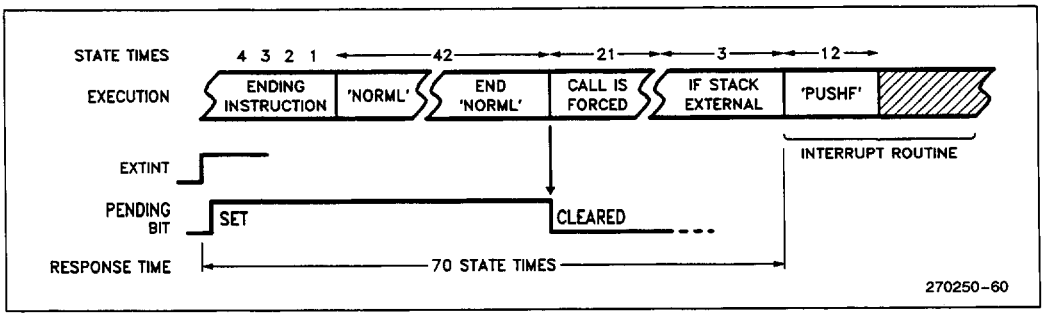


Figure 22A. Interrupt Response Time

Interrupt latency time can be reduced by careful selection of instructions in areas of code where interrupts are expected. Using 'EI' followed immediately by a long instruction (e.g. MUL, NORML, etc.) will increase the maximum latency by 4 state times, as an interrupt cannot occur between EI and the instruction following EI. The "DI", "PUSHF", "POPF" and "TRAP" instructions will also cause the same situation. Typically the PUSHF, POPF and TRAP instructions would only effect latency when one interrupt routine is already in process, as these instructions are seldom used at other times.

5.0 TIMERS

Two 16-bit timers are available for use on the 8096. The first is designated "Timer 1", the second, "Timer 2". Timer 1 is used to synchronize events to real time, while Timer 2 can be clocked externally and synchronizes events to external occurrences.

5.1 Timer 1

Timer 1 is clocked once every eight state times and can be cleared only by executing a reset. The only other way to change its value is by writing to 000CH but this is a test mode which sets both timers to 0FFFXH and should not be used in programs.

5.2 Timer 2

Timer 2 can be incremented by transitions (one count each transition, rising *and* falling) on either T2CLK or HSI.1. T2CLK is not available on the 8X98. The multiple functionality of the timer is determined by the state of I/O Control Register 0, bit 7 (IOC0.7). To ensure that all CAM entries are checked each count of Timer 2, the maximum transition speed is limited to once per eight state times. Timer 2 can be cleared by: executing a reset, by setting IOC0.1, by triggering HSO channel 0EH, or by pulling T2RST or HSI.0 high. The HSO and CAM are described in Section 7 and 8. IOC0.3 and ICO0.5 control the resetting of Timer 2. Figure 23 shows the different ways of manipulating Timer 2. It is recommended that the IOC0 register only be used once during power on reset to initialize the timers and pins, followed by an HSO command 14 to clear Timer 2 internally; or externally cleared by the T2RST or HSI.0 pins. T2RST is not available on the 8X98. Some 8X9XBH devices have errata associated with Timer 2. See the data sheets for more information.

5.3 Timer Interrupts

Both Timer 1 and Timer 2 can be used to trigger a timer overflow interrupt and set a flag in the I/O Status Register 1 (IOS1). The interrupts are controlled by IOC1.2 and IOC1.3 respectively. The flags are set in IOS1.5 and IOS1.4, respectively.

Caution must be used when examining the flags, as any access (including Compare and Jump on Bit) of IOS1 clears bits 0 through 5 including the software timer flags. It is, therefore, recommended to write the byte to a temporary register before testing bits. The general enabling and disabling of the timer interrupts are controlled by the Interrupt Mask Register bit 0. In all cases, setting a bit enables a function, while clearing a bit disables it.

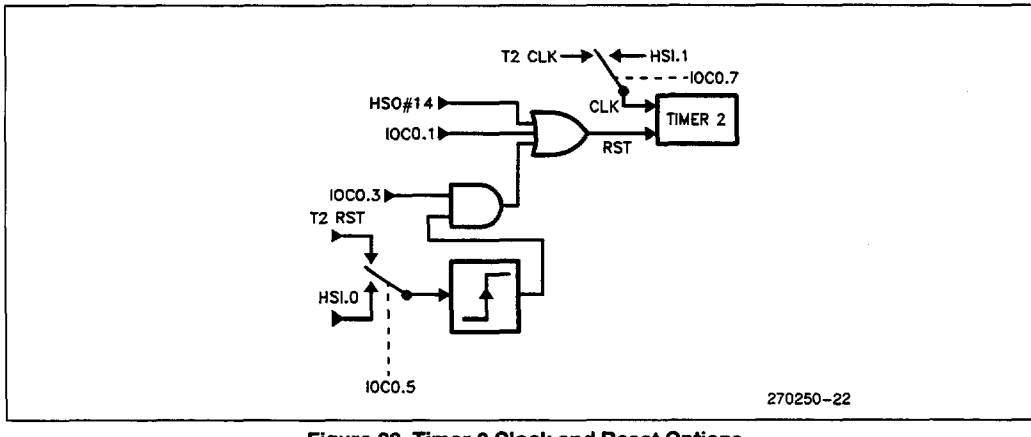


Figure 23. Timer 2 Clock and Reset Options

5.4 Timer Related Sections

The High Speed I/O unit is coupled to the timers in that the HSI records the value on Timer 1 when transitions occur and the HSO causes transitions to occur based on values of either Timer 1 or Timer 2. The baud rate generator can use the T2CLK pin as input to its counter. a complete listing of the functions of IOS1, IOC0, and IOC1 are in Section 11.

6.0 HIGH SPEED INPUTS

The High Speed Input Unit (HSI), can be used to record the time at which an event occurs with respect to Timer 1. There are 4 lines (HSI.0 through HSI.3) which can be used in this mode and up to a total of 8 events can be recorded. HSI.2 and HSI.3 are bidirectional pins which can also be used as HSO.4 and HSO.5. The I/O Control Registers (IOC0 and IOC1) are used to determine the functions of these pins. A block diagram of the HSI unit is shown in Figure 24.

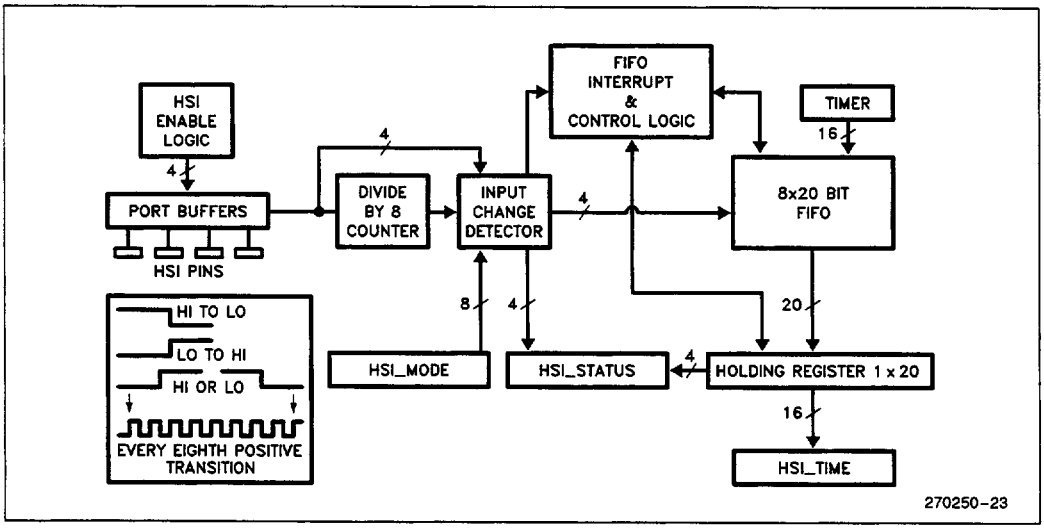


Figure 24. High Speed Input Unit

6.1 HSI Modes

There are 4 possible modes of operation for each of the HSI pins. The HSI mode register is used to control which pins will look for what type of events. The 8-bit register is set up as shown in Figure 25.

High and low levels each need to be held for at least 1 state time to ensure proper operation. The maximum input speed is 1 event every 8 state times except when the 8 transition mode is used, in which case it is 1 transition per state time. The divide by eight counter can only be zeroed in mid-count by performing a hardware reset on the 8X9X.

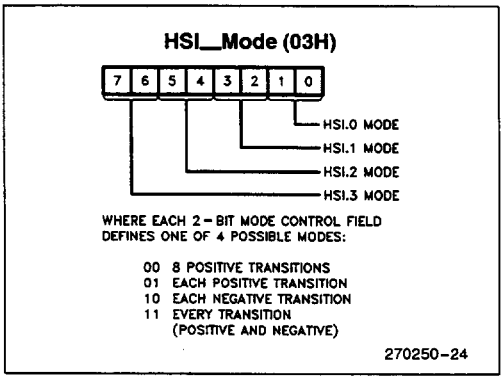


Figure 25. HSI Mode Register Diagram

The HSI lines can be individually enabled and disabled using bits in IOC0, at location 0015H. Figure 26 shows the bit locations which control the HSI pins. If the pin is disabled, transitions will not be entered in the FIFO.

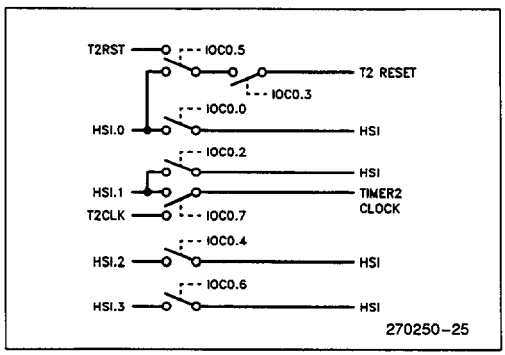


Figure 26. IOC0 Control of HSI Pin Functions

6.2 HSI FIFO

When an HSI event occurs, a 9×20 FIFO stores the 16 bits of Timer 1 and the 4 bits indicating which pins had events. It can take up to 8 state times for this information to reach the holding register. For this reason, 8 state times must be allowed between consecutive reads of HSI_TIME. When the FIFO is full, for a total of 8 events, were be stored by considering the holding register part of the FIFO. If the FIFO and holding register are full, any additional events will cause an overflow condition. Any eight consecutive events will overflow on the ninth event if the program does not clear all entries in the FIFO before the ninth event occurs. Some versions of the 8X9X have errata associated with the HSI unit. See the data sheets for more information.

6.3 HSI Interrupts

Interrupts can be generated by the HSI unit in three ways; two FIFO related interrupts and 0 to 1 transitions on the HSI.0 pin. The HSI.0 pin can generate interrupts even if it is not enabled to the HSI FIFO. Interrupts generated by this pin cause a vector through location 2008H. The FIFO related interrupts are controlled by bit 7 of I/O Control Register 1, (IOC1.7). If the bit is a 0, then an interrupt will be generated every time a value is loaded into the holding register. If it is a 1, an interrupt will only be generated when the FIFO, (independent of the holding register), has six entries in it. Since all interrupts are rising edge triggered, if IOC1.7 = 1, the processor will not be re-interrupted until the FIFO first contains 5 or less records, then contains six or more.

6.4 HSI Status

Bits 6 and 7 of the I/O Status register 1 (IOS1) indicate the status of the HSI FIFO. If bit 6 is a 1, the FIFO contains at least six entries. If bit 7 is a 1, the FIFO contains at least 1 entry and the HSI holding register has data available to be read. The FIFO may be read after verifying that it contains valid data. Caution must be used when reading or testing bits in IOS1, as this action clears bits 0–5, including the software and hardware timer overflow flags. It is best to store the byte and then test the stored value. See Section 11.

Reading the HSI is done in two steps. First, the HSI Status register is read to obtain the current state of the HSI pins and which pins had changed at the recorded time. The format of the HSI_STATUS Register is shown in Figure 27. Second, the HSI Time register is read. Reading the Time register unloads one level of the FIFO, so if the Time register is read before the Status register, the event information in the Status register will be lost. The HSI Status register is at location 06H and the HSI Time registers are in locations 04H and 05H.

If the HSI_TIME register is read without the holding register being loaded, the returned value will be indeterminate. Under the same conditions, the four bits in

HSI_STATUS indicating which events have occurred will also be indeterminate. The four HSI_STATUS bits which indicate the current state of the pins will always return the correct value.

It should be noted that many of the Status register conditions are changed by a reset, see Section 13. A complete listing of the functions of IOS0, IOS1, and IOC1 can be found in Section 11.

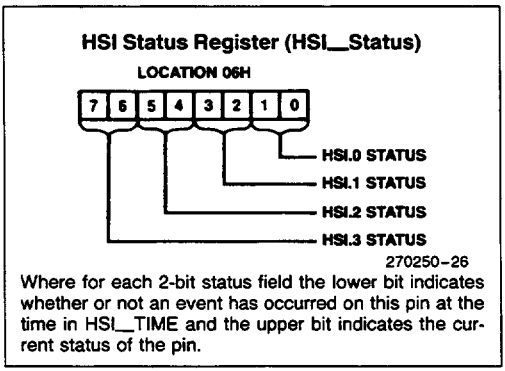
7.0 HIGH SPEED OUTPUTS

The High Speed Output unit, (HSO), is used to trigger events at specific times with minimal CPU overhead. These events include: starting an A to D conversion, resetting Timer 2, setting 4 software flags, and switching 6 output lines (HSO.0 through HSO.5). Up to eight events can be pending at one time and interrupts can be generated whenever any of these events are triggered. HSO.4 and HSO.5 are bidirectional pins which can also be used as HSI.2 and HSI.3 respectively. Bits 4 and 6 of I/O Control Register 1, (IOC1.4, IOC1.6), enable HSO.4 and HSO.5 as outputs.

The HSO unit can generate two types of interrupts. The HSO execution interrupt (vector = (2006H)) is generated (if enabled) for HSO commands which operate one or more of the six output pins. The other HSO interrupt is the software timer interrupt (vector = (200BH)) which is generated (if enabled) by any other HSO command, (e.g. triggering the A/D, resetting Timer 2 or generating a software time delay).

7.1 HSO CAM

A block diagram of the HSO unit is shown in Figure 28. The Content Addressable Memory (CAM) file is the center of control. One CAM register is compared with the timer values every state time, taking 8 state times to compare all CAM registers with the timers. This defines the time resolution of the HSO to be 8 state times (2.0 microseconds at an oscillator frequency of 12 MHz).



Where for each 2-bit status field the lower bit indicates whether or not an event has occurred on this pin at the time in HSI_TIME and the upper bit indicates the current status of the pin.

Figure 27. HSI Status Register Diagram

Each CAM register is 23 bits wide. Sixteen bits specify the time at which the action is to be carried out and 7 bits specify both the nature of the action and whether Timer 1 or Timer 2 is the reference. The format of the

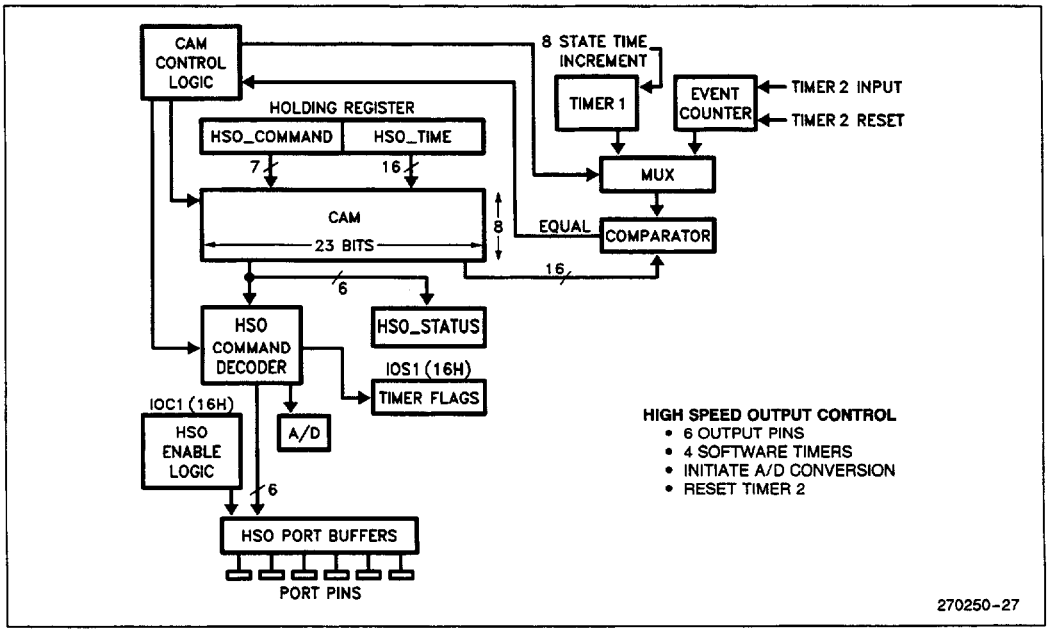


Figure 28. High Speed Output Unit

command to the HSO unit is shown in Figure 29. Note that bit 5 is ignored for command channels 8 through 0FH.

To enter a command into the CAM file, write the 7-bit "Command Tag" into location 0006H followed by the time at which the action is to be carried out into word address 0004H. The typical code would be:

```
LDB HSO_COMMAND,#what_to_do
ADD HSO_TIME,TIMER1,#when_to_do_it
```

Writing the time value loads the HSO Holding Register with both the time and the last written command tag. The command does not actually enter the CAM file until an empty CAM register becomes available.

Commands in the holding register will not execute even if their time tag is reached. Commands must be in the CAM for this to occur. Commands in the holding register can also be overwritten. Since it can take up to 8 state times for a command to move from the holding register to the CAM, 8 states must be allowed between successive writes to the CAM.

To provide proper synchronization, the minimum time that should be loaded to Timer 1 is Timer 1 + 2. Smaller values may cause the Timer match to occur 65,636 counts later than expected. A similar restriction applies if Timer 2 is used.

Care must be taken when writing the command tag for the HSO. If an interrupt occurs during the time between writing the command tag and loading the time value, and the interrupt service routine writes to the HSO time register, the command tag used in the interrupt routine will be written to the CAM at both the time specified by the interrupt routine and the time specified by the main program. The command tag from the main program will not be executed. One way of avoiding this problem would be to disable interrupts when writing commands and times to the HSO unit. See also Section 4.5.

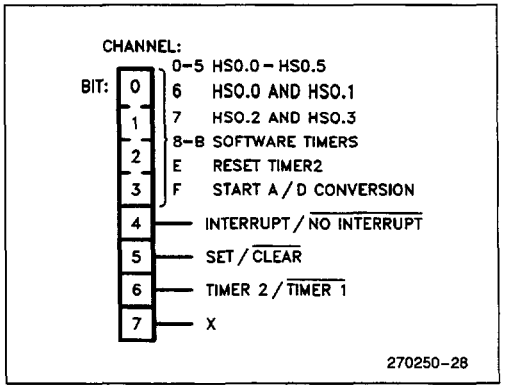


Figure 29. HSO Command Tag Format

7.2 HSO Status

Before writing to the HSO, it is desirable to ensure that the Holding Register is empty. If it is not, writing to the HSO will overwrite the value in the Holding Register. I/O Status Register 0 (IOS0) bits 6 and 7 indicate the status of the HSO unit. This register is described in Section 11. If IOS0.6 equals 0, the holding register is empty and at least one CAM register is empty. If IOS0.7 equals 0, the holding register is empty.

The programmer should carefully decide which of these two flags is the best to use for each application.

7.3 Clearing the HSO

All 8 CAM locations of the HSO are compared before any action is taken. This allows a pending external event to be cancelled by simply writing the opposite event to the CAM. However, once an entry is placed in the CAM, it cannot be removed until either the specified timer matches the written value or the chip is reset. If, as an example, a command has been issued to set HSO.1 when TIMER 1 = 1234, then entering a second command which clears HSO.1 when TIMER 1 = 1234 will result in no operation on HSO.1. Both commands will remain in the CAM until TIMER 1 = 1234.

Internal events are not synchronized to Timer 1, and therefore cannot be cleared. This includes events on HSO channels 8 through F and all interrupts. Since interrupts are not synchronized it is possible to have multiple interrupts at the same time value.

7.4 Using Timer 2 with the HSO

Timer 1 is incremented only once every 8 state-times. When it is being used as the reference timer for an HSO action, the comparator has a chance to look at all 8 CAM registers before Timer 1 changes its value. Following the same reasoning, Timer 2 has been synchronized to allow it to change at a maximum rate of once per 8 state-times. Timer 2 increments on both edges of the input signal.

When using Timer 2 as the HSO reference, caution must be taken that Timer 2 is not reset prior to the highest value for a Timer 2 match in the CAM. This is because the HSO CAM will hold an event pending until a time match occurs, if that match is to a time value on Timer 2 which is never reached, the event will remain pending in the CAM until the device is reset.

Additional caution must be used when Timer 2 is being reset using the HSO unit, since resetting Timer 2 using the HSO is an internal event and can therefore happen at any time within the eight-state-time window. This situation arises when the event is set to occur when

Timer 2 is equal to zero. If HSI.0 or the T2RST pin is used to clear Timer 2, and Timer 2 equal to zero triggers the event, then the event may not occur. This is because HSI.0 and T2RST clear Timer 2 asynchronously, and Timer 2 may then be incremented to one before the HSO CAM entry can be read and acted upon. This can be avoided by setting the event to occur when Timer 2 is equal to one. This method will ensure that there is enough time for the CAM entry recognition.

The same asynchronous nature can affect events scheduled to occur at the same time as an internal Timer 2 reset. These events should be logged into the CAM with a Timer 2 value of zero. When using this method to make a programmable modulo counter, the count will stay at the maximum Timer 2 value only until the Reset T2 command is recognized. The count will stay at zero for the transition which would have changed the count from "N" to zero, and then changed to a one on the next transition.

7.5 Software Timers

The HSO can be programmed to generate interrupts at preset times. Up to four such "Software Timers" can be in operation at a time. As each preprogrammed time is reached, the HSO unit sets a Software Timer Flag. If the interrupt bit in the command tag was set then a Software Timer Interrupt will also be generated. The interrupt service routine can then examine I/O Status register 1 (IOS1) to determine which software timer expired and caused the interrupt. When the HSO resets Timer 2 or starts an A to D conversion, it can also be programmed to generate a software timer interrupt but there is no flag to indicate that this has occurred.

If more than one software timer interrupt occurs in the same time frame it is possible that multiple software timer interrupts will be generated.

Each read or test of any bit in IOS1 will clear bits 0 through 5. Be certain to save the byte before testing it unless you are only concerned with 1 bit. See also Section 11.5.

A complete listing of the functions of IOS0, IOS1, and IOC1 can be found in Section 11. The Timers are described in Section 5 and the HSI is described in Section 6.

8.0 ANALOG INTERFACE

The 8X9X can easily interface to analog signals using its Analog to Digital Converter and its Pulse-Width-Modulated (PWM) output and HSO Unit. There are 8 inputs to the 10-bit A to D converter on the 8X9XBH and 8X9XJF. There are 4 inputs on the 8X98. The PWM and HSO units provide digital signals which can be filtered for use as analog outputs.

8.1 Analog Inputs

A to D conversion is performed on one input at a time using successive approximation with a result equal to the ratio of the input voltage divided by the analog supply voltage. If the ratio is 1.00, then the result will be all ones. The A/D converter is available on selected members of the MCS-96 family. See Section 14 for the device selection matrix.

Each conversion on the 8X9X requires 88 state-times (22 μs at 12 MHz) independent of the accuracy desired or value of input voltage. The input voltage must be in the range of 0 to V_{REF}, the analog reference and supply voltage. For proper operation, V_{REF} (the reference voltage and analog power supply) must be between 4.5V and 5.5V. The A/D result is calculated from the formula:

$$1023 \times (\text{input voltage-ANGND})/(V_{REF}\text{-ANGND})$$

It can be seen from this formula that changes in V_{REF} or ANGND effect the output of the converter. This can be advantageous if a ratiometric sensor is used since these sensors have an output that can be measured as a proportion of V_{REF}.

ANGND must be tied to V_{SS} (digital ground) in order for the 8X9X to operate properly. This common connection should be made as close to the chip as possible, and using good bulk and high frequency by-pass capacitors to decouple power supply variations and noise from the circuit. Analog design rules call for one and only one common connection between analog and digital returns to eliminate unwanted ground variations.

The A/D converter has sample and hold. The sampling window is open for 4 state times which are included in the 88 state-time conversion period. The exact timings of the A/D converter can be found in Section 3 of the Hardware Design chapter.

8.2 A/D Commands

Analog signals can be sampled by any one of the 8 analog input pins (ACH0 through ACH7) which are shared with Port 0. ACH7 can also be used as an external interrupt if IOC1.1 is set (see Sections 4 and 11). The A/D Command Register, at location 02H, selects which channel is to be converted and whether the conversion should start immediately or when the HSO (Channel #0FH) triggers it. The A/D command register must be written to for each conversion, even if the HSO is used as the trigger. A to D commands are formatted as shown in Figure 30.

The command register is double buffered so it is possible to write a command to start a conversion triggered by the HSO while one is still in progress. Care must be taken when this is done since if a new conversion is started while one is already in progress, the conversion in progress is cancelled and the new one is started. When a conversion is started, the result register is cleared. For this reason the result register must be read before a new conversion is started or data will be lost.

8.3 A/D Results

Results of the analog conversions are read from the A/D Result Register at locations 02H and 03H. Although these addresses are on a word boundary, they must be read as individual bytes. Information in the A/D Result register is formatted as shown in Figure 31. Note that the status bit may not be set until 8 state

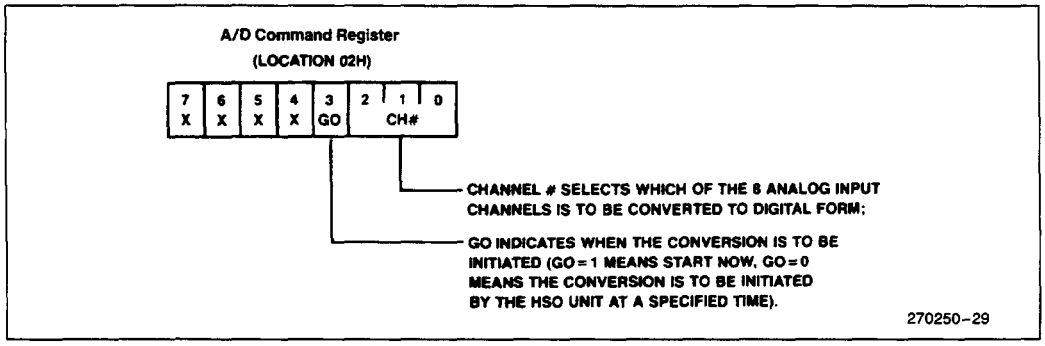


Figure 30. A/D Command Register

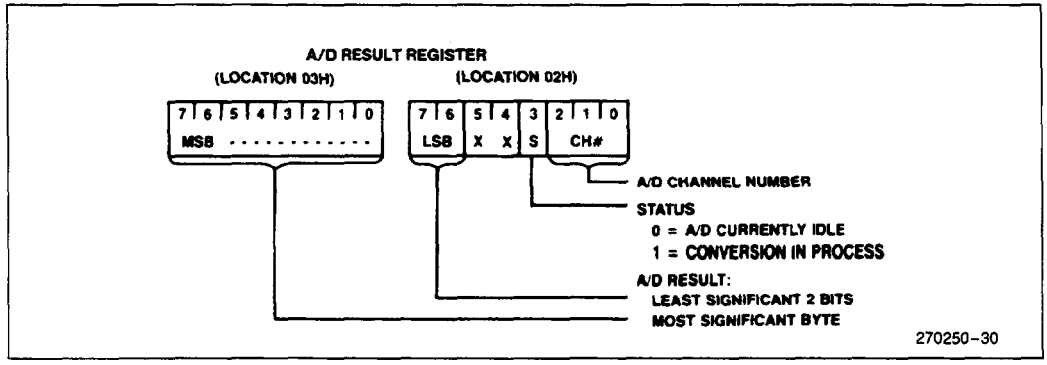


Figure 31. A/D Result Register

times after the go command, so it is necessary to wait 8 state times before testing it. Information on using the HSO is in Section 7.

8.4 Pulse Width Modulation Output (D/A)

Digital to analog conversion can be done with the Pulse Width Modulation output; a block diagram of the circuit is shown in Figure 32. The 8-bit counter is incremented every state time. When it equals 0, the PWM output is set to a one. When the counter matches the value in the PWM register, the output is switched low. When the counter overflows, the output is once again switched high. A typical output waveform is shown in Figure 33. Note that when the PWM register equals 00, the output is always low. Additionally, the PWM register will only be reloaded from the temporary latch when the counter overflows. This means that the compare circuit will not recognize a new value to compare against until the counter has expired the remainder of the current 8-bit count.

The output waveform is a variable duty cycle pulse which repeats every 256 state times (64 μs at 12 MHz). Changes in the duty cycle are made by writing to the PWM register at location 17H. There are several types of motors which require a PWM waveform for most efficient operation. Additionally, if this waveform is integrated it will produce a DC level which can be changed in 256 steps by varying the duty cycle.

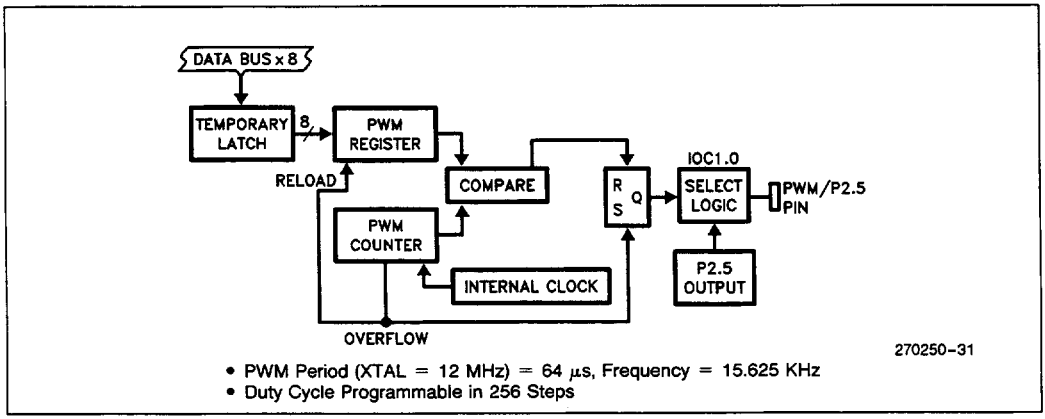


Figure 32. Pulse Width Modulated (D/A) Output

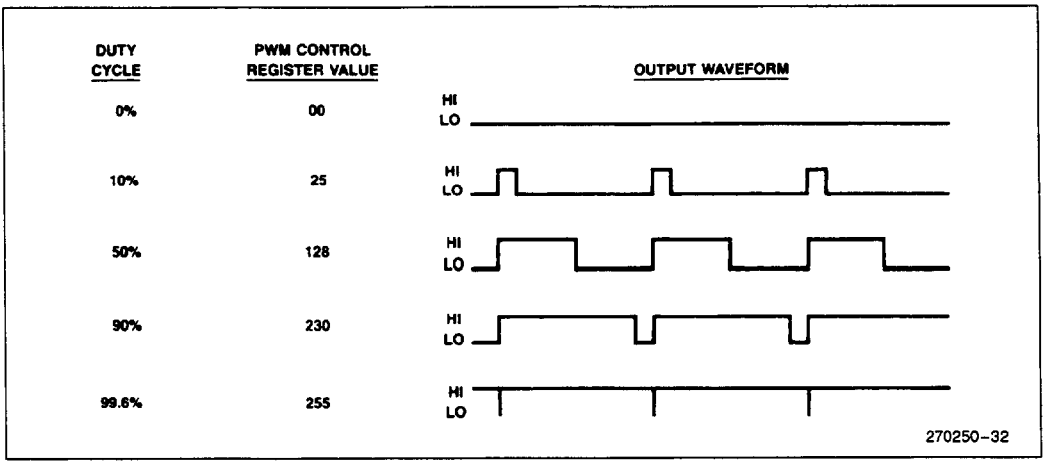


Figure 33. Typical PWM Outputs

Details about the hardware required for smooth, accurate D/A conversion can be found in Section 4 of the Hardware Design chapter. Typically, some form of buffer and integrator are needed to obtain the most usefulness from this feature.

The PWM output shares a pin with Port 2, pin 5 so that these two features cannot be used at the same time. IOC1.0 equal to 1 selects the PWM function instead of the standard port function. More information on IOC1 is in Section 11.

8.5 PWM Using the HSO

The HSO unit can be used to generate PWM waveforms with very little CPU overhead. If the HSO is not being used for other purposes, a 4 line PWM unit can be made by loading the on and off times into the CAM in sets of 4. The CAM would then always be loaded and only 2 interrupts per PWM period would be needed.

9.0 SERIAL PORT

The serial port on the 8X9X has 3 asynchronous and one synchronous mode. The asynchronous modes are full duplex, meaning they can transmit and receive at the same time. The receiver is double buffered so that the reception of a second byte can begin before the first byte has been read. The port is functionally compatible with the serial port on the MCS-51 family of microcontrollers, although the software used to control the ports is different.

Control of the serial port is handled through the Serial Port Control/Status Register at location 11H. Figure 37 shows the layout of this register. The details of using it to control the serial port will be discussed in Section 9.2.

Data to and from the serial port is transferred through SBUF (rx) and SBUF (tx), both located at 07H. Although these registers share the same address, they are physically separate, with SBUF (rx) containing the data received by the serial port and SBUF (tx) used to hold data ready for transmission. The program cannot write to SBUF (rx) or read from SBUF (tx).

The baud rate at which the serial port operates is controlled by an independent baud rate generator. The inputs to this generator can be either the XTAL1 or the T2CLK pin. Details on setting up the baud rate are given in Section 9.3.

9.1 Serial Port Modes

MODE 0

Mode 0 is a synchronous mode which is commonly used for shift register based I/O expansion. In this mode the TXD pin outputs a set of 8 pulses while the RXD pin either transmits or receives data. Data is transferred 8 bits at a time with the LSB first. A diagram of the relative timing of these signals is shown in Figure 34. Note that this is the only mode which uses RXD as an output.

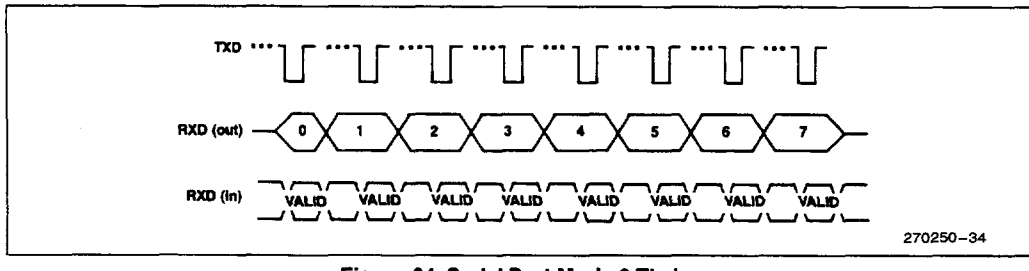


Figure 34. Serial Port Mode 0 Timing

Although it is not possible to transmit and receive at the same time using this mode, two external gates and a port pin can be used to time-multiplex the two functions. An example of multiplexing transmit and receive is discussed in Section 6.1 of the Hardware Design chapter.

MODE 1

Mode 1 is the standard asynchronous communications mode. The data frame used in this mode is shown in Figure 35. It consists of 10 bits; a start bit (0), 8 data bits (LSB first), and a stop bit (1). If parity is enabled, (the PEN bit is set to a 1), an even parity bit is sent instead of the 8th data bit and parity is checked on reception.

MODE 2

Mode 2 is the asynchronous 9th bit recognition mode. This mode is commonly used with Mode 3 for multiprocessor communications. Figure 36 shows the data frame used in this mode. It consists of a start bit (0), 9 data bits (LSB first), and a stop bit (1). When transmitting, the 9th bit can be set to a one by setting the TB8 bit in the control register before writing to SBUF (tx). The TB8 bit is cleared on every transmission, so it must be set prior to writing to SBUF (tx) each time it is desired. During reception, the serial port interrupt and the Receive Interrupt (RI) bit will not be set unless the 9th bit being received is set. This provides an easy way to have selective reception on a data link. Parity cannot be enabled in this mode.

MODE 3

Mode 3 is the asynchronous 9th bit mode. The data frame for this mode is identical to that of Mode 2. The transmission differences between Mode 3 and Mode 2 are that parity can be enabled (PEN = 1) and cause the 9th data bit to take the even parity value. The TB8 bit can still be used if parity is not enabled (PEN = 0). When in Mode 3, a reception always causes an interrupt, regardless of the state of the 9th bit. The 9th bit is stored if PEN = 0 and can be read in bit RB8. If PEN = 1 then RB8 becomes the Receive Parity Error (RPE) flag.

9.2 Controlling the Serial Port

Control of the serial port is done through the Serial Port Control (SP_CON) and Serial Port Status (SP_STAT) registers shown in Figure 37. Writing to location 11H accesses SP_CON while reading it access SP_STAT. Note that reads of SP_STAT will return indeterminate data in the lower 5 bits and writing to the upper 3 bits of SP_CON has no effect on chip functionality. The TB8 bit is cleared after each transmission and both TI and RI are cleared whenever SP_STAT (not SP_CON) is accessed. Whenever the TXD pin is used for the serial port it must be enabled by setting IOC1.5 to a 1. IOC1 is discussed further in Section 11.3. Information on the hardware connections and timing of the serial port is in Section 6 of the Hardware Design chapter.

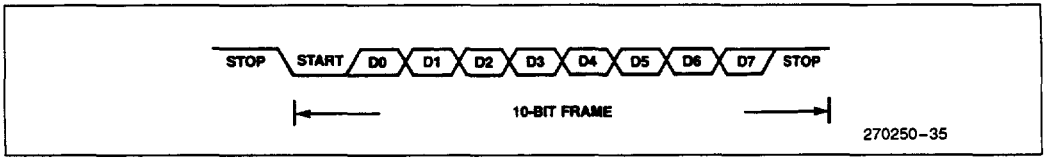


Figure 35. Serial Port Frame—Mode 1

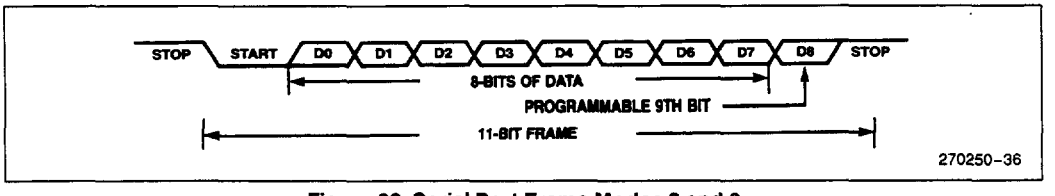
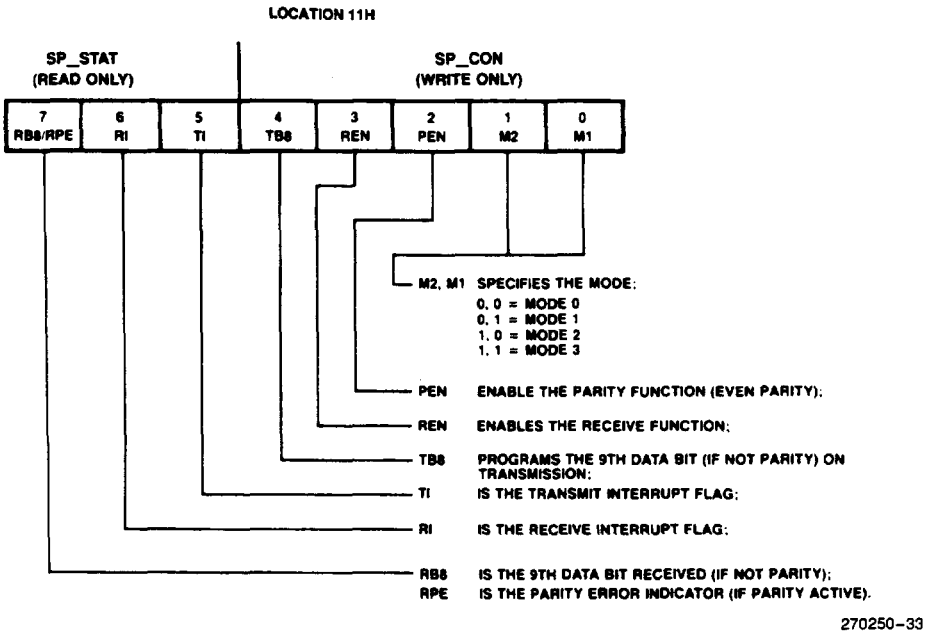


Figure 36. Serial Port Frame Modes 2 and 3



NOTE:
TI and RI are cleared when SP__STAT is read.

Figure 37. Serial Port Control/Status Register

In Mode 0, if REN = 0, writing to SBUF (tx) will start a transmission. Causing a rising edge on REN, or clearing RI with REN = 1, will start a reception. Setting REN = 0 will stop a reception in progress and inhibit further receptions. To avoid a partial or complete undesired reception, REN must be set to zero before RI is cleared. This can be handled in an interrupt environment by using software flags or in straight-line code by using the Interrupt Pending register to signal the completion of a reception.

In the asynchronous modes, writing to SBUF (tx) starts a transmission. A falling edge on RXD will begin a reception if REN is set to 1. New data placed in SBUF (tx) is held and will not be transmitted until the end of the stop bit has been sent.

In all modes, the RI flag is set after the last data bit is sampled approximately in the middle of the bit time. Also for all modes, the TI flag is set after the last data bit (either 8th or 9th) is sent, also in the middle of the bit time. The flags clear when SP__STAT is read, but do not have to be clear for the port to receive or transmit. The serial port interrupt bit is set as a logical OR of the RI and TI bits. Note that changing modes will reset the Serial Port and abort any transmission or reception in progress on the channel. If the T_X and R_X pins are tied together for loopback testing, the RI flag will be written first.

9.3 Determining Baud Rates

Baud rates in all modes are determined by the contents of a 16-bit register at location 000EH. This register must be loaded sequentially with 2 bytes (least significant byte first). The serial port will not function between the loading of the first and second bytes. The MSB of this register selects one of two sources for the input frequency to the baud rate generator. If it is a 1, the frequency on the XTAL1 pin is selected, if not, the external frequency from the T2CLK pin is used. It should be noted that the maximum speed of T2CLK is one transition every 2 state times, with a minimum period of 16 XTAL1 cycles. This provides the needed synchronization to the internal serial port clocks.

The unsigned integer represented by the lower 15 bits of the baud rate register defines a number B, where B has a maximum value of 32767. The baud rate for the four serial modes using either XTAL1 or T2CLK as the clock source is given by:

Using XTAL1:

$$\text{Mode 0: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{XTAL1 frequency}}{4 * (B + 1)}; \quad B \neq 0$$

$$\text{Others: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{XTAL1 frequency}}{64 * (B + 1)}$$

Using T2CLK:

$$\text{Mode 0: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{T2CLK frequency}}{B}; \quad B \neq 0$$

$$\text{Others: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{T2CLK frequency}}{16 * B}; \quad B \neq 0$$

Note that B cannot equal 0, except when using XTAL1 in other than mode 0.

Common baud rate values, using XTAL1 at 12 MHz, are shown below.

Baud Rate	Baud Register Value	
	Mode 0	Others
9600	8137H	8013H
4800	8270H	8026H
2400	84E1H	804DH
1200	89C3H	809BH
300	A70FH	8270H

The maximum baud rates are 1.5 Mbaud synchronous and 187.5 Kbaud asynchronous with 12 MHz on XTAL1.

9.4 Multiprocessor Communications

Mode 2 and 3 are provided for multiprocessor communications. In Mode 2 if the received 9th data bit is not 1, the serial port interrupt is not activated. The way to use this feature in multiprocessor systems is described below.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address frame which identifies the target slave. An address frame will differ from a data frame in that the 9th data bit is 1 in an address frame and 0 in a data frame. Slaves in Mode 2 will not be interrupted by a data frame. An address frame, however, will interrupt all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave switches to Mode 3 to receive the coming data frames, while the slaves that were not addressed stay in Mode 2 and go on about their business.

10.0 I/O PORTS

There are five 8-bit I/O ports on the 8096. Some of these ports are input only, some are output only, some are bidirectional and some have alternate functions. In addition to these ports, the HSI/O unit can be used to provide extra I/O lines if the timer related features of these lines are not needed.

Input ports connect to the internal bus through an input buffer. Output ports connect through an output buffer to an internal register that hold the bits to be output. Bidirectional ports consist of an internal register, an input buffer, and an output buffer.

Port 0 is an input port which is also used as the analog input for the A to D converter. Port 1 is a quasi-bidirectional port. Port 2 contains three types of port lines: quasi-bidirectional, input and output. The input and output lines are shared with other functions in the 8X9X as shown in Table 4. Ports 3 and 4 are open-drain bidirectional ports which share their pins with the address/data bus.

Table 4. Port 2 Alternate Functions

Port	Function	Alternate Function	Controlled by
P2.0	Output	TXD (Serial Port Transmit)	IOC1.5
P2.1	Input	RXD (Serial Port Receive M1-3)	N/A
	Output	RXD (Serial Port Output M0)	
P2.2	Input	EXTINT (External Interrupt)	IOC1.1
P2.3	Input	T2CLK (Timer 2 Input)	IOC0.7
P2.4	Input	T2RST (Timer 2 Reset)	IOC0.5
P2.5	Output	PWM (Pulse-Width Modulation)	IOC1.0
P2.6	Quasi-Bidirectional		
P2.7	Quasi-Bidirectional		

Section 2 of the Hardware Design chapter contains additional information on the timing, drive capabilities, and input impedances of I/O pins.

10.1 Input Ports

Input ports and pins can only be read. There are no output drivers on these pins. The input leakage of these pins is in the microamp range. The specific values can be found in the data sheet for the device being considered.

In addition to acting as a digital input, each line of Port 0 can be selected to be the input of the A to D converter as discussed in Section 8. The pins on Port 0 are tested

to have D.C. leakage of 3 microamps or less, as specified in the data sheet for the device being considered. The capacitance on these pins is approximately 5 pF and will instantaneously increase by around 5 pF when the pin is being sampled by the A to D converter.

The 8X98 devices only have 4 Port 0 pins.

The 8X9X samples the input to the A/D for 4 state times at the beginning of the conversion. Details on the A to D converter can be found in Section 8 of this chapter and in Section 3 of the Hardware Design chapter.

10.2 Quasi-Bidirectional Ports

Port 1, Port 2.6 and Port 2.7 are quasi-bidirectional ports. Port 1, Port 2.6 and Port 2.7 are not available on the 8X98. "Quasi-bidirectional" means that the port pin has a weak internal pullup that is always active and an internal pulldown which can be on to output a 0, or off to output a 1. If the internal pulldown is left off (by writing a 1 to the pin), the pin's logic level can be controlled by an external pulldown. If the external pulldown is on, it will input a 0 to the 8X9X, if it is off, a 1 will be input. From the user's point of view, the main difference between a quasi-bidirectional port and a standard input port is that the quasi-bidirectional port will source current if externally pulled low. It will also pull itself high if left unconnected.

In parallel with the weak internal pullup is a much stronger internal pullup that is activated for one state time when the pin is internally driven from 0 to 1. This is done to speed up the 0-to-1 transition time. When this pullup is on the pin can typically source 30 milliamps to V_{SS}.

When the processor writes to the pins of a quasi-bidirectional port it actually writes into a register which in turn drives the port pin. When the processor reads these ports, it senses the status of the pin directly. If a port pin is to be used as an input then the software should write a one to its associated SFR bit, this will cause the low-impedance pull-down device to turn off and leave the pin pulled up with a relatively high impedance pullup device which can be easily driven down by the device driving the input.

If some pins of a port are to be used as inputs and some are to be used as outputs the programmer should be careful when writing to the port.

Particular care should be exercised when using XOR opcodes or any opcode which is a read-modify-write instruction. It is possible for a Quasi-Bidirectional Pin to be written as a one, but read back as a zero if an external device (i.e., a transistor base) is pulling the pin below V_{IH}. See the Hardware Design Chapter Section 2.2 for further details on using the Quasi-Bidirectional Ports.

10.3 Output Ports

Output pins include the bus control lines, the HSO lines, and some of Port 2. These pins can only be used as outputs as there are no input buffers connected to them. It is not possible to use immediate logical instructions such as XOR PORT2, #00111B to toggle these pins. The output currents on these ports is higher than that of the quasi-bidirectional ports.

10.4 Ports 3 and 4/AD0-15

These pins have two functions. They are either bidirectional ports with open-drain outputs or System Bus pins which the memory controller uses when it is accesing off-chip memory. If the $\overline{EA}$ line is low, the pins always act as the System Bus. Otherwise they act as bus pins only during a memory access. If these pins are being used as ports and bus pins, ones must be written to them prior to bus operations.

Accessing Port 3 and 4 as I/O is easily done from internal registers. Since the LD and ST instructions require the use of internal registers, it may be necessary to first move the port information into an internal location before utilizing the data. If the data is already internal, the LD is unnecessary. For instance, to write a word value to Port 3 and 4 . . .

```
LD intreg, portdata  ; register ← data
                     ; not needed if already internal

ST intreg, 1FFEH     ; register → Port 3 and 4
```

To read Port 3 and 4 requires that "ones" be written to the port registers to first setup the input port configuration circuit. Note that the ports are reset to this input condition, but if zeroes have been written to the port, then ones must be re-written to any pins which are to be used as inputs. Reading Port 3 and 4 from a previously written zero condition is as follows . . .

```
LD intregA, #0FFFFH ; setup port change mode pattern

ST intregA, 1FFEH   ; register → Port 3 and 4
                    ; LD & ST not needed if previously
                    ; written as ones

LD intregB, 1FFEH   ; register ← Port 3 and 4
```

Note that while the format of the LD and ST instructions are similar, the source and destination directions change.

When acting as the system bus the pins have strong drivers to both V_{CC} and V_{SS}. These drivers are used whenever data is being output on the system bus and are not used when data is being output by Ports 3 and 4. Only the pins and input buffers are shared between the bus and the ports. The ports use different output buffers which are configured as open-drain, and require pullup resistors. (open-drain is the MOS version of open-collector.) The port pins and their system bus functions are shown in Table 5.

Table 5. P3,4/AD0–15 Pins

Port Pin	System Bus Function
P3.0	AD0
P3.1	AD1
P3.2	AD2
P3.3	AD3
P3.4	AD4
P3.5	AD5
P3.6	AD6
P3.7	AD7
P4.0	AD8
P4.1	AD9
P4.2	AD10
P4.3	AD11
P4.4	AD12
P4.5	AD13
P4.6	AD14
P4.7	AD15

11.0 STATUS AND CONTROL REGISTERS

There are two I/O Control registers, IOC0 and IOC1. IOC0 controls Timer 2 and the HSI lines. IOC1 controls some pin functions, interrupt sources and 2 HSO pins.

Whenever input lines are switched between two sources, or enabled, it is possible to generate transitions on these lines. This could cause problems with respect to edge sensitive lines such as the HSI lines, Interrupt line, and Timer 2 control lines.

11.1 I/O Control Register 0 (IOC0)

IOC0 is located at 0015H. The four HSI lines can be enabled or disabled to the HSI unit by setting or clearing bits in IOC0. Timer 2 functions including clock and reset sources are also determined by IOC0. The control bit locations are shown in Figure 38. IOC0 is for initialization only.

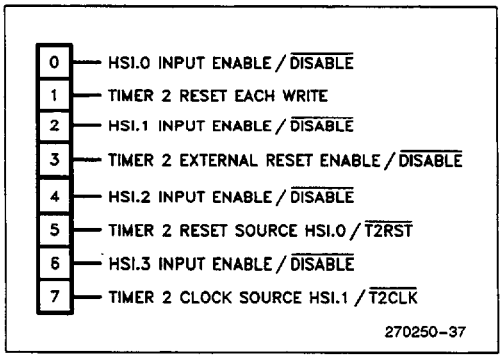


Figure 38. I/O Control Register 0 (IOC0)

11.2 I/O Control Register 1 (IOC1)

IOC1 is used to select some pin functions and enable or disable some interrupt sources. Its location is 0016H. Port pin P2.5 can be selected to be the PWM output instead of a standard output. The external interrupt source can be selected to be either EXTINT (same pin as P2.2) or Analog Channel 7 (ACH7, same pin as P0.7). Timer 1 and Timer 2 overflow interrupts can be individually enabled or disabled. The HSI interrupt can be selected to activate either when there is 1 FIFO entry or 7. Port pin P2.0 can be selected to be the TXD output. HSO.4 and HSO.5 can be enabled or disabled to the HSO unit. More information on interrupts is available in Section 4. The positions of the IOC1 control bits are shown in Figure 39.

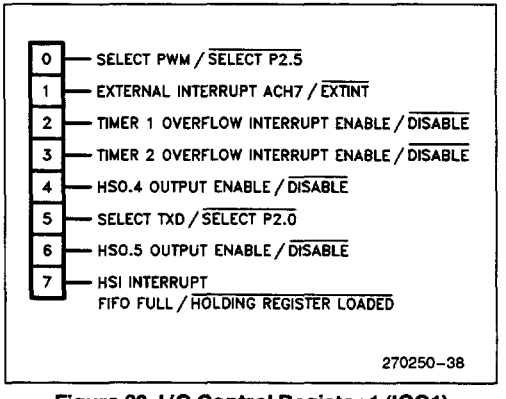


Figure 39. I/O Control Register 1 (IOC1)

11.3 I/O Status Register 0 (IOS0)

There are two I/O Status registers, IOS0 and IOS1. IOS0, located at 0015H, holds the current status of the HSO lines and CAM. The status bits of IOS0 are shown in Figure 40.

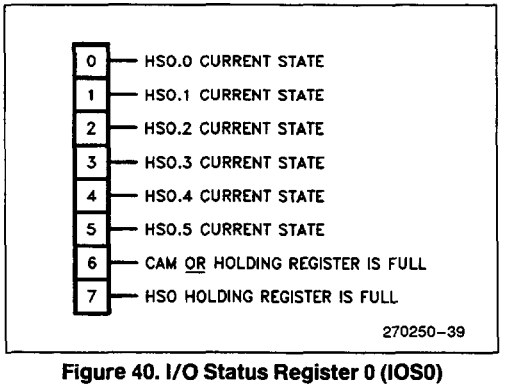


Figure 40. I/O Status Register 0 (IOS0)

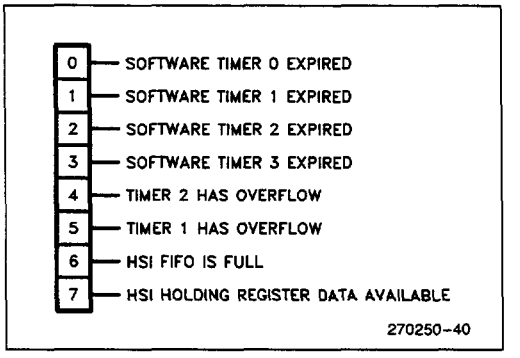


Figure 41. HSIO Status Register 1 (IOS1)

11.4 I/O Status Register 1 (IOS1)

IOS1 is located at 016H. It contains status bits for the timers and the HSI/O. The positions of these bits are shown in Figure 41.

Whenever the processor reads this register all of the time-related flags (bits 5 through 0) are cleared. This applies not only to explicit reads such as:

```
LDB    AL,IOS1
```

but also to implicit reads such as:

```
JB     IOS1.3,somewhere_else
```

which jumps to somewhere_else if bit 3 of IOS1 is set. In most cases this situation can best be handled by having a byte in the register file which is used to maintain an image of lower five bits of the register. Any time a hardware timer interrupt or a HSO software timer interrupt occurs the byte can be updated:

```
ORB    IOS1_image,IOS1
```

leaving IOS1_image containing all the flags that were set before plus all the new flags that were read and cleared from IOS1. Any other routine which needs to sample the flags can safely check IOS1_image. Note that if these routines need to clear the flags that they have acted on, then the modification of IOS1_image must be done from inside a critical region (see Section 4.4).

12.0 WATCHDOG TIMER

The WatchDog Timer (WDT) provides a means to recover gracefully from a software upset. When the watchdog is enabled it will initiate a hardware reset unless the software clears it every 64K state times.

The WDT is implemented as an 8-bit timer with an 8-bit prescaler. The prescaler is not synchronized, so the timer will overflow between 65280 and 65535 state times after being reset. When the timer overflows it pulls down the $\overline{\text{RESET}}$ pin for at least one state time, resetting the 8X9X and any other devices tied to the $\overline{\text{RESET}}$ line. If a large capacitor is connected to the line, the pin may take a long time to go low. This will effect the length of time the pin is low and the voltage on the pin when it is finished falling. Section 1.4 of the Hardware Design chapter contains more information about reset hardware connections.

The WDT is enabled the first time it is cleared. Once it is enabled, it can only be disabled by resetting the 8X9X. The internal bit which controls the watchdog can typically maintain its state through power glitches as low as V_{SS} and as high as 7.0V for up to one millisecond.

Enabling and clearing the WDT is done by writing a "01EH" followed by a "0E1H" to the WDT register at location 0AH. This double write is used to help prevent accidental clearing of the timer.

12.1 Software Protection Hints

Glitches and noise on the PC board can cause software upsets, typically by changing either memory locations or the program counter. These changes can be internal to the chip or be caused by bad data returning to the chip.

There are both hardware and software solutions to noise problems, but the best solution is good design practice and a few ounces of prevention. The software can be designed so that the watchdog times out if the program does not progress properly. The watchdog will also time-out if the software error was due to ESD (Electrostatic Discharge) or other hardware related problems. This prevents the controller from having a malfunction for longer than 16 milliseconds if a 12 MHz oscillator is used.

When using the WDT to protect software it is desirable to reset it from only one place in code. This will lessen the chance that an undesired WDT reset will occur. The section of code that resets the WDT should monitor the other code sections for proper operation. This can be done by checking variables to make sure they are within reasonable values. Simply using a software timer to reset the WDT every 15 milliseconds will not provide much protection against minor problems.

It is also recommended that unused areas of code be filled with NOPs and periodic jumps to an error routine or RST (reset chip) instructions. This is particularly important in the code around lookup tables, since if lookup tables are executed undesired results will occur. Wherever space allows, each table should be surrounded by 7 NOPs (the longest 8096 instruction has 7 bytes) and a RST or jump to error routine instruction. Since RST is a one-byte instruction, the NOPs are not needed if RSTs are used instead of jumps to an error routine. This will help to ensure a speedy recovery should the processor have a glitch in the program flow. Since RST instruction has an opcode of 0FFH, pulling the data lines high with resistors will cause an RST to be executed if unimplemented memory is addressed.

12.2 Disabling The Watchdog

The watchdog should be disabled by software not initializing it. If this is not possible, such as during program development, the watchdog can be disabled by holding the $\overline{\text{RESET}}$ pin at 2.0V to 2.5V. Voltages over 2.5V on the pin could quickly damage the device. Even at 2.5V, using this technique for other than debugging purposes is not recommended, as it may effect long term reliability. It is further recommended that any device used in this way for more than several seconds, not be used in production versions of products. Section 1.6 of the Hardware Design chapter has more information on disabling the Watchdog Timer.

13.0 RESET

13.1 Reset Signal

As with all processors, the 8X9X must be reset each time the power is turned on. This is done by holding the $\overline{\text{RESET}}$ pin low for at least 2 state times after the power supply is within tolerance and the oscillator has stabilized. (See Figure 44, TRLPV.)

After the $\overline{\text{RESET}}$ pin is brought high, a ten state reset sequence is executed. During this time, the Chip Configuration Byte (CCB) is read from location 2018H and written to the 8X9X Chip Configuration Register (CCR). If the voltage on the $\overline{\text{EA}}$ pin selects the internal/external execution mode the CCB is read from internal ROM/EPROM. If the voltage on the $\overline{\text{EA}}$ pin selects the external execution only mode the CCB is read from external memory.

The 8X9X can be reset using a capacitor, 1-shot, or any other method capable of providing a pulse of at least 2 state times longer than required for V_{CC} and the oscillator to stabilize.

For best functionality, it is suggested that the reset pin be pulled low with an open collector device. In this way, several reset sources can be wire ORed together. Remember, the $\overline{\text{RESET}}$ pin itself can be a reset source when the RST instruction is executed or when the Watchdog Timer overflows. Details of hardware suggestions for reset can be found in Section 1.4 of the Hardware Design chapter.

13.2 Reset Status

The I/O lines and control lines of the 8X9X will be in their reset state within 10 XTAL1 periods after reset is low, with V_{CC} and the oscillator stabilized (See Figure 44, TRLPV). Prior to that time, the status of the I/O lines is indeterminate. After the 10 state time reset sequence, the Special Function Registers will be set as follows:

Register	Reset Value
Port 1	XXXXXXXXB
Port 2	XX0XXXX1B
Port 3	11111111B
Port 4	11111111B
PWM Control	00H
Serial Port (Transmit)	undefined
Serial Port (Receive)	undefined
Baud Rate Register	undefined
Serial Port Control	XXXX0XXXB
Serial Port Status	X00XXXXXB
A/D Command	undefined
A/D Result	undefined
Interrupt Pending	undefined
Interrupt Mask	00000000B
Timer 1	0000H
Timer 2	0000H
Watchdog Timer	0000H
HSI Mode	XXXXXXXXB
HSI Status	undefined
IOS0	00000000B
IOS1	00000000B
IOC0	X0X0X0X0B
IOC1	X0X0XXX1B
HSI FIFO	empty
HSO CAM	empty
HSO SFR	000000B
PSW	0000H
Stack Pointer	undefined
Program Counter	2080H

Figure 42. Register Reset Status

Port 1 and Port 2.6, 2.7 reset to a strong or weak pull-up condition. HSO.4 and HSO.5 reset to a floating condition as they are disabled by IOC1.4 and IOC1.6.

Other conditions following a reset are:

Pin	Reset Value
$\overline{\text{RD}}$	high
$\overline{\text{WR}}$/$\overline{\text{WRL}}$	high
ALE/$\overline{\text{ADV}}$	high
$\overline{\text{BHE}}$/$\overline{\text{WRH}}$	high
INST	low
HSO Lines	XX0000B

Figure 43. Bus Control Pins Reset Status

It is important to note that the Stack Pointer and Interrupt Pending Register are undefined, and need to be initialized in software. The Interrupts are disabled by both the mask register and PSW.9 after a reset.

13.3 Reset Sync Mode

The $\overline{\text{RESET}}$ line can be used to start the 8X9X at an exact state time to provide for synchronization of test equipment and multiple chip systems. $\overline{\text{RESET}}$ is active low. To synchronize devices, $\overline{\text{RESET}}$ is brought high on the rising edge of XTAL1. Complete details on synchronizing devices can be found in Section 1.5 of the Hardware Design chapter.

It is very possible that devices which start in sync may not stay that way. The best example of this would be when a "jump on I/O bit" is being used to hold the processor in a loop. If the line changes during the time it is being tested, one processor may see it as a one, while the other sees it as a zero. The result is that one processor will do an extra loop, thus putting it several states out of sync with the other.

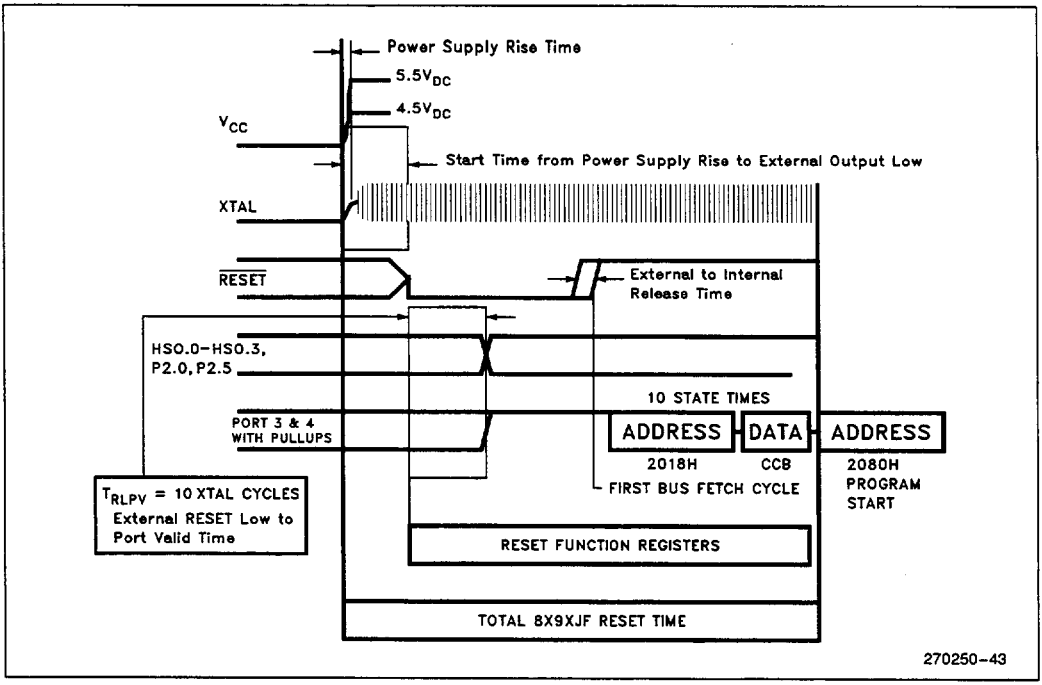


Figure 44. TRLPV

8X9X Hardware Design Information

2

intel®

November 1990

8X9X HARDWARE DESIGN INFORMATION

Order Number: 270246-004

8X9X HARDWARE DESIGN INFORMATION

CONTENTS

CONTENTS

8X9X HARDWARE DESIGN INFORMATION

OVERVIEW

This chapter of the manual is devoted to the hardware engineer. All of the information you need to connect the correct pin to the correct external circuit is provided. Many of the special function pins have different characteristics which are under software control. Therefore, it is necessary to define the system completely before the hardware is wired-up.

Frequently within this chapter a specification for a current, voltage, or time period is referred to; the values provided are to be used as an approximation only. The exact specification can be found in the latest data sheet for the particular device and temperature range that is being used.

This chapter is written about 8X9XBH, 8X9XJF, and 8X98 devices. These devices are generically referred to as the 8X9X. All information in this chapter refers to the 8X9XBH, the 8X9XJF, and the 8X98 unless otherwise noted.

1.0 REQUIRED HARDWARE CONNECTIONS

Although the 8X9X is a single-chip microcontroller, it still requires several external connections to make it work. Power must be applied, a clock source provided, and some form of reset circuitry must be present. We will look at each of these areas of circuitry separately. Figure 6 shows the connections that are needed for a single-chip system.

1.1 Power Supply Information

Power for the 8X9X flows through six pins. They are: three positive voltage pins—V_{CC} (digital), V_{REF} (Port 0 digital I/O and A/D power), V_{PD} (power down mode), and three common returns—two V_{SS} pins and one ANGND pin. All six of these pins **must** be connected on the 8X9X for normal operation. The V_{CC} pin, V_{REF} pin and V_{PD} pin should be tied to 5 volts. The two V_{SS} pins and the ANGND pin must be grounded. When the analog to digital converter is being used it may be desirable to connect the VREF pin to a separate power supply, or at least a separate power supply line.

The three common return pins should be connected at the chip with as short a lead as possible to avoid problems due to voltage drops across the wiring. There should be no measurable voltage difference between V_{SS1} and V_{SS2}. The two V_{SS} pins and the ANGND pin must all be nominally at 0 volts. The maximum current drain of the 8X9X is around 180 mA, with all lines unloaded.

When the analog converter is being used, clean, stable power must be provided to the analog section of the chip to assure highest accuracy. To achieve this, it may be desirable to separate the analog power supply from the digital power supply. The V_{REF} pin supplies the digital circuitry in the A/D converter and provides the 5 volt reference to the analog portion of the converter. V_{REF} and ANGND must be connected even if the A/D converter is not used. More information on the analog power supply is in Section 3.1.

1.2 Other Needed Connections

Several other connections are needed to configure the 8X9X. In normal operation the following pins should be connected to the indicated power supply.

Pin	Power Supply
NMI	V_{CC}
$\overline{EA}$	V_{CC} (to allow internal execution) V_{SS} (to force external execution)

Although the $\overline{EA}$ pin has an internal pulldown, it is best to tie this pin to the desired level. This will prevent induced noise from disturbing the system. Raising $\overline{EA}$ to +12.75 volts will place an 8X9X in a special operating mode designed for programming and program memory verification (see Section 10).

1.3 Oscillator Information

The 8X9X requires a clock source to operate. This clock is provided to the chip through the XTAL1 input. The frequency of operation is from 6 MHz to 12 MHz.

The on-chip circuitry for the 8X9X oscillator is a single stage linear inverter as shown in Figure 1. It is intended for use as a crystal-controlled, positive reactance oscillator with external connections as shown in Figure 2. In this application, the crystal is being operated in its fundamental response mode as an inductive reac-

tance in parallel resonance with shunt capacitance external to the crystal.

The crystal specifications and capacitance values (C1 and C2 in Figure 2) are not critical. Thirty picofarads can be used in these positions at any frequency with good quality crystals. For 0.5% frequency accuracy, the crystal frequency can be specified at series resonance or for parallel resonance with any load capacitance. (In other words, for that degree of frequency accuracy, the load capacitance simply doesn't matter.) For 0.05% frequency accuracy the crystal frequency should be specified for parallel resonance with 25 pF load capacitance, if C1 and C2 are 30 pF.

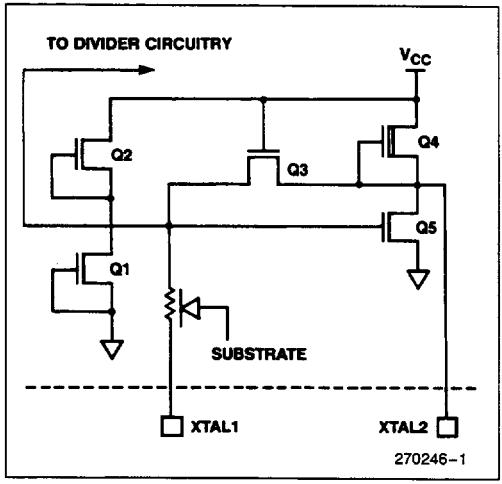


Figure 1. 8X9X Oscillator Circuit

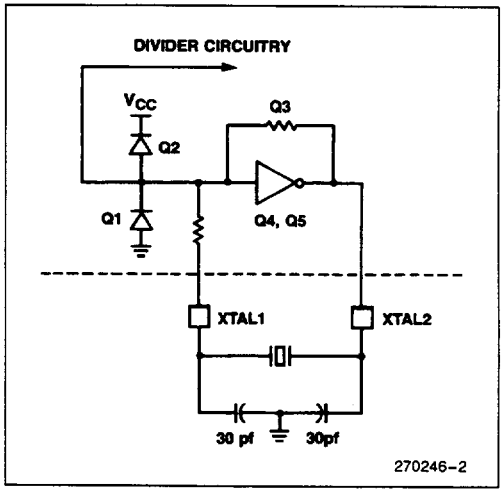


Figure 2. Crystal Oscillator Circuit

An external oscillator may encounter as much as a 100 pF load at XTAL1 when it starts up. This is due to interaction between the amplifier and its feedback capacitance. Once the external signal meets the V_{IL} and V_{IH} specifications the capacitance will not exceed 20 pF.

A more in-depth discussion of crystal specifications and the selection of values for C1 and C2 can be found in the Intel Application Note, AP-155, "Oscillators for Microcontrollers."

To drive the 8X9X with an external clock source, apply the external clock signal to XTAL1 and let XTAL2 float. An example of this circuit is shown in Figure 3. The required voltage levels on XTAL1 are specified in the data sheet. The signal on XTAL1 must be clean with good solid levels.

It is important that the minimum high and low times are met to avoid having the XTAL1 pin in the transition range for long periods of time. The longer the signal is in the transition region, the higher the probability that an external noise glitch could be seen by the clock generator circuitry. Noise glitches on the 8X9X internal clock lines will cause unreliable operation.

The clock generator provides a 3 phase clock output from the XTAL1 pin input. Figure 4 shows the waveforms of the major internal timing signals.

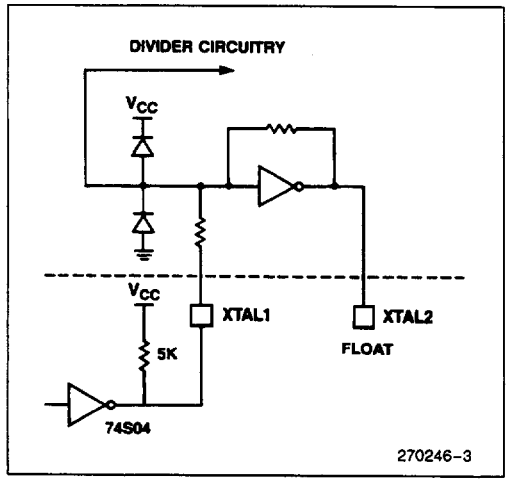


Figure 3. External Clock Drive

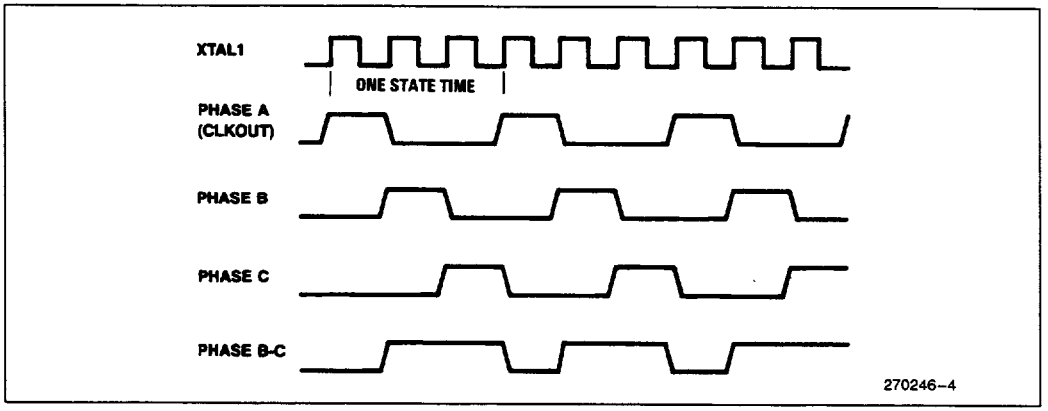


Figure 4. Internal Timings

1.4 Reset Information

In order for the 8X9X to function properly it must be reset. This is done by holding the $\overline{\text{RESET}}$ pin low for at least 10 XTAL1 cycles after the power supply is within tolerance and the oscillator has stabilized.

After the $\overline{\text{RESET}}$ pin is brought high, a ten state reset sequence is executed. During this time, the Chip Configuration Byte (CCB) is read from location 2018H and written to the 8X9X Chip Configuration Register (CCR). If the voltage on the $\overline{\text{EA}}$ pin selects the internal/external execution mode the CCB is read from internal ROM/EPROM. If the voltage on the $\overline{\text{EA}}$ pin selects the external execution only mode the CCB is read from external memory. See Figure 5, and 5A.

There are several ways to provide a good reset to an 8X9X, the simplest being just to connect a capacitor from the reset pin to ground. The capacitor should be on the order of 2 microfarads for every millisecond of reset time required. This method will only work if the rise time of V_{CC} is fast and the total reset time is less than around 50 milliseconds. It also may not work if the $\overline{\text{RESET}}$ pin is to be used to reset other devices on the board. An 8X9X with the minimum required connections is shown in Figure 6.

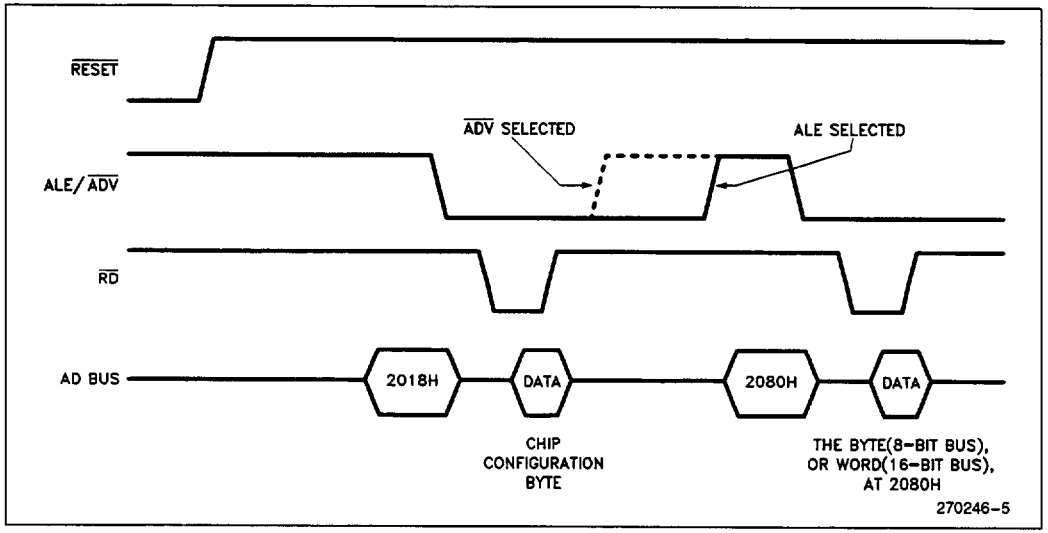


Figure 5. Reset Sequence

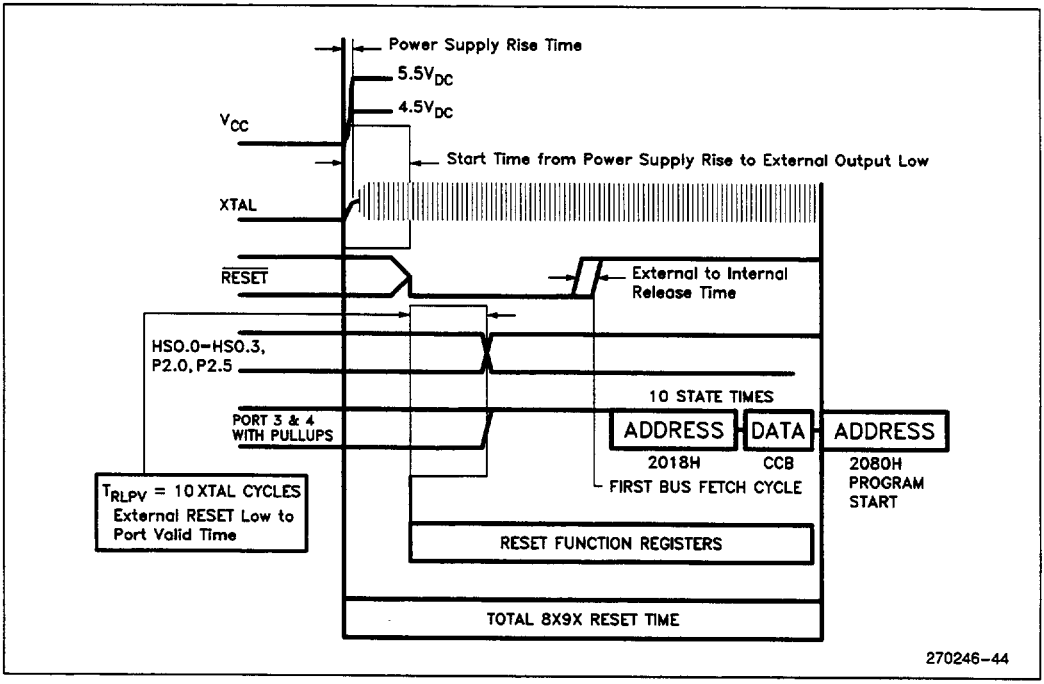
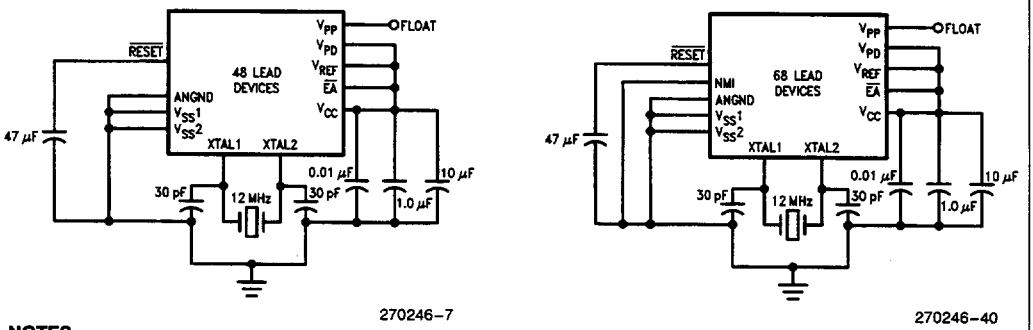


Figure 5A. T_{RLPV}

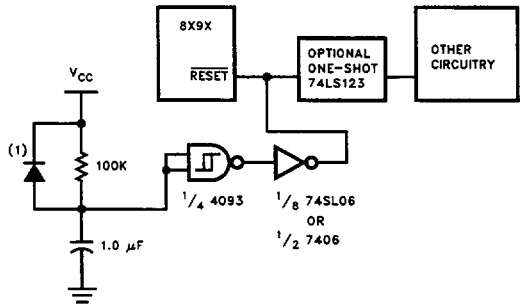


NOTES:
1. These capacitors are needed only if A to D is used.
2. V_{REF} & ANGND may be connected to the same traces as the digital power supply if the A to D is not used.

Figure 6. Minimum Hardware Connections

The 8X9X $\overline{\text{RESET}}$ pin can be used to allow other chips on the board to make use of the Watchdog Timer or the RST instruction. When this is done the reset hardware should be a one-shot with an open collector output. The reset pulse going to the other devices may have to be buffered and lengthened with a one-shot, since the $\overline{\text{RESET}}$ low duration is only one state. If this is done, it is possible that the 8X9X will be reset and start running before the other devices on the board are out of reset. The software must account for this possible problem.

A capacitor directly connected to $\overline{\text{RESET}}$ cannot be used to reset the device if the pin is to be used as an output. If a large capacitor is used, the pin will pull-down more slowly than normal. It will continue to pull-down until the 8X9X is reset. It could fall so slowly that it never goes below the internal switch point of the reset signal (1 to 1.5 volts), a voltage which may be above the guaranteed switch point of external circuitry connected to the pin. A circuit example is shown in Figure 7.



NOTE:
1. The diode will provide a faster cycle time repetitive power-on-resets.

Figure 7. Multiple Chip Reset Circuit

1.5 Sync Mode

If $\overline{\text{RESET}}$ is brought high at the same time as or just after the rising edge of XTAL1, the device will start executing the 10 state time RST instruction exactly 6½ XTAL1 cycles later. This feature can be used to synchronize several MCS-96 devices. A diagram of a typical connection is shown in Figure 8. It should be noted that devices that start in sync may not stay that way, due to propagation delays which may cause the synchronized devices to receive signals at slightly different times.

1.6 Disabling the Watchdog Timer

The Watchdog Timer will pull the $\overline{\text{RESET}}$ pin low when it overflows. See Figure 9. If the pin is being externally held above the low going threshold, the pulldown transistor will remain on indefinitely. This means that once the watchdog overflows, the device must be reset or $\overline{\text{RESET}}$ must be held high indefinitely. Just resetting the Watchdog Timer in software will not clear the flip-flop which keeps the $\overline{\text{RESET}}$ pulldown on.

The pulldown is capable of sinking on the order of 30 milliamps if it is held at 2.0 volts. This amount of current may cause some long term reliability problems due to localized chip heating. For this reason, devices that will be used in production should never have had the Watchdog Timer over-ridden for more than a second or two.

Whenever the reset pin is being pulled high while the pulldown is on, it should be through a resistor that will limit the voltage on $\overline{\text{RESET}}$ to 2.5 volts and the current through the pin to 40 milliamps.

If it is necessary to disable the Watchdog Timer for more than a brief test the software solution of never initiating the timer should be used. See Section 14 in the Architecture Chapter.

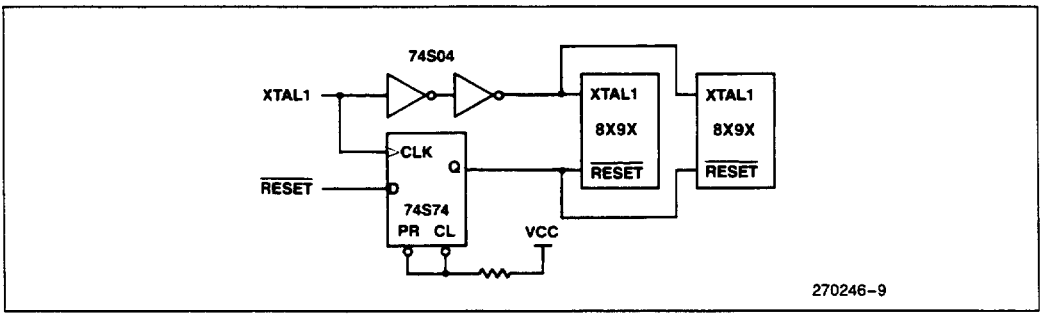


Figure 8. Reset Sync Mode

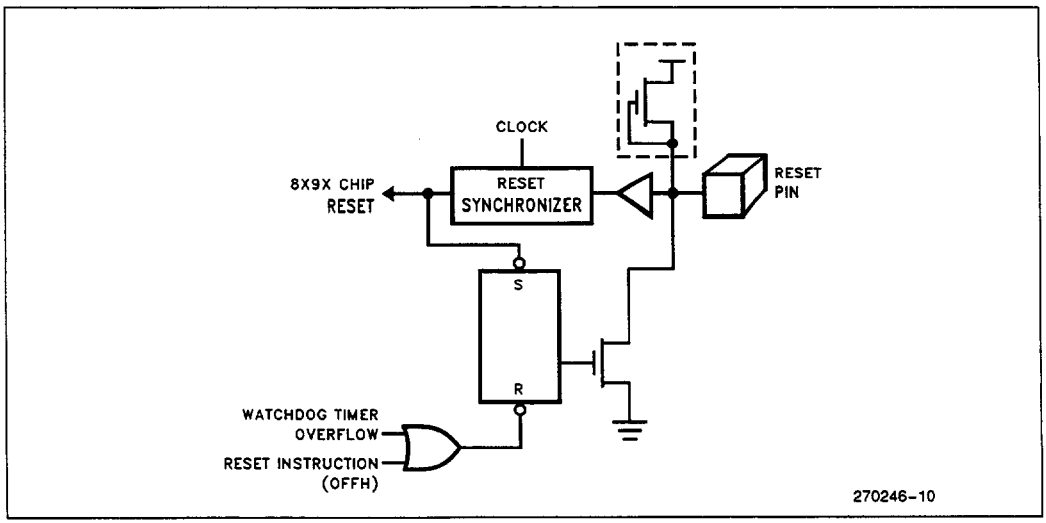


Figure 9. Reset Logic

1.7 Power Down Circuitry

Battery backup can be provided on the 8X9X with a 1 mA current drain at 5 volts. This mode will hold locations 0F0H through 0FFH valid as long as the power to the V_{PD} pin remains on. The required timings to put the device into power-down and an overview of this mode are given in Section 2.3 in the 8X9X Architecture Chapter.

A 'key' can be written into power-down RAM while the device is running. This key can be checked on reset to determine if it is a start-up from power-down or a complete cold start. In this way the validity of the power-down RAM can be verified. The length of this key determines the probability that this procedure will work, however, there is always a statistical chance that the RAM will power up with a replica of the key.

Under most circumstances, the power-fail indicator which is used to initiate a power-down condition must come from the unfiltered, unregulated section of the power supply. The power supply must have sufficient storage capacity to operate the 8X9X until it has completed its reset operation.

2.0 DRIVE AND INTERFACE LEVELS

There are five types of I/O lines on the 8X9X. Of these, two are inputs and three are outputs. All of the pins of the same type have the same current/voltage characteristics. Some of the control input pins, such as XTAL1 and $\overline{\text{RESET}}$, may have slightly different characteristics. These pins are discussed in Section 1.

While discussing the characteristics of the I/O pins some approximate current or voltage specifications will be given. The exact specifications are available in the lastest version of the data sheet that corresponds to the device being used.

2.1 Quasi-Bidirectional Ports

The Quasi-Bidirectional pins of Port 1, Port 2.6, and Port 2.7 have both input and output port configurations. They have three distinct states; low impedance current sink (Q2), low impedance current source (Q1), and high impedance current source (Q3). As a low impedance current sink, the pin has specification of sinking up to around 0.5 mA, while staying below 0.45 volts. The pin is placed in this condition by writing a '0' to the SFR (Special Function Register) controlling the pin.

Examine Figure 10. When the SFR contains a '0' and a '1' is written to it, Q1 (a low impedance MOSFET pullup) is turned on for one state, then it is turned off and the depletion pullup holds the line at a logical '1' state. The low-impedance pullup is used to shorten the rise time of the pin, and has current source capability on the order of 100 times that of the depletion pullup.

While the depletion mode pullup is the only device on, the pin may be used as an input with a leakage of around 100 microamps from 0.45 volts to V_{CC}. It is ideal for use with TTL or CMOS chips and may even be used directly with switches. However if the switch option is used, certain precautions should be taken. It is important to note that any time the pin is read, the value returned will be the value on the pin, not the value placed in the control register. This could cause logical operations made directly on these pins to indavertently write a 0 to pins being used as inputs. In order to perform logical operations on a port where a quasi-bidirectional pin is an input, it is necessary to guarantee that the bit associated with the input pin is always a one when writing to the port.

2.2 Quasi-Bidirectional Hardware Connections

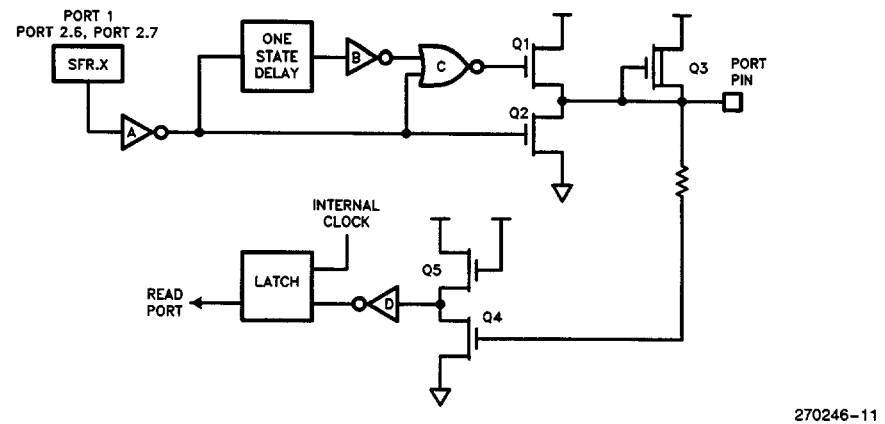
When using the quasi-bidirectional ports as inputs tied to switches, series resistors may be needed if the ports will be written to internally after the device is initialized. The amount of current sourced to ground from each pin is tyically 20 mA or more. Therefore, if all 8 pins are tied to ground, 160 mA will be sourced. This is equivalent to instantaneously doubling the power used by the chip and may cause noise in some applications.

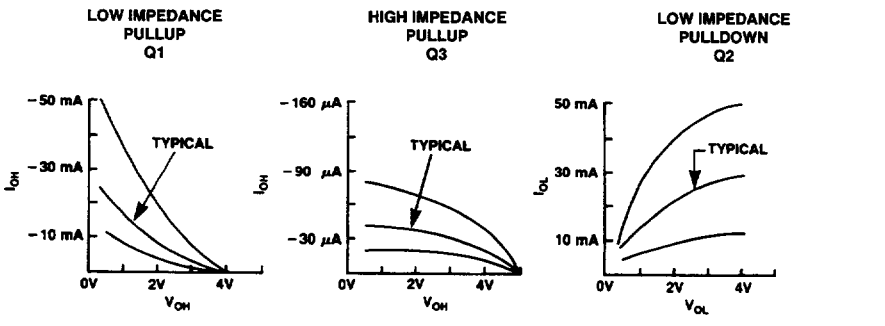
This potential problem can be solved in hardware or software. In software, never write a zero to a pin being used as an input.

In hardware, a 1K resistor in series with each pin will limit current to a reasonable value without impeding the ability to override the high impedance pullup. If all 8 pins are tied together a 120Ω resistor would be reasonable. The problem is not quite as severe when the inputs are tied to electronic devices instead of switches, as most external pulldowns will not hold 20 mA to 0.0 volts.

Writing to a Quasi-Bidirectional Port with electronic devices attached to the pins requires special attention. Consider using P1.0 as an input and trying to toggle P1.1 as an output:

```
ORB   IOPORT1, #00000001B ; Set P1.0
                          ; for input
XORB  IOPORT1, #00000010B ; Complement
                          ; P1.1
```





NOTE:
These graphs show typical pin capabilities, they are not guaranteed specifications

Figure 10. Quasi-Bidirectional Port

The first instruction will work as expected but two problems can occur when the second instruction executes. The first is that even though P1.1 is being driven high by the 8X9X it is possible that it is being held low externally. This typically happens when the port pin is used to drive the base of an NPN transistor which in turn drives whatever there is in the outside world which needs to be toggled. The base of the transistor will clamp the port pin to the transistor's Vbe above ground, typically 0.7V. The 8X9X will input this value as a zero even if a one has been written to the port pin. When this happens the XORB instruction will always write a one to the port pin's SFR and the pin will not toggle.

The second problem, which is related to the first, is that if P1.0 happens to be driven to a zero when Port 1 is read by the XORB instruction, then the XORB will write a zero to P1.0 and it will no longer be useable as an input.

The first situation can best be solved by the external driver design. A series resistor between the port pin and the base of the transistor often works by bringing up the voltage present on the port pin. The second case can be taken care of in the software fairly easily:

```
LDB  AL, IOPORT1
XORB AL, #010B
ORB  AL, #001B
STB  AL, IOPORT1
```

A software solution to both cases is to keep a byte in RAM as an image of the data to be output to the port; any time the software wants to modify the data on the port it can then modify the image byte and copy it to the port.

If a switch is used on a long line connected to a quasi-bidirectional pin, a pullup resistor is recommended to reduce the possibility of noise glitches and to decrease

the rise time of the line. On extremely long lines that are handling slow signals, a capacitor may be helpful in addition to the resistor to reduce noise.

2.3 Input Only Ports

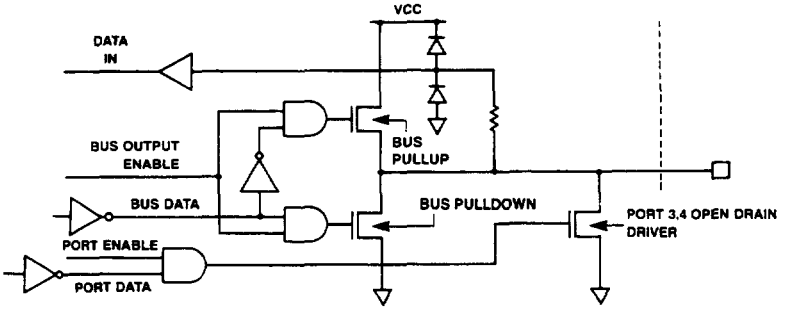
The high impedance input pins on the 8X9X have an input leakage of a few microamps and are predominantly capacitive loads on the order of 10 pF.

Port 0 pins are special in that they may individually be used as digital inputs, or as analog inputs. A Port 0 pin being used as a digital input acts as the high impedance input ports just described. However, Port 0 pins being used as analog inputs are required to provide current to the internal sample capacitor when a conversion begins. This means that the input characteristics of a pin will change if a conversion is being done on that pin. See Section 3. In either case, if Port 0 is to be used as analog or digital I/O, it will be necessary to provide power to this port through the V_{REF} pin.

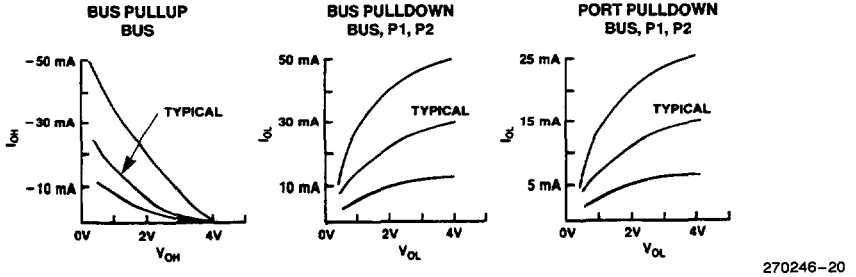
2.4 Open Drain Ports

Ports 3 and 4 on the 8X9X are open drain ports. There is no pullup when these pins are used as I/O ports. These pins have different characteristics when used as bus pins as described in the next section. A diagram of the output buffers connected to Ports 3 and 4 and the bus pins is shown in Figure 11.

When Ports 3 and 4 are to be used as inputs, or as bus pins, they must first be written with a '1'. This will put the ports in a high impedance mode. When they are used as outputs, a pullup resistor must be used externally. The sink capability of these pins is on the order of 0.8 milliamps so the total pullup current to the pin must be less than this. A 15K pullup resistor will source a maximum of 0.33 milliamps, so it would be a reasonable value to choose if no other circuits with pullups were connected to the pin.



270246-19



270246-20

NOTE:
These graphs show typical pin capabilities, they are not guaranteed specifications.

Figure 11. Bus and Port 3 and 4 Pins

2.5 HSO Pins, Control Outputs and Bus Pins

The control outputs and HSO pins have output buffers with the same output characteristics as those of the bus pins. Included in the category of control outputs are: TXD, RXD (in Mode 0), PWM, CLKOUT, ALE, $\overline{BHE}$, $\overline{RD}$, and $\overline{WR}$. The bus pins have 3 states: output high, output low, and high impedance input. As a high output, the pins are specified to source around 200 μA to 2.4 volts, but the pins can source on the order of ten times that value in order to provide the fast rise times. When used as a low output, the pins can sink around 2 mA at 0.45 volts, and considerably more as the voltage increases. When in the high impedance state, the pin acts as a capacitive load with a few microamps of leakage. Figure 11 shows the internal configuration of a bus pin.

3.0 ANALOG INPUTS

The on-chip A/D converter of the 8X9X can be used to digitize analog inputs while analog outputs can be generated with either the chip's PWM output or HSO unit. This section describes the analog input suggestions. See Section 4 for analog output.

The 8X9X's Integrated A/D converter includes an eight channel analog multiplexer, sample-and-hold circuit and 10-bit analog to digital converter (Figure 12). The 8X9X can therefore select one of eight analog inputs to convert, sample-and-hold the input voltage and convert the voltage into a digital value. Each conversion takes 22 microseconds, including the time required for the sample-hold (with XTAL1 = 12 MHz). The method of conversion is successive approximation.

Section 3.5 contains the definitions of numerous terms used in connection with the A/D converter.

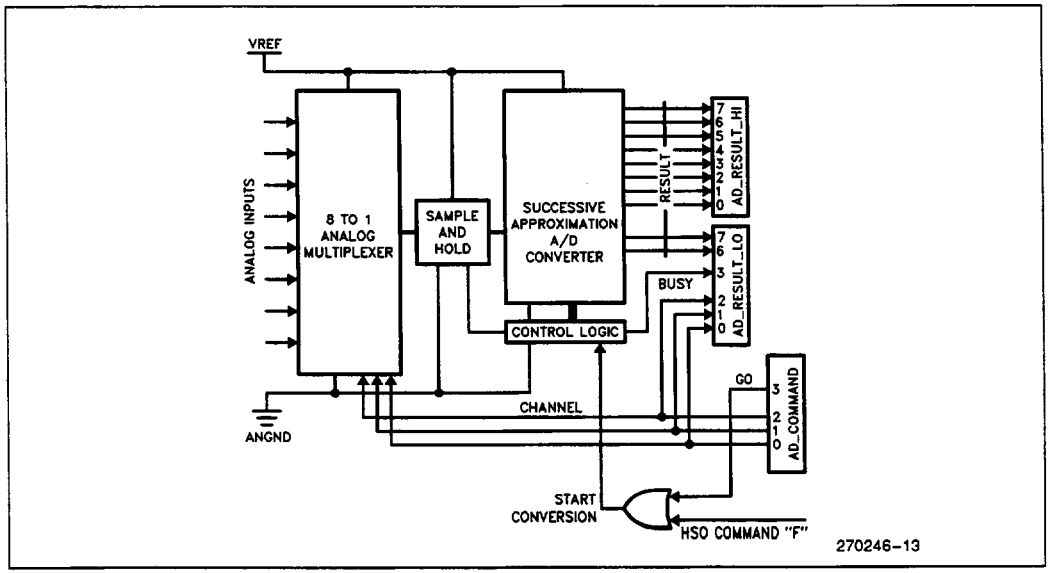


Figure 12. A/D Converter Block Diagram

3.1 A/D Overview

The conversion process is initiated by the execution of HSO command 0FH, or by writing a one to the GO Bit in the A/D Control Register. Either activity causes a start conversion signal to be sent to the A/D converter control logic. If an HSO command was used, the conversion process will begin when Timer 1 increments. This aids applications attempting to approach spectrally pure sampling, since successive samples spaced by equal Timer 1 delays will occur with a variance of about ±50 ns (assuming a stable clock on XTAL1). However, conversions initiated by writing a one to the ADCON register GO Bit will start within three state times after the instruction has completed execution resulting in a variance of about 0.75 μs (XTAL1 = 12 MHz).

Once the A/D unit receives a start conversion signal, there is a one state time delay before sampling (sample delay) while the successive approximation register is reset and the proper multiplexer channel is selected. After the sample delay, the multiplexer output is connected to the sample capacitor and remains connected for four state times (sample time). After this four state time "sample window" closes, the input to the sample capacitor is disconnected from the multiplexer so that changes on the input pin will not alter the stored charge while the conversion is in progress. The comparator is then auto-zeroed and the conversion begins. The sample delay and sample time uncertainties are each approximately ±50 ns, independent of clock speed.

To perform the actual analog-to-digital conversion the 8X9X implements a successive approximation algorithm. The converter hardware consists of a 256-resistor ladder, a comparator, coupling capacitors and a 10-bit successive approximation register (SAR) with logic that guides the process. The resistor ladder provides 20 mV steps (V_{REF} = 5.12V), while capacitive coupling is used to create 5 mV steps within the 20 mV ladder voltages. Therefore, 1024 internal reference voltages are available for comparison against the analog input to generate a 10-bit conversion result.

A successive approximation conversion is performed by comparing a sequence of reference voltages, to the analog input, in a binary search for the reference voltage that most closely matches the input. The ½ full scale reference voltage is the first tested. This corresponds to a 10-bit result where the most significant bit is zero, and all other bits are ones (0111.1111.11b). If the analog input was less than the test voltage, bit 10 of the SAR is left a zero, and a new test voltage of ¼ full scale (0011.1111.11b) is tried. If this test voltage was lower than the analog input, bit 9 of the SAR is set and bit 8 is cleared for the next test (0101.1111.11b). This binary search continues until 10 tests have occurred, at which time the valid 10-bit conversion result resides in the SAR where it can be read by software.

The total number of state times required is 88 for a 10-bit conversion. Attempting to short-cycle the 10-bit conversion process by reading A/D results before the done bit is set is not recommended.

3.2 A/D Interface Suggestions

The external interface circuitry to an analog input is highly dependent upon the application, and can impact converter characteristics. In the external circuit's design, important factors such as input pin leakage, sample capacitor size and multiplexer series resistance from the input pin to the sample capacitor must be considered.

For the 8X9X, these factors are idealized in Figure 13. The external input circuit must be able to charge a sample capacitor (C_S) through a series resistance (R_I) to an accurate voltage given a D.C. leakage (I_L). On the 8X9X, C_S is around 2 pF, R_I is around 5 KΩ and I_L is specified as 3 μA maximum. In determining the necessary source impedance R_S, the value of V_{BIAS} is not important.

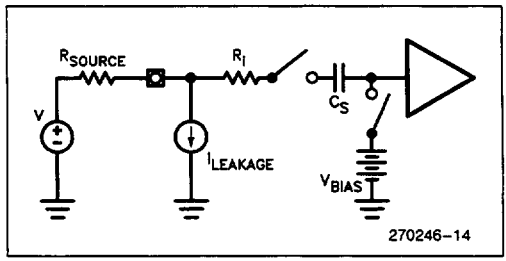


Figure 13. Idealized A/D Sampling Circuitry

External circuits with source impedances of 1 KΩ or less will be able to maintain an input voltage within a tolerance of about ±0.61 LSB (1.0 KΩ × 3.0 μA = 3.0 mV) given the D.C. leakage. Source impedances above 2 KΩ can result in an external error of at least one LSB due to the voltage drop caused by the 1 μA leakage. In addition, source impedances above 25 KΩ may degrade converter accuracy as a result of the internal sample capacitor not being fully charged during the 1 μs (12 MHz clock) sample window.

If large source impedances degrade converter accuracy because the sample capacitor is not charged during the sample time, an external capacitor connected to the pin will compensate for this degradation. Since the sample capacitor is 2 pF, a 0.005 μF capacitor will charge the sample capacitor to an accurate input voltage of ±0.5 LSB (2048 × 2 pF). An external capacitor does not compensate for the voltage drop across the source resistance, but charges the sample capacitor fully during the sample time.

Placing an external capacitor on each analog input will also reduce the sensitivity to noise, as the capacitor combines with series resistance in the external circuit to form a low-pass filter. In practice, one should include a small series resistance prior to the external capacitor on the analog input pin and choose the largest capacitor value practical, given the frequency of the signal being converted. This provides a low-pass filter on the input, while the resistor will also limit input current during over-voltage conditions.

Figure 14 shows a simple analog interface circuit based upon the discussion above. The circuit in the figure also provides limited protection against over-voltage conditions on the analog input. Should the input voltage inappropriately drop significantly below ground, diode D2 will forward bias at about 0.8 VDC. Since the specification of the pin has an absolute maximum low voltage of −0.3V, this will leave about 0.5V across the 270Ω transistor, or about 2 mA of current. This should limit the current to a safe amount. *However, before any circuit is used in an actual application, it should be thoroughly analyzed for applicability to the specific problem at hand.*

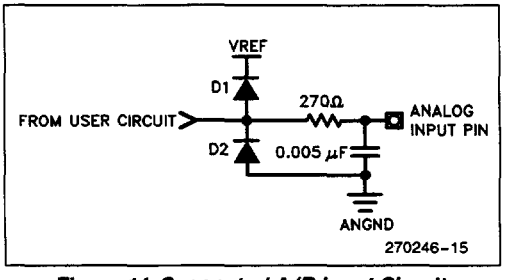


Figure 14. Suggested A/D Input Circuit

3.3 Analog References

Reference supply levels strongly influence the absolute accuracy of the conversion. For this reason, it is recommended that the ANGND pin be tied to the two V_{SS} pins as close to the chip as possible with minimum trace length. Bypass capacitors should also be used between V_{REF} and ANGND. ANGND should be within about a tenth of a volt V_{SS}. V_{REF} should be well regulated and used only for the A/D converter. The V_{REF} supply can be between 4.5V and 5.5V and needs to be able to source around 5 mA. Figure 6 shows all of these connections.

Note that if only ratiometric information is desired, V_{REF} can be connected to V. In addition, V_{REF} and ANGND must be connected even if the A/D converter is not being used. Remember that Port 0 receives its power from the V_{REF} and ANGND pins even when it is used as digital I/O.

3.4 The A/D Transfer Function

The conversion result is a 10-bit ratiometric representation of the input voltage, so the numerical value obtained from the conversion will be:

$$\text{INT}\,[1023 \times (V_{IN} - \text{ANGND})/(V_{REF} - \text{ANGND})].$$

This produces a stair-stepped transfer function when the output code is plotted versus input voltage (see Figure 15). The resulting digital codes can be taken as simple ratiometric information, or they can be used to provide information about absolute voltages or relative voltage changes on the inputs. The more demanding the application is on the A/D converter, the more important it is to fully understand the converter's operation. For simple applications, knowing the absolute error of the converter is sufficient. However, closing a servo-loop with analog inputs necessitates a detailed understanding of an A/D converter's operation and errors.

The errors inherent in an analog-to-digital conversion process are many: quantizing error; zero offset; full-scale error; differential non-linearity; and non-linearity. These are "transfer function" errors related to the A/D converter. In addition, converter temperature drift, V_{CC} rejection, sample-hold feedthrough, multiplexer off-isolation, channel-to-channel matching and random noise should be considered. Fortunately, one "Absolute Error" specification is available which describes the sum total of all deviations between the actual conversion process and an ideal converter. However, the various sub-components of error are important in many applications. These error components are described in Section 3.5 and in the text below where ideal and actual converters are compared.

An unavoidable error simply results from the conversion of a continuous voltage to an integer digital representation. This error is called quantizing error, and is always ±0.5 LSB. Quantizing error is the only error seen in a perfect A/D converter, and is obviously present in actual converters. Figure 15 shows the transfer function for an ideal 3-bit A/D converter (i.e. the Ideal Characteristic).

Note that in Figure 15 the Ideal Characteristic possesses unique qualities: it's first code transition occurs when the input voltage is 0.5 LSB; it's full-scale code transition occurs when the input voltage equals the full-scale reference minus 1.5 LSB; and it's code widths are all exactly one LSB. These qualities result in a digitization without offset, full-scale or linearity errors. In other words, a perfect conversion.

Figure 16 shows an Actual Characteristic of a hypothetical 3-bit converter, which is not perfect. When the Ideal Characteristic is overlaid with the imperfect characteristic, the actual converter is seen to exhibit errors in the location of the first and final code transitions and code widths. The deviation of the first code transition from ideal is called "zero offset", and the deviation of the final code transition from ideal is "full-scale error". The deviation of the code widths from ideal causes two types of errors. Differential Non-Linearity and Non-Linearity. Differential Non-Linearity is a local linearity error measurement, whereas Non-Linearity is an overall linearity error measure.

Differential Non-Linearity is the degree to which actual code widths differ from the ideal one LSB width. Differential Non-Linearity gives the user a measure of how much the input voltage may have changed in order to produce a one count change in the conversion result. Non-Linearity is the worst case deviation of code transitions from the corresponding code transitions of the Ideal Characteristic. Non-Linearity describes how much Differential Non-Linearities could add up to produce an overall maximum departure from a linear characteristic. If the Differential Non-Linearity errors are too large, it is possible for an A/D converter to miss codes or exhibit non-monotonicity. Neither behavior is desireable in a closed-loop system. A converter has no missed codes if there exists for each output code a unique input voltage range that produces that code only. A converter is monotonic if every subsequent code change represents an input voltage change in the same direction.

Differential Non-Linearity and Non-Linearity are quantified by measuring the Terminal Based Linearity Errors. A Terminal Based Characteristic results when an Actual Characteristic is shifted and rotated to eliminate zero offset and full-scale error (see Figure 17). The Terminal Based Characteristic is similar to the Actual Characteristic that would be seen if zero offset and full-scale error were externally trimmed away. In practice, this is done by using input circuits which include gain and offset trimming. In addition, V_{REF} on the 8X9X could also be closely regulated and trimmed within the specified range to affect full-scale error.

Other factors that affect a real A/D Converter system include sensitivity to temperature, failure to completely reject all unwanted signals, multiplexer channel dissimilarities and random noise. Fortunately these effects are small.

Temperature sensitivities are described by the rate at which typical specifications change with a change in temperature.

Undesired signals come from three main sources. First, noise on V_{CC}—V_{CC} Rejection. Second, input signal changes on the channel being converted after the sample window has closed—Feedthrough. Third, signals applied to channels not selected by the multiplexer—Off-Isolation.

Finally, multiplexer on-channel resistances differ slightly from one channel to the next causing Channel-to-Channel Matching errors, and random noise in general results in Repeatability errors.

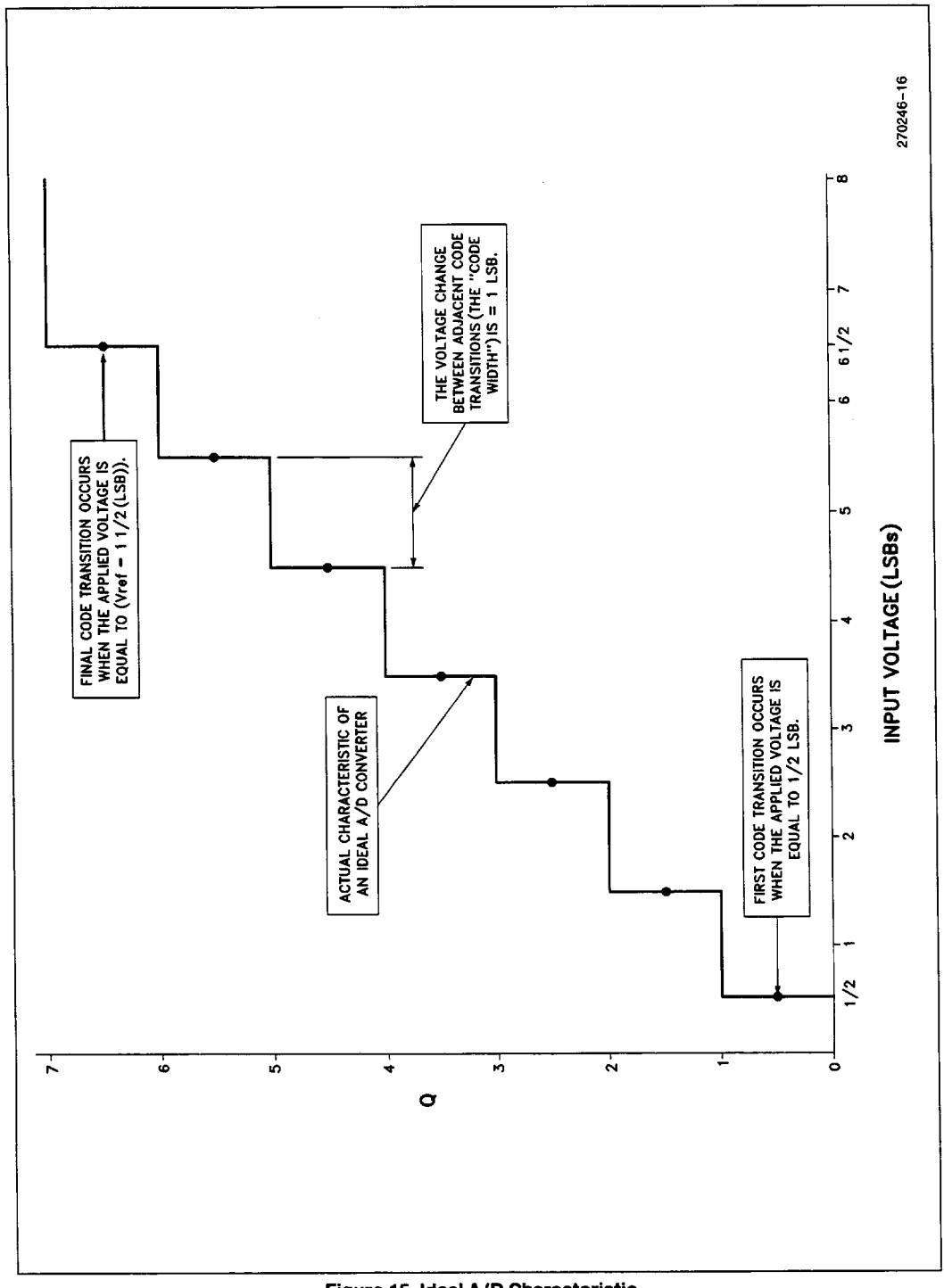


Figure 15. Ideal A/D Characteristic

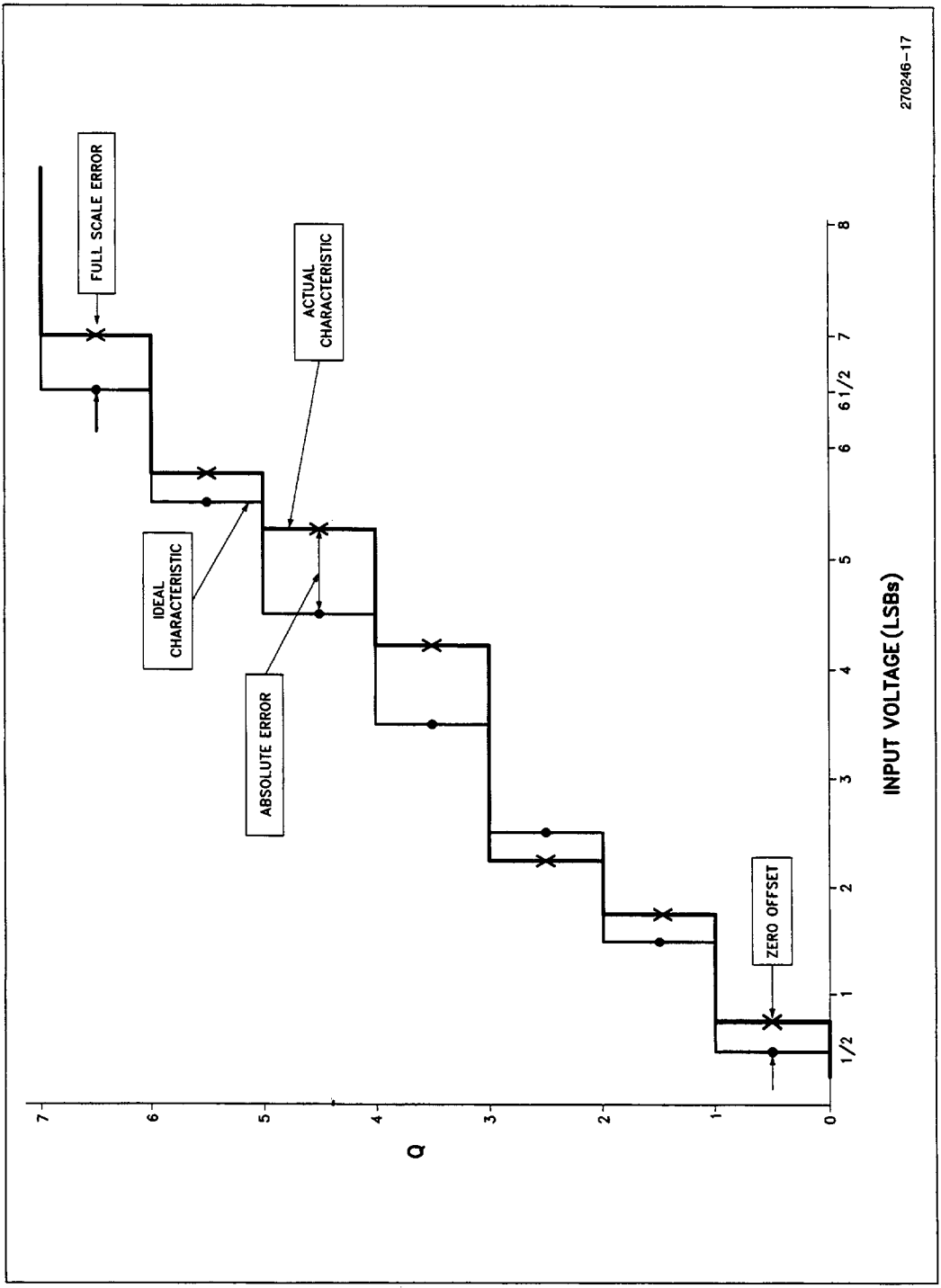


Figure 16. Actual and Ideal Characteristics

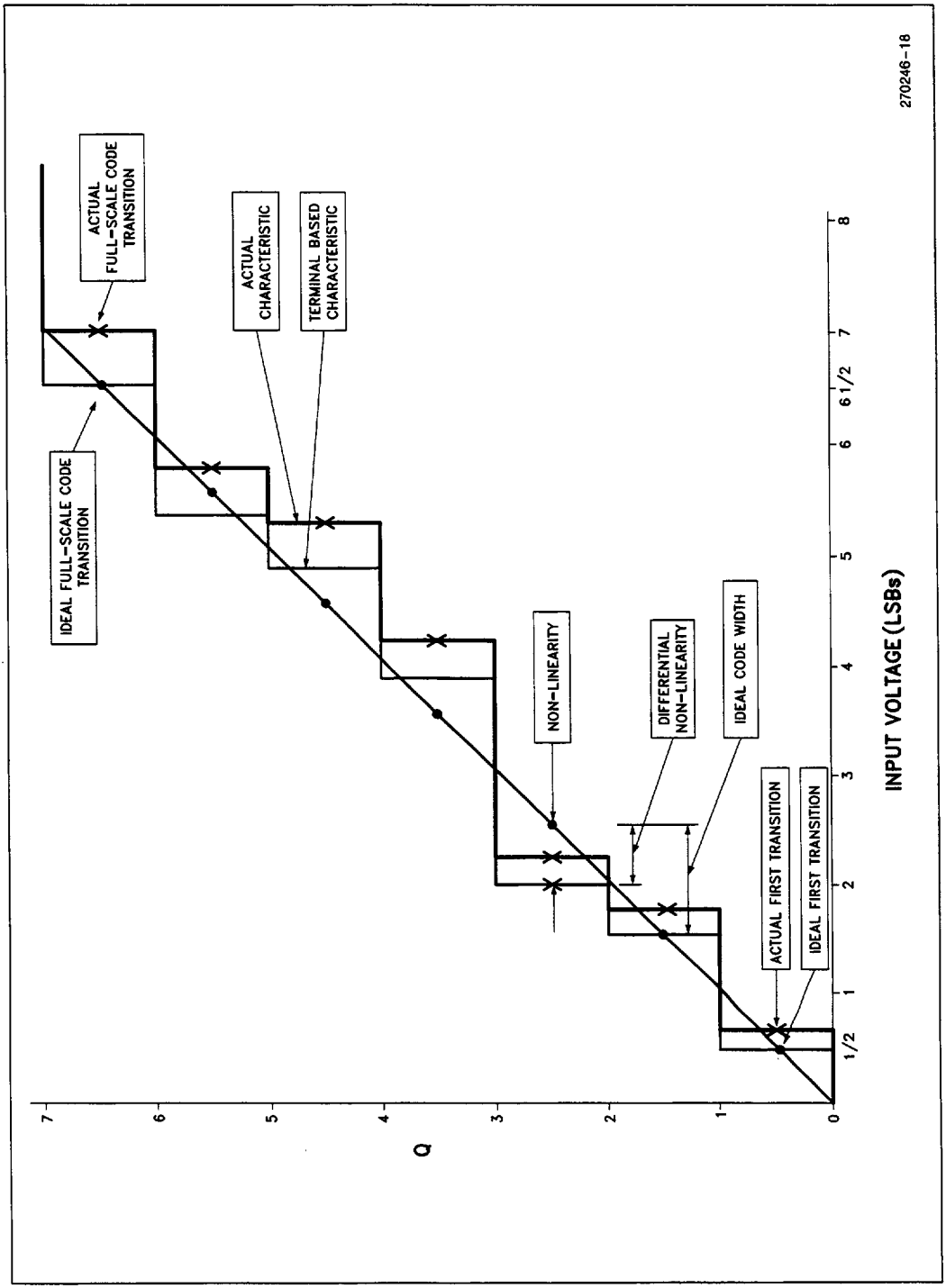


Figure 17. Terminal Based Characteristic

3.5 A/D Glossary of Terms

Figures 15, 16 and 17 display many of these terms.

ABSOLUTE ERROR—The maximum difference between corresponding actual and ideal code transitions. Absolute Error accounts for all deviations of an actual converter from an ideal converter.

ACTUAL CHARACTERISTIC—The characteristic of an actual converter. The characteristic of a given converter may vary over temperature, supply voltage, and frequency conditions. An Actual Characteristic rarely has ideal first and last transition locations or ideal code widths. It may even vary over multiple conversion under the same conditions.

BREAK-BEFORE-MAKE—The property of a multiplexer which guarantees that a previously selected channel will be deselected before a new channel is selected. (e.g. the converter will not short inputs together.)

CHANNEL-TO-CHANNEL MATCHING—The difference between corresponding code transitions of actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

CHARACTERISTIC—A graph of input voltage versus the resultant output code for an A/D converter. It describes the transfer function of the A/D converter.

CODE—The digital value output by the converter.

CODE CENTER—The voltage corresponding to the midpoint between two adjacent code transitions.

CODE TRANSITION—The point at which the converter changes from an output code of Q, to a code of Q + 1. The input voltage corresponding to a code transition is defined to be that voltage which is equally likely to produce either of two adjacent codes.

CODE WIDTH—The voltage corresponding to the difference between two adjacent code transitions.

CROSSTALK—See "Off-Isolation".

D.C. INPUT LEAKAGE—Leakage current to ground from an analog input pin.

DIFFERENTIAL NON-LINEARITY—The difference between the ideal and actual code widths of the terminal based characteristic of a converter.

FEEDTHROUGH—Attenuation of a voltage applied on the selected channel of the A/D converter after the sample window closes.

FULL SCALE ERROR—The difference between the expected and actual input voltage corresponding to the full scale code transition.

IDEAL CHARACTERISTIC—A characteristic with its first code transition at V_{IN} = 0.5 LSB, its last code transition at V_{IN} = (V_{REF} − 1.5 LSB) and all code widths equal to one LSB.

INPUT RESISTANCE—The effective series resistance from the analog input pin to the sample capacitor.

LSB—LEAST SIGNIFICANT BIT: The voltage value corresponding to the full scale voltage divided by 2^n, where n is the number of bits of resolution of the converter. For a 10-bit converter with a reference voltage of 5.12 volts, one LSB is 5.0 mV. Note that this is different than digital LSBs, since an uncertainty of two LSB, when referring to an A/D converter, equals 10 mV. (This has been confused with an uncertainty of two digital bits, which would mean four counts, or 20 mV.)

MONOTONIC—The property of successive approximation converters which guarantees that increasing input voltages produce adjacent codes of increasing value, and that decreasing input voltages produce adjacent codes of decreasing value.

NO MISSED CODES—For each and every output code, there exists a unique input voltage range which produces that code only.

NON-LINEARITY—The maximum deviation of code transitions of the terminal based characteristic from the corresponding code transitions of the ideal characteristics.

OFF-ISOLATION—Attenuation of a voltage applied on a deselected channel of the A/D converter. (Also referred to as Crosstalk.)

REPEATABILITY—The difference between corresponding code transitions from different actual characteristics taken from the same converter on the same channel at the same temperature, voltage and frequency conditions.

RESOLUTION—The number of input voltage levels that the converter can unambiguously distinguish between. Also defines the number of useful bits of information which the converter can return.

SAMPLE DELAY—The delay from receiving the start conversion signal to when the sample window opens.

SAMPLE DELAY UNCERTAINTY—The variation in the Sample Delay.

SAMPLE TIME—The time that the sample window is open.

SAMPLE TIME UNCERTAINTY—The variation in the sample time.

SAMPLE WINDOW—Begins when the sample capacitor is attached to a selected channel and ends when the sample capacitor is disconnected from the selected channel.

SUCCESSIVE APPROXIMATION—An A/D conversion method which uses a binary search to arrive at the best digital representation of an analog input.

TEMPERATURE COEFFICIENTS—Change in the stated variable per degree centigrade temperature change. Temperature coefficients are added to the typical values of a specification to see the effect of temperature drift.

TERMINAL BASED CHARACTERISTIC—An Actual Characteristic which as been rotated and translated to remove zero offset and full-scale error.

VCC REJECTION—Attenuation of noise on the V_{CC} line to the A/D converter.

ZERO OFFSET—The difference between the expected and actual input voltage corresponding to the first code transition.

4.0 ANALOG OUTPUTS

Analog outputs can be generated by two methods, either by using the PWM output or the HSO. Either device will generate a rectangular pulse train that varies in duty cycle and (for the HSO only) period. If a smooth analog signal is desired as an output, the rectangular waveform must be filtered.

In most cases this filtering is best done after the signal is buffered to make it swing from 0 to 5 volts since both of the outputs are guaranteed only to TTL levels. A block diagram of the type of circuit needed is shown in Figure 18. By proper selection of components, accounting for temperature and power supply drift, a highly accurate 8-bit D to A converter can be made using either the HSO or the PWM output. Figure 19 shows two typical circuits. If the HSO is used the accuracy could be theoretically extended to 16-bits, however the temperature and noise related problems would be extremely hard to handle.

When driving some circuits it may be desirable to use unfiltered Pulse Width Modulation. This is particularly true for motor drive circuits. The PWM output can be used to generate these waveforms if a fixed period on the order of 64 μs is acceptable. If this is not the case then the HSO unit can be used. The HSO can generate a variable waveform with a duty cycle variable in up to 65536 steps and a period of up to 131 milliseconds. Both of these outputs produce TTL levels.

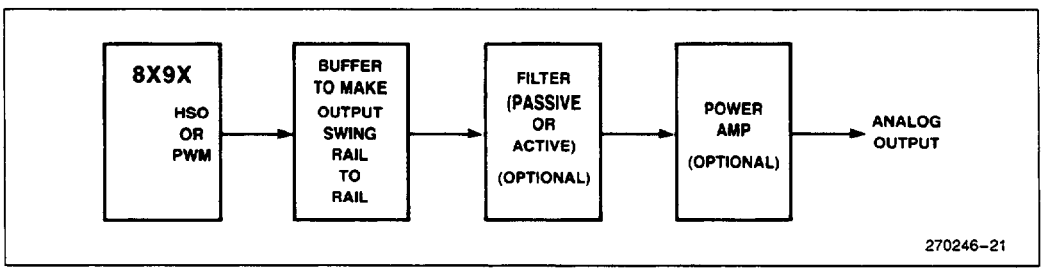


Figure 18. D/A Buffer Block Diagram

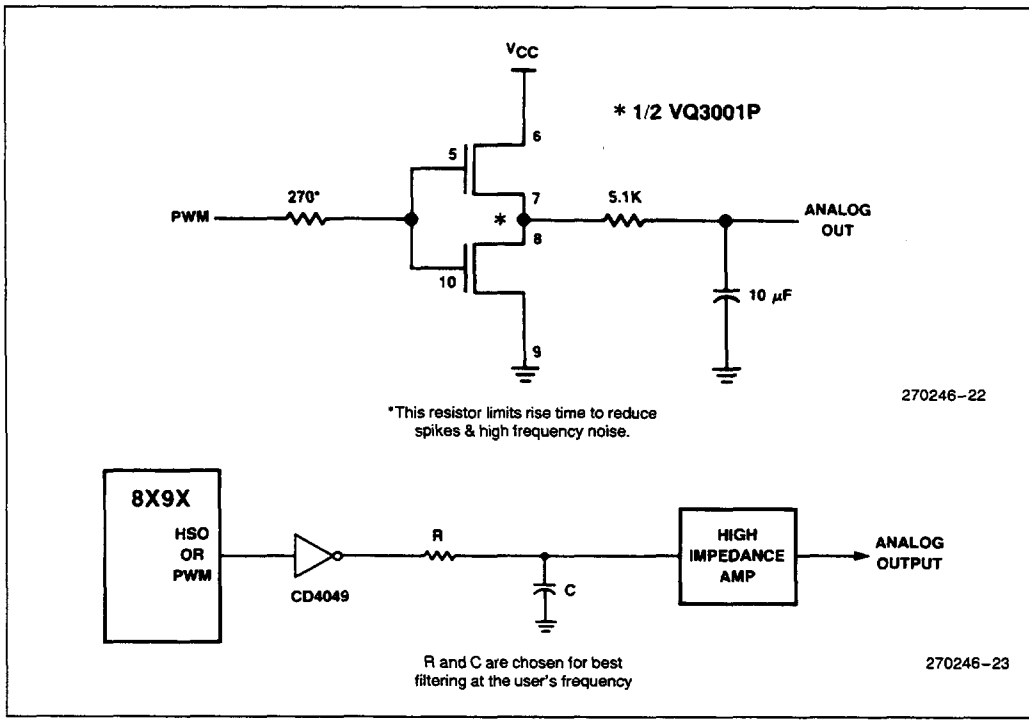


Figure 19. Buffer Circuits for D/A

5.0 I/O TIMINGS

The I/O pins on the 8X9X are sampled and changed at specific times within an instruction cycle. The changes occur relative to the internal phases shown in Figure 4. Note that the delay from XTAL1 to the internal clocks range from about 30 ns to 100 ns over process and temperature. Signals generated by internal phases are further delayed by 5 ns to 15 ns. The timings shown in this section are idealized; no propagation delay factors have been taken into account. Designing a system that depends on an I/O pin to change within a window of less than 50 ns using the information in this section is not recommended.

5.1 HSO Outputs

Changes in the HSO lines are synchronized to Timer 1. All of the external HSO lines due to change at a certain value of a timer will change just pior to the incrementing of Timer 1. This corresponds to an internal change during Phase B every eight state times. From an external perspective the HSO pin should change just prior to the rising edge of CLKOUT and be stable by its falling edge. Information from the HSO can be latched on the CLKOUT falling edge. Internal events can occur anytime during the 8 state time window.

Timer 2 is synchronized to increment no faster than Timer 1, so there will always be at least one incrementing of Timer 1 while Timer 2 is at a specific value.

5.2 HSI Input Sampling

The HSI pins are sampled internally once each state time. Any value on these pins must remain stable for at least 1 full state time to guarantee that it is recognized. The actual sample occurs at the end of Phase A, which, due to propagation delay, is just after the rising edge of CLKOUT. Therefore, if information is to be synchronized to the HSI it should be latched-in on CLKOUT

falling. The time restriction applies even if the divide by eight mode is being used. If two events occur on the same pin within the same 8 state time window, only one of the events will be recorded. If the events occur on different pins they will always be recorded, regardless of the time difference. The 8 state time window, (i.e. the amount of time during which Timer 1 remains constant), is stable to within about 20 ns. The window starts roughly around the rising edge of CLKOUT, however this timing is very approximate due to the amount of internal circuitry involved.

5.3 Standard I/O Port Pins

Port 0 is different from the other digital ports in that it is actually part of the A/D converter. The port is sampled once every state time, however, sampling is not synchronized to Timer 1. If this port is used, the input signal on the pin must be stable one state time before the reading of the SFR.

Port 1 and Port 2 have quasi-bidirectional I/O pins. When used as inputs the data on these pins must be stable one state time prior to reading the SFR. This timing is also valid for the input-only pins of Port 2 and is similar to the HSI in that the sample occurs just after the rising edge of CLKOUT. When used as outputs, the quasi-bidirectional pins will change state shortly after CLKOUT falls. If the change was from '0' to a '1' the low impedance pullup will remain on for one state time after the change.

Ports 3 and 4 are addressed as off-chip memory-mapped I/O. The port pins will change state shortly after the rising edge of CLKOUT. When these pins are used as Ports 3 and 4 they are open drains, their structure is different when they are used as part of the bus. See Section 10.4 of the 8X9X Architecture chapter. Additional information on port reconstruction is available in Section 7.7 of this chapter.

6.0 SERIAL PORT TIMINGS

The serial port on the 8X9X was designed to be compatible with the 8051 serial port. Since the 8051 uses a divide by 2 clock and the 8X9X uses a divide by 3, the serial port on the 8X9X had to be provided with its own clock circuit to maximize its compatibility with the 8051 at high baud rates. This means that the serial port itself does not know about state times. There is circuitry which is synchronized to the serial port and to the rest of the 8X9X so that information can be passed back and forth.

The baud rate generator is clocked by either XTAL1 or T2CLK. Because T2CLK needs to be synchronized to the XTAL1 signal its speed must be limited to $1/16$ that of XTAL1. The serial port will not function during the time between the consecutive writes to the baud rate register. Section 11.4 of the 8X9X Architecture chapter discusses programming the baud rate generator.

6.1 Mode 0

Mode 0 is the shift register mode. The TXD pin sends out a clock train, while the RXD pin transmits or receives the data. Figure 20 shows the waveforms and timing. Note that the port starts functioning when a '1' is written to the REN (Receiver Enable) bit in the serial port control register. If REN is already high, clearing the RI flag will start a reception.

In this mode the serial port can be used to expand the I/O capability of the 8X9X by simply adding shift registers. A schematic of a typical circuit is shown in Figure 21. This circuit inverts the data coming in, so it must be reinverted in software. The enable and latch connections to the shift registers can be driven by decoders, rather than directly from the low speed I/O ports, if the software and hardware are properly designed.

6.2 Mode 1 Timings

Mode 1 operation of the serial port makes use of 10-bit data packages, a start bit, 8 data bits and a stop bit. The transmit and receive functions are controlled by separate shift clocks. The transmit shift clock starts when the baud rate generator is initialized, the receive shift clock is reset when a '1 to 0' transition (start bit) is received. The transmit clock may therefore not be in sync with the receive clock, although they will both be at the same frequency.

The TI (Transmit Interrupt) and RI (Receive Interrupt) flags are set to indicate when operations are complete. TI is set when the last data bit of the message has been sent, not when the stop bit is sent. If an attempt to send another byte is made before the stop bit is sent the port will hold off transmission until the stop bit is complete. RI is set when 8 data bits are received, not when the stop bit is received. Note that when the serial port status register is read both TI and RI are cleared.

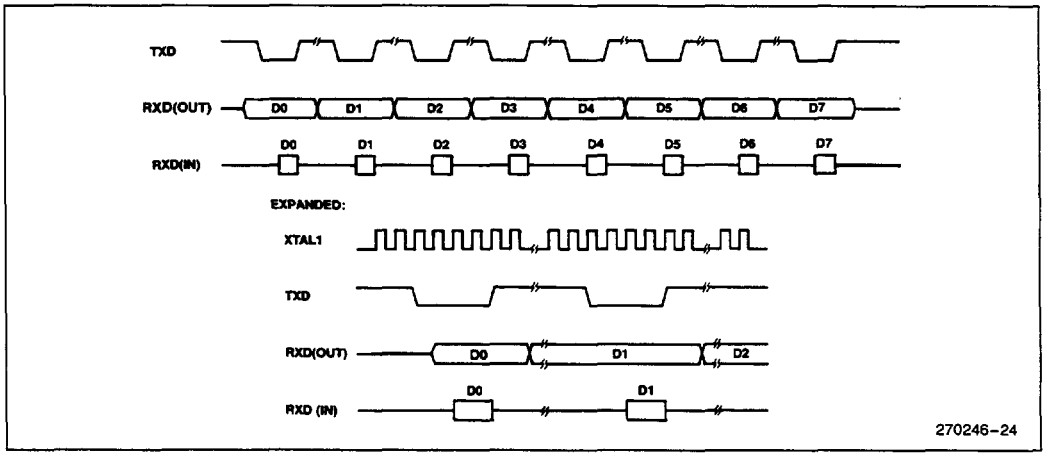


Figure 20. Serial Port Timings in Mode 0

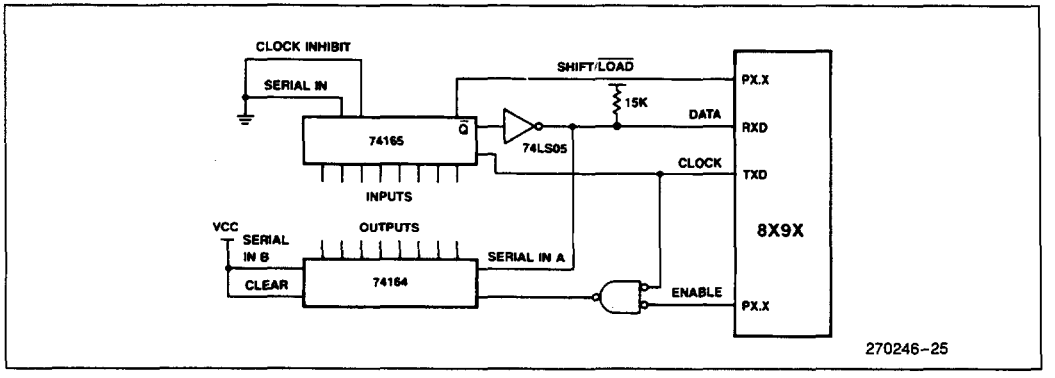


Figure 21. Mode 0 Serial Port Example

Caution should be used when using the serial port to connect more than two devices in half-duplex, (i.e. one wire for transmit *and* receive). If the receiving processor does not wait for one bit time after RI is set before starting to transmit, the stop bit on the link could be squashed. This could cause a problem for other devices listening on the link.

6.3 Mode 2 and 3 Timings

Modes 2 and 3 operate in a manner similar to that of Mode 1. The only difference is that the data is now made up of 9 bits, so 11-bit packages are transmitted and received. This means that TI and RI will be set on the 9th data bit rather than the 8th. The 9th bit can be used for parity or multiple processor communications (see Section 11 of the 8X9X Architecture chapter).

7.0 BUS TIMING AND MEMORY INTERFACE

7.1 Bus Functionality

The 8X9X has a multiplexed (address/data) bus which can be dynamically configured to have an 8-bit or 16-bit data width. There are control lines to demultiplex the bus ($\overline{ALE}$ or $\overline{ADV}$), indicate reads ($\overline{RD}$), indicate writes ($\overline{WRL}$ and $\overline{WRH}$, or $\overline{WR}$ with $\overline{BHE}$ and AD0), and a signal to indicate accesses that are for an instruction fetch (INST). Section 3.5 of the 8X9X Architecture chapter contains an overview of the bus operation.

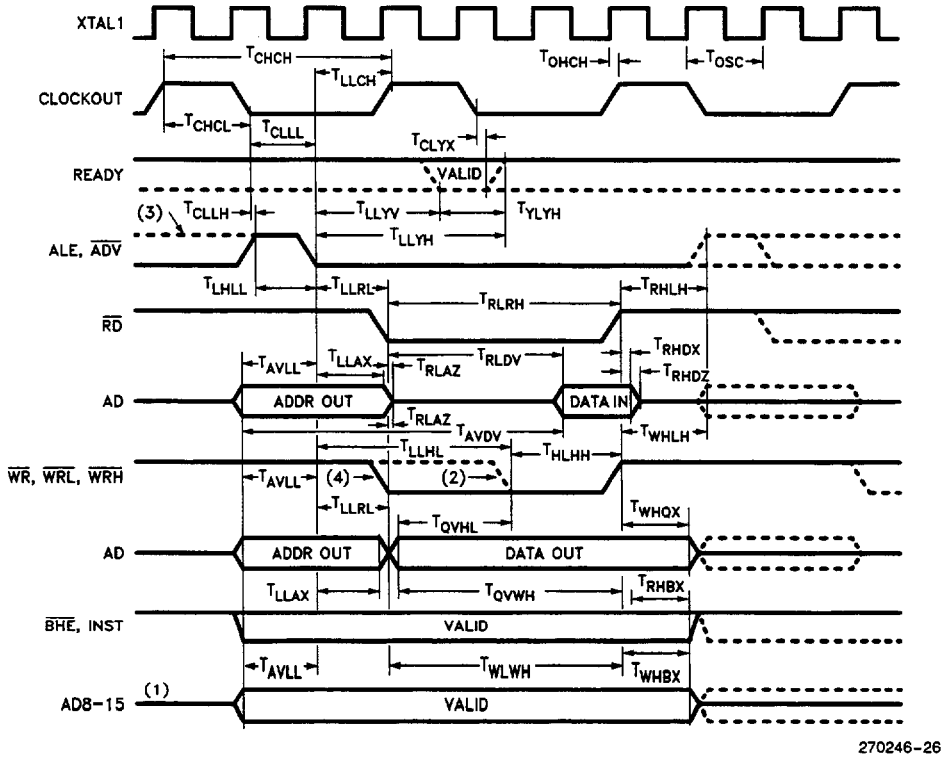
7.2 Timing Specifications

Figure 22 shows the timing of the bus signals and data lines. Please refer to the latest data sheet for the exact device you are using to ensure that your system is designed to the proper specifications. The major timing specifications are described in Figure 23.

7.3 READY Line Usage

When the processor has to address a memory location that cannot respond within the standard specifications, it is necessary to use the READY line to generate wait states. When the READY line is held low, the processor waits in a loop for the line to come high or until the number of inserted wait states is equal to the limit set in the Chip Configuration Register (see Section 2 of the MCS-96 Architecture chapter). There is a maximum time that the READY line can be held low without risking a processor malfunction due to dynamic nodes that have not been refreshed during the wait states. This time is shown as TYLYH in the data sheet.

In most cases the READY line is brought low after the address is decoded and it is determined that a wait state is needed. It is very likely that some addresses, such as those addressing memory mapped peripherals, would need wait states, and others would not. The READY line must be stable within the TLLYV specification after ALE falls or the processor could lock-up. There is



NOTES:
1. 8-bit bus only.
2. 8-bit or 16-bit bus and write strobe mode selected.
3. When $\overline{ADV}$ selected.
4. 8-bit or 16-bit bus and no write strobe mode selected.

Figure 22. Bus Signal Timings

no requirement as to when READY may go high, as long as the maximum READY low time (TYLYH) is not violated. To ensure that only one wait state is inserted it is necessary to provide external circuitry which brings READY high TLLYH after the falling edge of ALE/$\overline{\text{ADV}}$, or program the Chip Configuration Register to select a Ready Control limit of one.

Internally, the chip latches READY on the first falling edge of Phase A after ALE/$\overline{\text{ADV}}$ falls. Phase A is buffered and brought out externally as CLOCKOUT, so CLOCKOUT is a delayed Phase A. If a 1 is seen, the bus cycle proceeds uninterrupted with no wait state insertions. If a 0 is seen, one wait state (3 Tosc) is inserted.

If a wait state is inserted, READY is internally latched on the next rising edge of Phase A. If a 1 is found the bus cycle resumes with the net impact being the insertion of one wait state. If a 0 is seen, a second wait state is inserted.

The READY pin is again latched on the next rising edge of CLOCKOUT if two wait states were inserted. If the chip sees a 1, the bus cycle is resumed with the result being an insertion of two wait states. If another 0 is seen, a third wait state is inserted in the bus cycle and the READY pin is again latched on the following rising edge of CLOCKOUT. If internal Ready Control is not used, the READY line must at this point be a 1 to ensure proper operation.

Tosc—Oscillator Period, one cycle time on XTAL1.

Timings the Memory System Must Meet

TLLYH—ALE/$\overline{\text{ADV}}$ low to READY high: Maximum time after ALE/$\overline{\text{ADV}}$ falls until READY is brought high to ensure no more wait states. If this time is exceeded unexpected wait states may result. Nominally 1 Tosc + 3 Tosc × number of wait states desired.

TLLYV—ALE/$\overline{\text{ADV}}$ low to READY low: Maximum time after ALE/$\overline{\text{ADV}}$ falls until READY must be valid. If this time is exceeded the device could malfunction necessitating a chip reset. Nominally 2 Tosc periods.

TCLYX—READY hold after CLOCKOUT low: Minimum time that the value on the READY pin must be valid after CLOCKOUT falls. The minimum hold time is always zero nanoseconds.

TYLYH—READY low to READY high: Maximum time the part can be in the not-ready state. If it is exceeded, the 8X9X dynamic nodes which hold the current instruction may 'forget' how to finish the instruction.

TAVDV—ADDRESS valid to DATA valid: Maximum time that the memory has to output valid data after the 8X9X outputs a valid address. Nominally, a maximum of 5 Tosc periods.

TAVGV—ADDRESS valid to BUSWIDTH valid: Maximum time after ADDRESS becomes valid until BUSWIDTH must be valid. Nominally less than 2 Tosc periods.

TLLGV—ALE/$\overline{\text{ADV}}$ low to BUSWIDTH valid: Maximum time after ALE/$\overline{\text{ADV}}$ is low until BUSWIDTH must be valid. If this time is exceeded the part could malfunction necessitating a chip reset. Nominally less than 1 Tosc.

TLLGX—BUSWIDTH hold after ALE/$\overline{\text{ADV}}$ low: Minimum time that BUSWIDTH must be valid after ALE/$\overline{\text{ADV}}$ is low Nominally 1 Tosc.

TRLDV—$\overline{\text{READ}}$ low to DATA valid: Maximum time that the memory has to output data after $\overline{\text{READ}}$ goes low. Nominally, a maximum of 3 Tosc periods.

TRHDZ—$\overline{\text{READ}}$ high to DATA float: Time after $\overline{\text{READ}}$ is high until the memory must float the bus. The memory signal can be removed as soon as $\overline{\text{READ}}$ is not low, and must be removed within the specified maximum time from when $\overline{\text{READ}}$ is high. Nominally a maximum of 1 Tosc period.

TRHDX—DATA hold after $\overline{\text{READ}}$ goes high: Minimum time that memory must hold input DATA valid after $\overline{\text{RD}}$ is high. The hold time minimum is always zero nanoseconds.

TRLAZ—READ low to ADDRESS float: This is the bus control specifying the time from an active low READ signal until the 8X9X ADDRESS drivers for the cycle are off the bus. This is specified in order for data to be returned from the memory system without bus contention. Typically this is 0 ns for no bus contention. However, up to 10 ns is acceptable in systems.

Figure 23. Timing Specification Explanations

Timings the 8096 Will Provide

TOHCH—XTAL1 high to CLOCKOUT high: Delay from the rising edge of XTAL1 to the resultant rising edge on CLOCKOUT. Needed in systems where the signal driving XTAL1 is also used as a clock for external devices. Typically 50 to 100 nanoseconds.

TCHCH—CLKOUT high to CLKOUT high: The period of CLKOUT and the duration of one state time. Always 3 Tosc average, but individual periods could vary by a few nanoseconds.

TCHCL—CLKOUT high to CLKOUT low: Nominally 1 Tosc period.

TCLLH—CLKOUT low to ALE high: A help in deriving other timings. Typically plus or minus 5 ns to 10 ns.

TCLVL—CLOCKOUT low to ALE/$\overline{\text{ADV}}$ low: A help in deriving other timings. Nominally 1 Tosc.

TLLCH—ALE/$\overline{\text{ADV}}$ low to CLKOUT high: Used to derive other timings, nominally 1 Tosc period.

TLHLL—ALE/$\overline{\text{ADV}}$ high to ALE/$\overline{\text{ADV}}$ low: ALE/$\overline{\text{ADV}}$ high time. Useful in determining ALE/$\overline{\text{ADV}}$ rising edge to ADDRESS valid time. Nominally 1 Tosc period for ALE and 1 Tosc for $\overline{\text{ADV}}$ with back-to-back bus cycles.

TAVLL—ADDRESS valid to ALE/$\overline{\text{ADV}}$ low: Length of time ADDRESS is valid before ALE/$\overline{\text{ADV}}$ falls. Important timing for address latch circuitry. Nominally 1 Tosc period.

TLLAX—ALE/$\overline{\text{ADV}}$ low to ADDRESS invalid: Length of time ADDRESS is valid after ALE/$\overline{\text{ADV}}$ falls. Important timing for address latch circuitry. Nominally 1 Tosc period.

TLLRL—ALE/$\overline{\text{ADV}}$ low to $\overline{\text{READ}}$ or $\overline{\text{WRITE}}$ low: Length of time after ALE/$\overline{\text{ADV}}$ falls before $\overline{\text{RD}}$ or $\overline{\text{WR}}$ fall. Could be needed to ensure that proper memory decoding takes place before it is output enabled. Nominally 1 Tosc period.

TLLHL—ALE/$\overline{\text{ADV}}$ low to $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ low: Minimum time after ALE/$\overline{\text{ADV}}$ is low that the write strobe signals will go low. Could be needed to ensure that proper memory decoding takes place before it is output enabled. Nominally 2 Tosc periods.

TRLRH—$\overline{\text{READ}}$ low to $\overline{\text{READ}}$ high: $\overline{\text{RD}}$ pulse width, nominally 1 Tosc period.

TRHLH—$\overline{\text{READ}}$ high to ALE/$\overline{\text{ADV}}$ high: Time between $\overline{\text{RD}}$ going inactive and next ALE/$\overline{\text{ADV}}$, also used to calculate time between $\overline{\text{RD}}$ inactive and next ADDRESS valid. Nominally 1 Tosc period.

TRHBX—$\overline{\text{READ}}$ high to INST, $\overline{\text{BHE}}$, AD8–15 Inactive: Minimum time that the INST and $\overline{\text{BHE}}$ lines will be valid after $\overline{\text{RD}}$ goes high. Also the minimum time that the upper eight address lines (8-bit bus mode) will remain valid after $\overline{\text{RD}}$ goes high. Nominally 1 Tosc.

TWHBX—$\overline{\text{WRITE}}$ high to INST, $\overline{\text{BHE}}$, AD8–15 Inactive: Minimum time that the INST and $\overline{\text{BHE}}$ lines will be valid after $\overline{\text{WR}}$ goes high. Also the minimum time that the upper eight address lines (8-bit bus mode) will remain valid after $\overline{\text{WR}}$ goes high. Nominally 1 Tosc.

TWLWH—$\overline{\text{WRITE}}$ low to $\overline{\text{WRITE}}$ high: Write pulse width, nominally 3 Tosc periods.

THLHH—$\overline{\text{WRL}}$, $\overline{\text{WRH}}$ low to $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ high: Write strobe signal pulse width. Nominally 2 Tosc periods.

TQVHL—OUTPUT valid to $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ low: Minimum time that OUTPUT data is valid prior to write strobes becoming active. Needed for interfacing to memories that read data on the falling edge of write. Nominally 1 Tosc.

TQVWH—OUTPUT valid to $\overline{\text{WRITE}}$ high: Time that the OUTPUT data is valid before $\overline{\text{WR}}$ is high. Nominally 3 Tosc periods.

TWHQX—$\overline{\text{WRITE}}$ high to OUTPUT not valid: Time that the OUTPUT data is valid after $\overline{\text{WR}}$ is high. Nominally 1 Tosc period.

TWHLH—$\overline{\text{WRITE}}$ high to ALE/$\overline{\text{ADV}}$ high: Time between write high and next ALE/$\overline{\text{ADV}}$, also used to calculate the time between $\overline{\text{WR}}$ high and next ADDRESS valid. Nominally 2 Tosc periods.

Figure 23. Timing Specification Explanations (Continued)

7.4 INST Line Usage

The INST (Instruction) line is high during bus cycles that are for an instruction fetch and low for any other bus cycle. The INST signal (not present on 48-pin versions) can be used with a logic analyzer to debug a system. In this way it is possible to determine if a fetch was for instructions or data, making the task of tracing the program much easier.

7.5 BUSWIDTH Pin Usage

The BUSWIDTH pin is a control input which determines the width of the bus access in progress. BUSWIDTH is sampled after the rising edge of the first CLOCKOUT after ALE/$\overline{\text{ADV}}$ goes low. If a one is seen, the bus access progresses as a 16-bit cycle. If a zero is seen, the bus access progresses as an 8-bit cycle. The BUSWIDTH setup and hold timing requirements appear in the data sheet.

The BUSWIDTH pin can be overridden by causing the BUS WIDTH SELECT bit in the Chip Configuration Register (CCR) to be zero. This will permanently select an 8-bit bus width. However, if the BUS WIDTH SELECT bit in the CCR is a one, the BUSWIDTH pin determines the bus width. See Section 3.5 of the 8X9X Architecture chapter. Since the BUSWIDTH pin is not available on 48-pin or 64-pin devices, the BUS WIDTH SELECT bit in the CCR determines bus width.

7.6 Address Decoding

The multiplexed bus of the 8X9X must be demultiplexed before it can be used. This can be done with two 74LS373 transparent latches for an 8X9X in 16-bit bus mode, or one 74LS373 for an 8X9X in 8-bit bus mode. As explained in Section 3.5 of the 8X9X Architecture chapter, the latched address signals will be referred to as MA0 through MA15 (Memory Address), and the data lines will be called MD0 through MD15 (Memory Data).

Since the 8X9X can make accesses to memory for either bytes or words, it is necessary to have a way of determining the type of access desired when the bus is 16-bits wide. For write cycles, the signals Write Low ($\overline{\text{WRL}}$) and Write High ($\overline{\text{WRH}}$) are provided. $\overline{\text{WRL}}$ will go low during all word writes and during all byte writes to an even location. Similarly, $\overline{\text{WRH}}$ will go low during all word writes and during all byte writes to an odd location. During read cycles, an 8X9X in 16-bit bus mode will always do a word read of an even location. If only one byte of the word is needed, the chip discards the byte it does not need.

Since 8X9X memory accesses over an 8-bit wide bus are always bytes, only one write strobe is needed for write cycles. For this purpose the $\overline{\text{WRL}}$ signal was made to go low for all write cycles during 8-bit bus accesses. When a word operation is requested, the bus controller performs two byte-wide bus cycles.

In many cases it may be desirable to have a write signal with a longer pulse width than $\overline{\text{WRL}}$/$\overline{\text{WRH}}$. The Write ($\overline{\text{WR}}$) line of the 8X9X is an alternate control signal that shares a pin with $\overline{\text{WRL}}$ and is only available in 16-bit bus mode. $\overline{\text{WR}}$ is nominally one Tosc longer than the $\overline{\text{WRL}}$/$\overline{\text{WRH}}$ signals, but goes low for any write cycle. Therefore it is necessary to decode for the type of write (byte or word) desired.

The Byte High Enable ($\overline{\text{BHE}}$) signal and MA0 can be used for this purpose. $\overline{\text{BHE}}$ is an alternate control

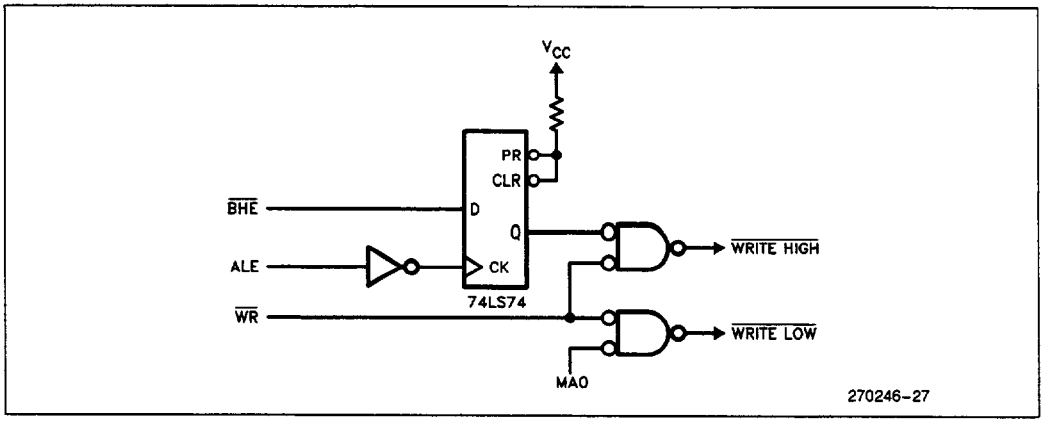


Figure 24. Decoding $\overline{\text{WR}}$ and $\overline{\text{BHE}}$ to Generate $\overline{\text{WriteLow}}$ and $\overline{\text{WriteHigh}}$

signal that shares a pin with $\overline{WRH}$. When $\overline{BHE}$ is low, the high byte of the 16-bit bus is enabled. When MA0 is low, the lower byte is enabled. When MA0 is low and $\overline{BHE}$ is low, both bytes are enabled. Figure 24 shows how to use $\overline{WR}$, $\overline{BHE}$ and $\overline{MA0}$ to decode bus accesses. It's important to note that this decoding inserts a delay in the write signal which must be considered in a system timing analysis.

External memory systems for the 8X9X can be set up in many ways. Figures 25 through 28 show block diagrams of memory systems using an 8-bit bus with a single EPROM, using an 8-bit bus with RAM and EPROM, using a 16-bit bus with two external EPROMs and using a 16-bit bus in a RAM and ROM system. (The timings for the systems shown are optimized for 10 MHz operation.)

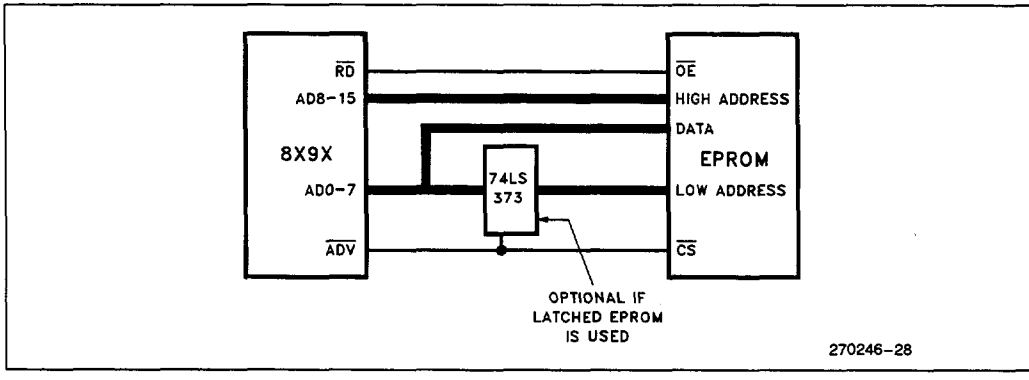


Figure 25. An 8-Bit Bus with EPROM Only

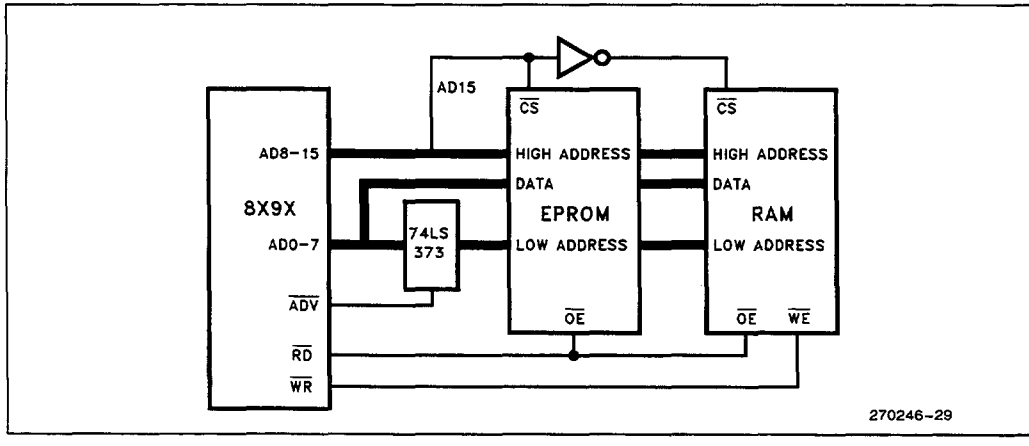


Figure 26. An 8-Bit Bus with EPROM and RAM

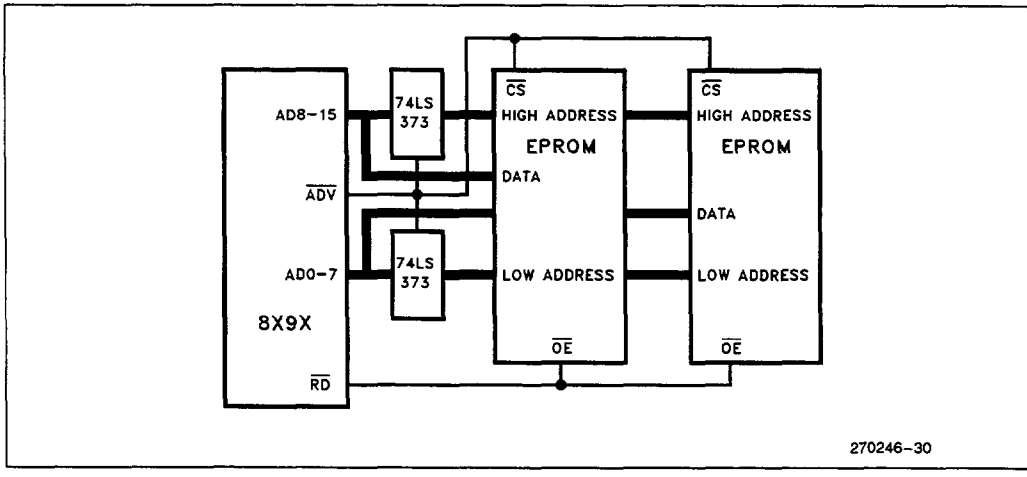


Figure 27. A 16-Bit Bus with EPROM Only

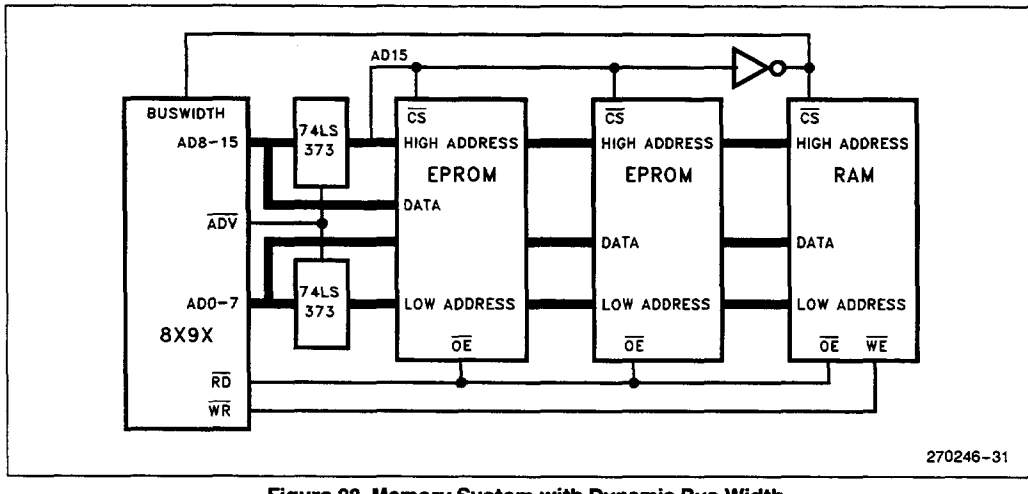


Figure 28. Memory System with Dynamic Bus Width

7.7 I/O Port Reconstruction

When a single-chip system is being designed using a multiple chip system as a prototype, it may be necessary to reconstruct I/O Ports 3 and 4 using a memory-mapped I/O technique. The circuit shown in Figure 30 provides this function. It can be attached to a 8X9X system which has the required address decoding and bus demultiplexing.

The output circuitry is basically just a latch that operates when 1FFEH or 1FFFH are placed on the MA lines. The inverters surrounding the latch create an open-collector output to emulate the open-drain output found on the 8X9X. The 'reset' line is used to set the ports to all 1's when the 8X9X is reset. It should be noted that the voltage and current characteristics of the port will differ from those of the 8X9X, but the basic functionality will be the same.

The input circuitry is just a bus transceiver that is addressed at 1FFEH or 1FFFH. If the ports are going to be used for either input or output, but not both, some of the circuitry can be eliminated.

8.0 NOISE PROTECTION TIPS

Designing controllers differs from designing other computer equipment in the area of noise protection. A microcontroller circuit under the hood of a car, in a photocopier, CRT terminal, or a high speed printer is subject to many types of electrical noise. Noise can get to the processor directly through the power supply, or it can be induced onto the board by electromagnetic fields. It is also possible for the PC board to find itself in the path of electrostatic discharges. Glitches and noise on the PC board can cause the processor to act unpredictably, usually by changing either the memory locations or the program counter.

There are both hardware and software solutions to noise problems, but the best solution is good design practice and a few ounces of prevention. The 8X9X has a Watchdog Timer which will reset the device if it fails to execute the software properly. The software should be set up to take advantage of this feature.

It is also recommended that unused areas of code be filled with NOPs and periodic jumps to an error routine or RST (reset chip) instructions. This is particularly important in the code around lookup tables, since if lookup tables are executed all sorts of bad things can happen. Wherever space allows, each table should be surrounded by 7 NOPs (the longest 8X9X instruction has 7 bytes) and a RST or jump to error routine instruction. This will help to ensure a speedy recovery should the processor have a glitch in the program flow.

Many hardware solutions exist for keeping PC board noise to a minimum. Ground planes, gridded ground and V_{CC} structures, bypass capacitors, transient absorbers and power busses with built-in capacitors can all be of great help. It is much easier to design a board with these features than to try to retrofit them later. Proper PC board layout is probably the single most important and, unfortunately, least understood aspect of project design. Minimizing loop areas and inductance, as well as providing clean grounds are very important. More information on protecting against noise can be found in the Application Note AP-125, "Designing Microcontroller Systems for Noisy Environments".

9.0 PACKAGING

The MCS-96 family of products is offered in many versions. They are available in 48-pin or 68-pin packages, with or without on-chip ROM/EPROM and with or without an A/D converter. A summary of the available options is shown in Figure 31.

The 48-pin versions are available in ceramic and plastic 48-pin Dual-In-Line package (DIP). The ceramic versions have order numbers with the prefix "C". The plastic versions have the prefix "P".

The 68-pin versions are available in a ceramic pin grid array (PGA), a plastic leaded chip carrier (PLCC) and a Type B leadless chip carrier (LCC). PGA devices have part numbers with the prefix "C". PLCC devices have the prefix "N". LCC devices have the prefix "R".

SHRINK-DIP is offered in 64-pin packages with a package designator of "U".

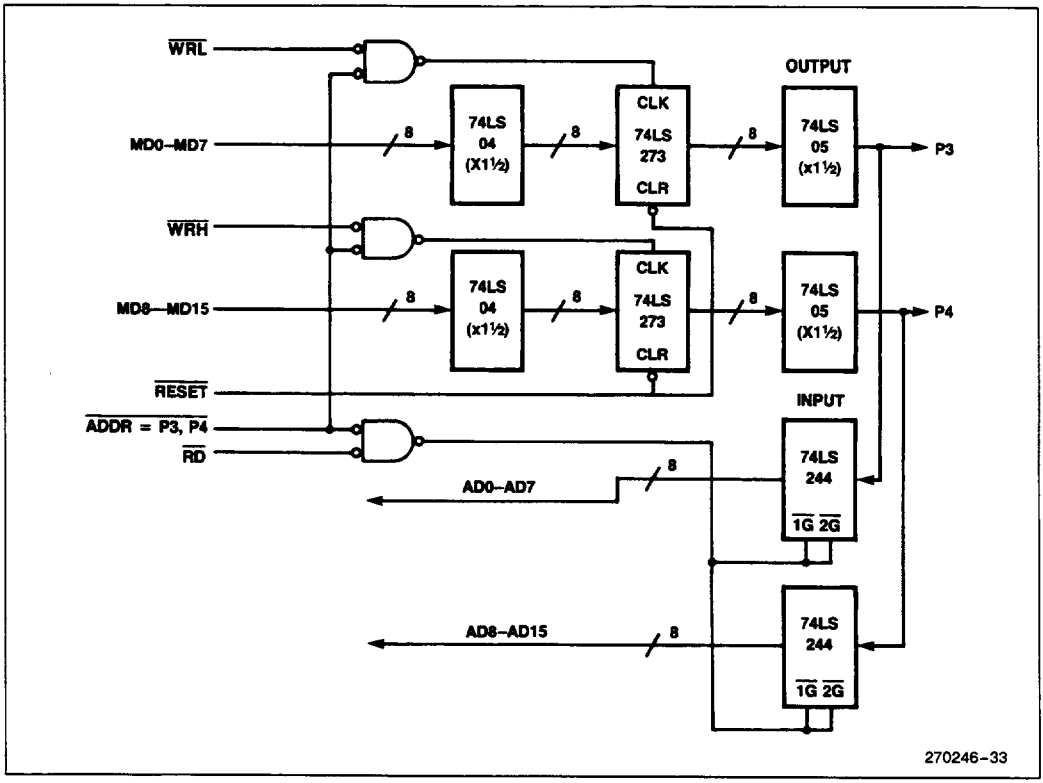


Figure 29. I/O Port Reconstruction

	Factory Masked ROM			CPU			User Programmable					
							EPROM			OTP		
	68-Pin	64-Pin	48-Pin	68-Pin	64-Pin	48-Pin	68-Pin	64-Pin	48-Pin	68-Pin	64-Pin	48-Pin
ANALOG	8397BH 8397JF	8397BH 8397JF	8395BH 8398	8097BH 8097JF	8097BH 8097JF	8095BH 8098	8797BH		8795BH 8798	8797BH 8797JF	8797JF 8797BH	8798
NO ANALOG	8396BH			8X9X								

Figure 30. HMOS MCS®96 Packaging

- 48-Pin devices have four Analog Input pins.
- For ROM/OTP/EPROM devices, 8X9XBH and 8X98 = 8 Kbytes, 8X9XJF = 16 Kbytes
- 64-Pin devices have all 48-Pin device features plus the following:
  - Four additional Analog Input channels
  - One additional Quasi-Bidirectional 8 Bit Parallel Port
  - Four additional Port 2 pins with multiplexed features
    - Timer 2 Clock Source Pin
    - Timer 2 Reset pin
    - Two additional quasi-bidirectional port pins
- 68-Pin devices have all 48- and 64-pin features plus the following:
  - Dynamic Buswidth sizing (8 or 16 bit bus)
  - Dedicated System Clock Output (CLKOUT)
  - INST pin for memory expansion
  - Non-Maskable Interrupt for debugging
- Package Designators:
  - N = PLCC
  - C = Ceramic DIP
  - A = Ceramic Pin Grid Array
  - P = Plastic DIP
  - R = Ceramic LCC
  - U = Shrink DIP

10.0 USING THE EPROM

This section refers to the 879XBH, 8798, and 879XJF devices. These devices are generically referred to as the 879X. All information in this section refers to all three devices unless otherwise noted.

879XBH and the 8798 contain 8 Kbytes of ultraviolet Erasable and electrically Programmable Read Only Memory (EPROM). The 879XJF contains 16 Kbytes of EPROM. When $\overline{EA}$ is a TTL high, the EPROM is located at memory locations 2000H through 3FFFH on the 879XBH and the 8798. It is at locations 2000H through 5FFFH on the 879XJF.

Applying + 12.75V to $\overline{EA}$ when the chip is reset places the 879X device in the EPROM Programming Mode. The Programming Mode supports EPROM programming and verification. The following is a brief description of each of the programming modes:

The Auto Configuration Byte Programming Mode programs the Programming Chip Configuration Byte and the Chip Configuration Byte.

The Auto Programming Mode enables an 879X to program itself and up to 15 other 879X's.

The Slave Programming Mode provides a standard interface to program any number of 879X's by a master device such as an EPROM programmer or another 879X.

The Run-Time Programming Mode allows individual EPROM locations to be programmed at run-time under complete software control. (Run-Time Programming is done with $\overline{EA}$ = 5V.)

Some I/O pins have new functions for programming. These pins determine the programming function, provide programming control signals and slave ID numbers, and pass error information. Figure 32 shows how the pins are renamed. Figure 33 describes each new pin function. PMODE selects the function to be performed (see Figure 31).

PMODE	Programming Mode
0–4	Reserved
5	Slave Programming Mode
6	ROM Dump Mode
7–0BH	Reserved
0CH	Auto Programming Mode
0DH	Program Configuration Byte
0EH–0FH	Reserved

Figure 31. Programming Function PMODE Values

When an 879X EPROM device is not being erased, the window must be covered with an opaque label. This prevents functional degradation and data loss from the array.

10.1 Power-Up and Power-Down

To avoid damaging devices during programming, follow these rules:

RULE #1— V_{PP} must be within 1V of V_{CC} while V_{CC} is below 4.5V.

RULE #2— V_{PP} can not be higher than 5.0V until V_{CC} is above 4.5V.

RULE #3— V_{PP} must not have a low impedance path to ground when V_{CC} is above 4.5V.

RULE #4— $\overline{EA}$ must be brought to 12.75V before V_{PP} is brought to 12.75V (not needed for run-time programming).

RULE #5— The PMODE and SID pins must be in their desired state before RESET rises.

RULE #6— All voltages must be within tolerance and the oscillator stable before RESET rises.

RULE #7— The supplies to V_{CC}, V_{PP}, $\overline{EA}$ and RESET must be well regulated and free of glitches and spikes.

To adhere to these rules you can use the following power-up and power-down sequences:

POWER UP

RESET = 0V

$V_{CC} = V_{PP} = \overline{EA} = 5V$

CLOCK on (if using an external clock instead of the internal oscillator)

PALE = PROG = PORT3, 4 = V_{IH}(1)

SID and PMODE valid

$\overline{EA}$ = 12.75V(2)

V_{PP} = 12.75V(3)

WAIT (wait for supplies and clock to settle)

RESET = 5V

WAIT Tshll (see data sheet)

BEGIN

POWER DOWN

RESET = 0V

V_{PP} = 5V

$\overline{EA}$ = 5V

PALE = PROG = SID = PMODE = PORT3, 4 = 0V

V_{CC} = V_{PP} = $\overline{EA}$ = 0V

CLOCK OFF

NOTES:

1. V_{IH} = logical '1' (2.4V minimum)
2. The same power supply can be used for $\overline{EA}$ and V_{PP}. However, the $\overline{EA}$ pin must be powered up before V_{PP} is powered up. Also, $\overline{EA}$ should be protected from noise to prevent damage to $\overline{EA}$.
3. Exceeding the maximum limit on V_{PP} for any amount of time could damage the device permanently. The V_{PP} source must be well regulated and free of glitches and spikes.

10.2 Reserved Locations

Fill all Intel Reserved locations except address 2019H, when mapped internally or externally, with 0FFH to ensure compatibility with future devices. Fill address 2019H with 20H.

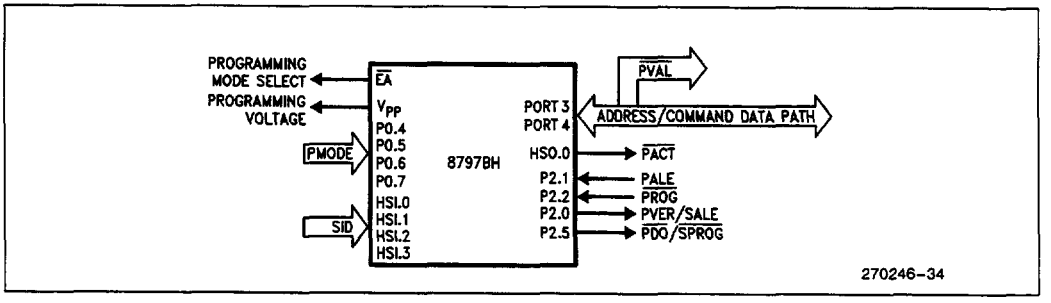


Figure 32. Programming Mode Pin Function

Mode	Name	Function
General	PMODE (P0-.4, .5, .6, .7)	**PROGRAMMING MODE SELECT:** Determines the EPROM programming algorithm that is performed. PMODE is sampled after a chip reset and should be static while the device is operating.
Auto Programming Mode	$\overline{\text{PACT}}$ (HSO.0)	**PROGRAMMING ACTIVE:** Used in the Auto-Programming Mode. Indicates when programming activity is complete.
	$\overline{\text{PVAL}}$ (Ports 3 and 4)	**PROGRAM VALID:** These signals indicate the success or failure of programming in the Auto Programming Mode and when using this mode for gang programming. For the Auto Programming Mode this signal is asserted at Port 3.0. When using this mode for gang programming, all bits of Port 3 and Port 4 are asserted to indicate programming validity of the various slaves. A zero indicates successful programming on PVAL.0. A zero on PVAL.1 through PVAL.15 indicates a fail.
	SALE (P2.0)	**SLAVE ALE:** Output signal from an 879X in the Auto Programming Mode. A falling edge on SALE indicates that Ports 3 and 4 contain valid address/ command information for slave 879XBHs that may be attached to the master.
	$\overline{\text{SPROG}}$ (P2.5)	**SLAVE PROGRAMMING PULSE:** Output from an 879X in the Auto Programming Mode. A falling edge on $\overline{\text{SPROG}}$ indicates that Ports 3 and 4 contain valid data for programming into slave 879XBHs that may be attached to the master.
	Ports 3 and 4	**ADDRESS/COMMAND/DATA BUS:** Used by devices in the Auto Programming Mode to pass command, addresses and data to slaves. Also used in the Auto Programming Mode as a regular system bus to access external memory. Each line should be pulled up to VCC through a resistor. Also used as PVAL (see above).
Slave Programming Mode	SID (HSI-0, .1, .2, .3)	**SLAVE ID NUMBER:** Used to assign a pin of Port 3 or 4 to each slave to pass programming verification acknowledgement. For example, if gang programming in the Slave Programming Mode, the slave with SID = 0001 will use Port 3.1 to signal correct or incorrect program verification.
	PALE (P2.1)	**PROGRAMMING ALE INPUT**: Accepted by an 879X that is in the Slave Programming Mode. Indicates that Ports 3 and 4 contain a command/ address.
	$\overline{\text{PROG}}$ (P2.2)	**PROGRAMMING PULSE:** Accepted by 879X that is in the Slave Programming Mode. Used to indicate that Ports 3 and 4 contain the data to be programmed. A falling edge on $\overline{\text{PROG}}$ signifies data valid and starts the programming cycle. A rising edge on $\overline{\text{PROG}}$ will halt programming in the slaves.
	PVER (P2.0)	**PROGRAM VERIFIED:** A signal output after a programming operation by devices in the Slave Programming Mode. This signal is on Port 2.0 and is asserted as a logic 1 if the bytes program correctly.
	$\overline{\text{PDO}}$ (P2.5)	**PROGRAMMING DURATION OVERFLOWED:** A signal output by devices in the Slave Programming Mode. Used to signify that the $\overline{\text{PROG}}$ pulse applied for a programming operation was longer than allowed.
	Ports 3 and 4	**ADDRESS/COMMAND/DATA BUS:** Used to pass commands, addresses and data to and from slave mode 879X's.
Auto PCCB Programming Mode	PVER (P2.0)	**PROGRAM VERIFIED:** A signal output after programming in the Auto Configuration Byte Programming Mode. The signal is on Port 2.0 and is asserted as a logic 1 if the bytes program correctly.
	PALE (2.1)	**PROGRAMMING ALE INPUT:** Used by a device in the Auto Program Configuration Byte Mode to indicate that Port 3 contains the data to be programmed into the PCCB and CCB.

Figure 33. Programming Mode Pin Definitions

10.3 Auto Configuration Byte Programming Mode

The Programming Chip Configuration Byte (PCCB) is a non-memory mapped EPROM location. It gets loaded into the CCR during reset for auto and slave programming. The Auto Configuration Byte Programming Mode programs the PCCB.

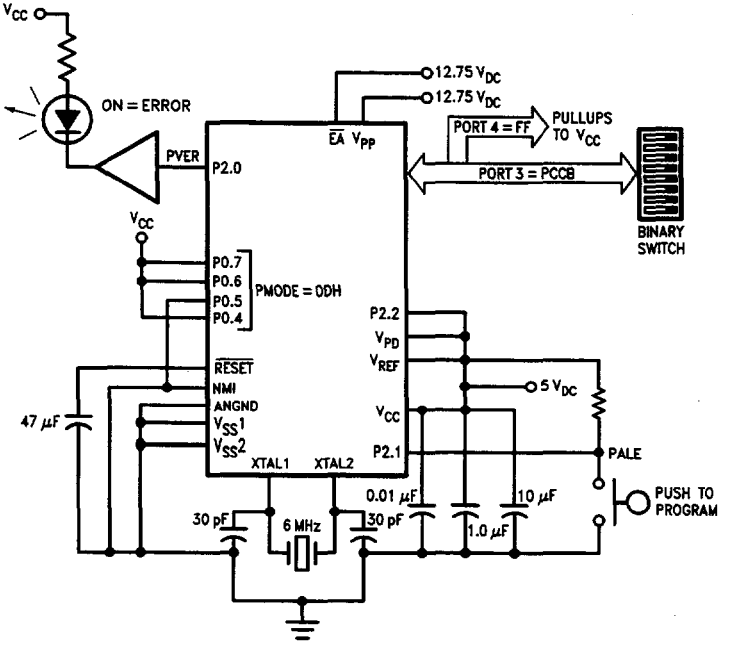
The Chip Configuration Byte (CCB) is at location 2018H and can be programmed like any other EPROM location using auto, slave and run-time programming. However, you can also use the Auto Configuration Byte Programming to program the CCB when no other locations need to be programmed. The CCB is programmed with the same value as the PCCB.

The Auto Configuration Byte Programming Mode is entered by following the power-up sequence described in Section 10.1 with PMODE = 0DH, Port 4 = 0FFH, and Port 3 = the data to be programmed into the PCCB and CCB. When a 0 is placed on PALE, the CCB and PCCB are automatically programmed with the data on Port 3. After programming, PVER is driven high if the bytes programmed correctly and low if they did not. Programming takes approximately 250 ms. Figure 34 shows a minimum configuration for Auto Configuration Byte Programming.

Once the CCB and PCCB are programmed, all programming activities and bus operations use the selected bus width, READY control, bus controls, and READ/WRITE protection until you erase the device. You must be careful when programming the READ and WRITE lock bits in the CCB and PCCB. If you enable the READ and WRITE lock bits in the CCB or the PCCB and then reset the device, the array may no longer be programmed or verified (see Figure 41 in Section 10.7.1). Therefore, you should program the buswidth, READY control, and bus controls using the Auto Configuration Byte Programming Mode. You should program the READ and WRITE lock bits when all programming is complete.

If the PCCB is not programmed, the CCR will be loaded with 0FFFH when the device is in the Programming Mode.

Specific requirements for CCB and PCCB programming are included in the Auto, Slave, and Run-time Programming sections.



NOTES:
1. Tie Port 3 to the value desired to be programmed into CCB, and PCCB.
2. Make all necessary minimum connections for power, ground and clock.

Figure 34. The Auto CCR Programming Mode

10.4 Auto Programming Mode

The Auto Programming Mode provides the ability to program the 879X EPROM without using an EPROM programmer. For this mode follow the power-up sequence described in Section 10.1 with PMODE = 0CH. When RESET rises, the 879XBH and 8798 devices automatically program themselves with the data found at external locations 4000H through 5FFFH. The 879XJF programs itself with the data found at external locations 4000H through 7FFF. Figure 35 shows a minimum configuration for using an 8K x 8 EPROM to program one 879X in the Auto Programming Mode.

The 879X reads a word from external memory, then the Modified Quick-Pulse Programming™ Algorithm (described later) is used to program the appropriate EPROM location. Since the erased state of a byte is 0FFH, the Auto Programming Mode will skip locations where the data to be programmed is 0FFH. When all 8K of the 879XBH and 8798 and all 16K of the 879XJF has been programmed, $\overline{\text{PACT}}$ goes high and the devices outputs a 0 on Port 3.0 ($\overline{\text{PVAL}}$) if it programmed correctly and a 1 if it failed.

10.4.1. AUTO PROGRAMMING MODE AND THE CCB/PCCB

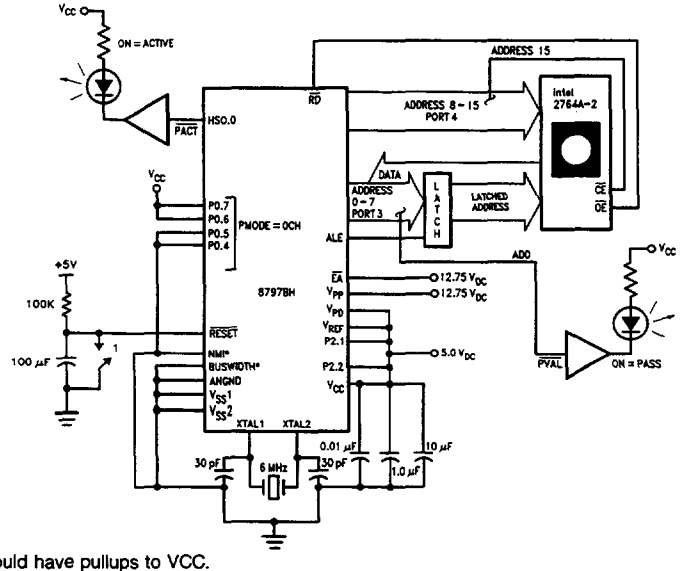
In the Auto Programming Mode the CCR is loaded with the PCCB. The PCCB must correspond to the memory system of the programming setup, including the READY and bus control selections. You can program the PCCB using the Auto Configuration Byte Programming Mode (see Section 10.3).

Auto Programming must be done in 8-bit bus mode. For 68L devices you must tie the BUSWIDTH pin to ground. You do not need to program the buswidth selection bit in the PCCB (PCCB.1). For 48L and 64L devices there is no BUSWIDTH pin. You must program PCCB.1 using the Auto Configuration Byte Programming Mode before programming the array.

The data in the PCCB takes effect upon reset. If you enable either the READ or WRITE lock bits during Auto Programming but do not reset the device, Auto Programming will continue. If you enable either the READ or WRITE lock bits and then reset the device, the device will no longer program or verify. You should program these bits when no more programming will be done.

10.4.2 GANG PROGRAMMING WITH THE AUTO PROGRAMMING MODE

An 879X in the Auto Programming Mode can also be used as a programmer for up to 15 other 879XBHs that are configured in the Slave Programming Mode. The 879X acts as the master. The master programs the slaves with the same data the master is programming itself with. The master outputs the necessary slave command/data pairs on Ports 3 and 4. It also provides the Slave ALE (SALE) and Slave $\overline{\text{PROG}}$ ($\overline{\text{SPROG}}$) signals to demultiplex the commands from the data. Figure 36 is a block diagram of a gang programming system using one 879XBH in the Auto Programming Mode. The Slave Programming Mode is described in the next section.



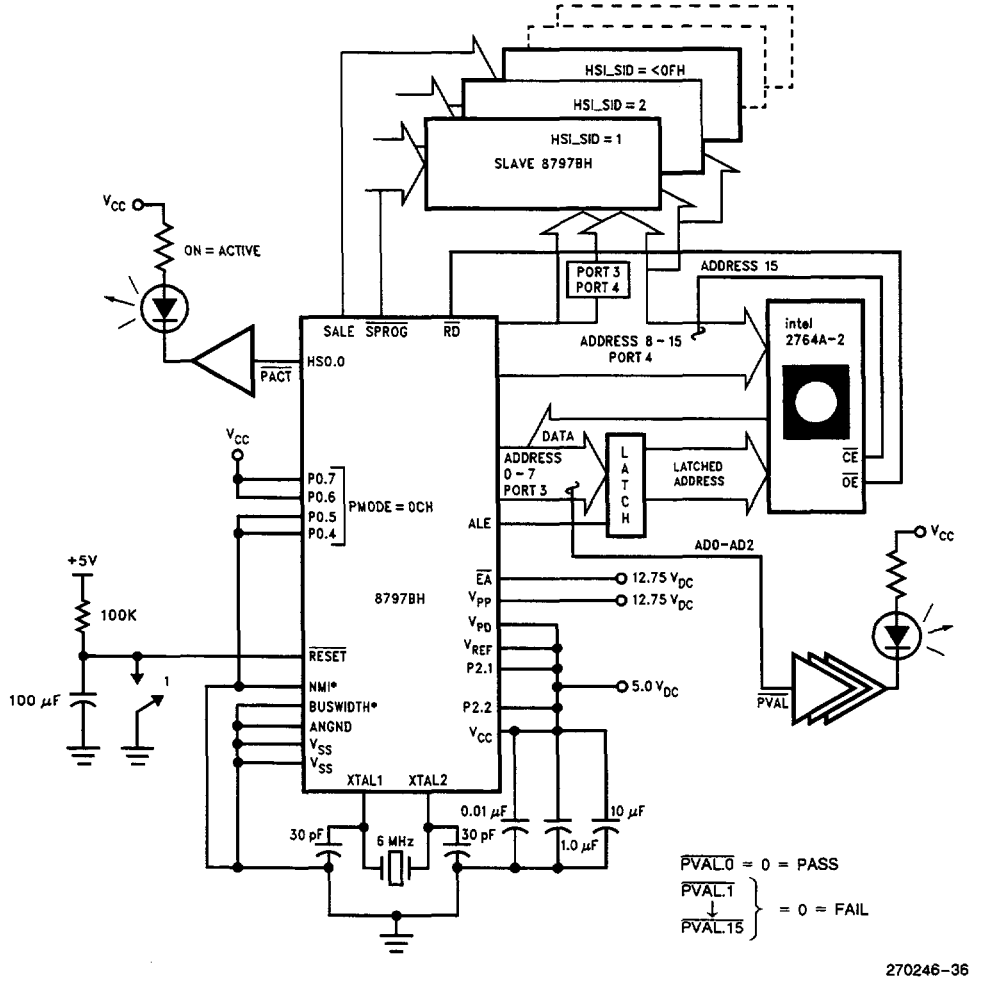
NOTES:
*Ports 3 and 4 should have pullups to VCC.
1. Allow RESET to rise after the voltages to V_{CC}, $\overline{EA}$, and V_{PP} are stable.

270246-35

Figure 35. The Auto Programming Mode

The master 879X reads a word from the external memory controlled by ALE, $\overline{RD}$ and $\overline{WR}$. It then drives Ports 3 and 4 with a Data Program command using the appropriate address and alerts the slaves with a falling edge on SALE. Next, the data to be programmed is driven onto Ports 3 and 4 and slave programming begins with a falling edge on $\overline{SPROG}$. At the same time, the master begins to program its own EPROM location with the data read in. Intel's Modified Quick-Pulse Programming™ Algorithm is used, with Data Verify commands being given to the slaves after each programming pulse.

When programming is complete $\overline{PACT}$ goes high and Ports 3 and 4 are driven with all 1s if all devices programmed correctly. Individual bits of Port 3 and 4 will be driven to 0 if the slave with that bit number as an SID did not program correctly. The 879X used as the master assigns itself an SID of 0.



NOTES:
*$\overline{EA}$ and VPP on slaves must be at +12.75 Vdc. Each slave's PMODE must equal 05H. Ports 3 and 4 should have pullups to VCC. Minimum configuration connections must also be made for slaves. A 10 MHz clock is recommended for the slaves.
1. Allow RESET to rise after the voltages to V_{CC}, $\overline{EA}$, and V_{PP} are stable.

Figure 36. Gang Programming with the Auto Programming Mode

10.5 Slave Programming Mode

Any number of 879Xs can be programmed by a master programmer using the Slave Programming Mode.

In Slave Programming Mode, the device being programmed uses Port 3, 4 as a command/data path. PALE and PROG demultiplex the commands and data. PVER, PDO and Ports 3 and 4 pass error information to the programmer. There is no 879X dependent limit to the number of devices that can be gang programmed in the slave mode.

It is important to note that the interface to an 879X in the slave mode is similar to a multiplexed bus. Issuing consecutive PALE pulses without a corresponding $\overline{\text{PROG}}$ pulses will produce unexpected results, as will issuing consecutive $\overline{\text{PROG}}$ pulses without the corresponding PALE pulses.

10.5.1 SLAVE PROGRAMMING COMMANDS

The commands sent to the slaves are 16-bits wide and contain two fields. Bits 14 and 15 specify the action that the slaves are to perform. Bits 0 through 13 specify the address upon which the action is to take place. On the 879XJF, P4.6 is both the least significant bit of the "Data Program Upper 8K" command and the most significant bit of the address. Commands are sent via Ports 3 and 4 and are available to cause the slaves to program a word, verify a word, or dump a word (Table 1). The address part of the command sent to the slaves ranges from 2000H to 3FFFH on the 879XBH and the 8798 and from 2000H to 5FFFH on the 879XJF and refers to the internal EPROM memory space. The following sections describe each slave programming mode command.

Table 1. Slave Programming Mode Commands

P4.7	P4.6	Action
0	0	Word Dump
0	1	Data Verify
1	0	Data Program Lower 8K
1	1	Data Program Upper 8K (879XJF)

DATA PROGRAM COMMAND—After a Data Program Command has been sent to the slaves, $\overline{\text{PROG}}$ must be pulled low to program the data on Ports 3 and 4 into the location specified during the command. The falling edge of $\overline{\text{PROG}}$ indicates data valid and also triggers the hardware programming of the word specified. The slaves will begin programming 48 states after $\overline{\text{PROG}}$ falls, and will continue to program the location until $\overline{\text{PROG}}$ rises.

After the rising edge of $\overline{\text{PROG}}$, the slaves automatically perform a verification of the address just programmed. The result of this verification is then output on PVER (Program Verify) and $\overline{\text{PDO}}$ (Program Duration Overflowed). Therefore, verification information is available for programming systems that cannot use the Data Verify command.

If PVER and $\overline{\text{PDO}}$ of all slaves are 1s after $\overline{\text{PROG}}$ rises then the data program was successful everywhere. If any slave's PVER is a 0, then the data programmed did not verify correctly in that device. If any slave's $\overline{\text{PDO}}$ is a 0, then the programming pulse in those devices was terminated by an internal safety feature rather than the rising edge of $\overline{\text{PROG}}$. The safety feature prevents overprogramming in the slave mode. Figure 37 shows the relationship of PALE, $\overline{\text{PROG}}$, PVER and $\overline{\text{PDO}}$ to the Command/Data Path on Ports 3 and 4 for the Data Program Command.

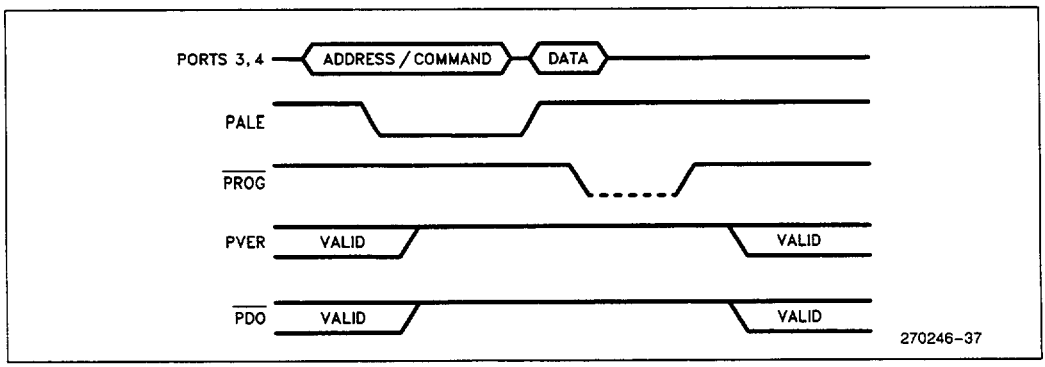


Figure 37. Data Program Signals in Slave Programming Mode

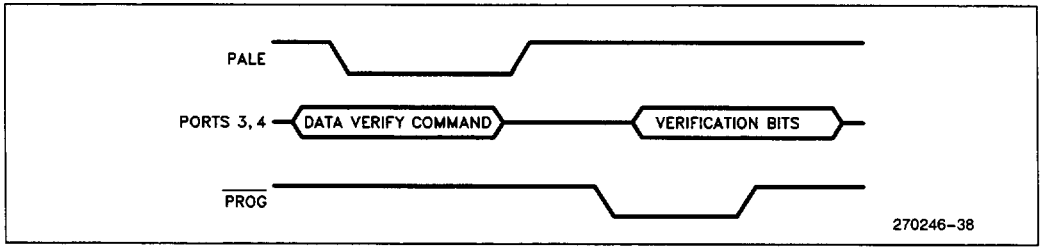


Figure 38. Data Verify Command Signals

DATA VERIFY COMMAND—When the Data Verify Command is sent, the slaves indicate correct or incorrect verification of the previous Data Program by driving one bit of Ports 3 and 4. A 1 indicates correct verification, while a 0 indicates incorrect verification. The SID (Slave ID Number) of each slave determines which bit of Ports 3 and 4 is driven. $\overline{PROG}$ from the programmer governs when the slaves drive the bus. Figure 38 shows the relationship of Ports 3 and 4 to PALE and $\overline{PROG}$.

The data verify command is always preceded by a Data Program Command in a programming system with as many as 16 slaves. However, a Data Verify Command does not have to follow every Data Program Command.

WORD DUMP COMMAND—When the Word Dump Command is issued, the 879X being programmed adds 2000H to the address field of the command and places the value found at the new address on Ports 3 and 4. For example, when the slave receives the command #0100H, it will place the word found at location 2100H on Ports 3 and 4. $\overline{PROG}$ from the programmer governs when the slave drives the bus. The signals are the same as shown in Figure 22.

Note that this command only works when a single slave is attached to the bus, and that there is no restriction on commands that precede or follow a Word Dump Command.

10.5.2 GANG PROGRAMMING WITH THE SLAVE PROGRAMMING MODE

Gang programming of 879Xs can be done using the Slave Programming Mode. There is no 879X based limit on the number of chips that may be hooked to the same Port 3/Port 4 data path for gang programming.

If more than 16 chips are being gang programmed, the PVER and $\overline{PDO}$ outputs of each chip can be used for verification. The master programmer can issue a data program command and then either watch every chip's error signals, or AND all the signals together to get a system PVER and $\overline{PDO}$.

If 16 or fewer 879Xs are to be gang programmed at once, a more flexible form of verification is available by giving each chip being programmed a unique SID. The master programmer can then issue a data verify command after the data program command. When a verify command is seen by the slaves, each will drive one pin of Port 3 or 4 with a 1 if the programming verified correctly or a 0 if programming failed. The SID of each slave determines which Port 3, 4 bit it is assigned. An 879X in the Auto Programming Mode could be the master programmer if 15 or fewer slaves need to be programmed (see Gang Programming with the Auto Programming Mode).

10.5.3 SLAVE PROGRAMMING MODE AND THE CCB/PCCB

Devices in the Slave Programmng Mode use Ports 3 and 4 as the command/data path. The data bus is not used. Therefore, you do not need to program either the CCB or the PCCB before starting slave programming.

You can program the CCB during slave mode programming like any other location. Data programmed into the CCB takes effect upon reset. If you enable either the READ or WRITE lock bits in the CCB and do not reset the device, slave programming will continue. If you enable either the READ or WRITE lock bits and do reset the device, the device will no longer program or verify. You should program the READ and WRITE lock bits using slave programming when the array is fully programmed and verified.

10.6 Run-Time Programming

Using Run-Time programming, the 879X can program itself under software control. One byte or word can be programmed instead of the whole array. The only additional requirements are that you apply a programming voltage to V_{PP} and have the ambient temperature at 25°C. Run-time programming is done with $\overline{EA}$ at a TTL high (internal memory enabled).

To run-time program the user writes a byte or word to the location to be programmed. The 879X will continually program that location until another data read or data write to the EPROM occurs. The user must control the duration of the programming pulse by implementing the Modified Quick-Pulse Programming Algorithm (see Section 10.8) in software.

Figure 39 is an example of code for programming an EPROM location while the device is executing internally. Upon entering the PROGRAM routine, the device retrieves the address and data from the STACK. A software timer is set to expire after one programming pulse. The 879X starts programming a location by writing to it. The device then goes into a "Jump to Self" loop while the location is programmed. ("Jump to Self" is a two byte instruction which can be CALL'ed from address 201AH.) When the software timer interrupt occurs, the device escapes from the "Jump to Self" loop, ending the programming pulse. The minimum interrupt service routine would remove the 201AH return address from the STACK and return.

Once you start programming a location, you should not perform any program fetches or pre-fetches from the EPROM. The fetches will be done but programming will stop. Using the "Jump to Self" prevents this from happening because address 201AH is not part of the EPROM. If the program is executing from external memory no program fetches or pre-fetches will occur from internal memory.

10.6.1 RUN-TIME PROGRAMMING AND THE CCB/PCCB

For run-time programming, the CCR is loaded with the CCB. Run-time programming is done with $\overline{EA}$ equal to a TTL-high (internal execution) so the internal CCB must correspond to the memory system of the application setup. You can use Auto Configuration Byte Programming or a generic programmer to program the CCB before using run-time programming.

The CCB can also be programmed during Run-Time Programming like any other EPROM location.

```
PROGRAM:

    POP  temp                                   ;take parameters from the
                                                 STACK
    POP  address_temp
    POP  data_temp
    PUSH temp

    PUSHF                                       ;save current status
    LDB  int_mask , #enable_swt_only            ;enable only swt interrups
    LDB  HSO_COMMAND , #SWT0_ovf                ;load swt command to interrupt
    ADD  HSO_TIME,TIMER1, #program_pulse        ;when program pulse time
                                                ;has elapsed
    EI
    ST   data-temp, [address_temp]              ;start programming

    CALL 201AH                                  ;"Jump to Self" until
                                                ;the program pulse time
                                                ;has expired

    POPF
    RET

SWT_ISR:
    . . .

swt0_expired:
    POP  0
    RET
    . . .
```

Figure 39. Programming the EPROM from Internal Memory Execution

Data programmed into the CCB takes effect upon reset. If the WRITE lock bit of the CCB is enabled the array can no longer be programmed. You should only program the WRITE lock bit when no further programming will be done to the array. If the READ lock bit is enabled the array can still be programmed using run-time programming but data accesses will only be performed when the program counter is between 2000H and 3FFFH on the 879XBH and the 8798 and between 2000H and 5FFFH on the 879XJF.

10.7 ROM/EPROM Program Lock

Protection mechanisms have been provided on the ROM and EPROM versions of the 8X9X to inhibit unauthorized accesses of internal program memory. However, there must always be a way to allow authorized program memory dumps for testing purposes. The following describes 8X9X lock features and the mode provided for authorized memory dumps.

10.7.1 LOCK FEATURES

Write protection is provided for EPROM devices while READ protection is provided for both ROM and EPROM devices.

Write protection is enabled by causing the LOC0 bit in the CCR to take the value 0. When WRITE protection is selected, the bus controller will cycle through the write sequence, but will not actually drive data to the EPROM and will not enable V_{PP} to the EPROM. This protects the entire EPROM (locations 2000H–3FFFH on the 879XBH and the 8798 and locations 2000H–5FFFH on the 879XJF) from inadvertant or unauthorized programming. It also prevents writes to the EPROM from upsetting program execution. If write protection is not enabled, a data write to an internal EPROM location will begin programming that location, and continue programming the location until a data access of the internal EPROM is executed. While programming, instruction fetches from internal EPROM will not be successful and programming will stop.

READ protection is selected by causing the LOC1 bit in the CCR to take the value 0. When READ protection is enabled, the bus controller will only perform a data read from the address range 2020H–3FFFH if the slave program counter is in the range 2000H–3FFFH on the 879XBH and the 8798. The bus controller will only perform a data read from the address range 2020H–5FFFH if the slave program counter is in the range 2000H–5FFAH on 879XJF. Note that since the slave PC can be many bytes ahead of the CPU program counter, an instruction that is located after address 3FFAH may not be allowed to access protected memory, even though the instruction is itself protected.

If the bus controller receives a request to perform a READ of protected memory, the READ sequence occurs with indeterminant data being returned to the CPU.

Figure 41 shows the effects of enabling the READ and WRITE lock bits.

CCB.1 RD Lock	CCB.0 WR Lock	PCCB.1 RD Lock	PCCB.0 WR Lock	Protection
1	1	1	1	Array is unprotected. ROM Dump Mode and all programming modes are allowed.
0	1	1	1	Array is read protected. Run-time programming and ROM Dump Mode (with security key verification) are allowed. Auto, slave, and auto PCCB programming are not allowed.
0	1	0	1	Same as above.
1	0	1	1	Array is write protected. ROM dump mode (with security key verification) is allowed. Auto, slave, auto PCCB, and run-time programming are not allowed.
1	0	1	0	Same as above.
0	0	1	1	Array is read and write protected. ROM dump mode (with security key verification) is allowed. Auto, slave, auto PCCB, and run-time programming are not allowed.
0	0	0	0	Same as above.

Figure 41

Other enhancements were also made to the 8X9X for program protection. For example, the value of $\overline{EA}$ is latched on reset so that the device cannot be switched from external to internal execution mode at run-time. In addition, if READ protection is selected, an NMI event will cause the device to switch to external only execution mode. Internal execution can only resume by resetting the chip.

10.7.2 ROM DUMP MODE

You can use the security key and ROM dump mode to dump the internal ROM/EPROM for testing purposes.

The security key is a 16-byte number. The internal ROM/EPROM must contain the security key at locations 2020H–202FH. The user must place the same security key at external address 4020H–402FH. Before doing ROM dump, the device checks that the keys match.

The ROM dump mode is entered by following the power-up sequence described in Section 10.1 with PMODE = 06H. The device first verifies the security keys. If the security keys do not match, the device puts itself into an endless loop of internal execution. If the keys match, the device dumps data to external locations 4000H–5FFFH and 9000H–91FFH on the 879XBH and the 8798 and to external locations 4000H–7FFFH and 9000H–937FH on the 879XJF. The data starting at location 9000H will be indeterminate. The data starting at location 4000H will contain the internal ROM/EPROM, beginning with internal address 2000H.

10.8 Modified Quick-Pulse Programming™ Algorithm

The Modified Quick-Pulse Programming Algorithm calls for each EPROM location to receive 25 separate 100 μs (±5 μs) program cycles. Verification of correct programming is done after the 25 pulses. If the location verifies correctly, the next location is programmed. If the location fails to verify, the location has failed.

Once all locations are programmed and verified, the entire EPROM is again verified.

Programming of 879X devices is done with V_{PP} = 12.75V ±0.25V and V_{CC} = 5.0V ±0.5V.

10.9 Signature Word

The 8X9X contains a signature word at location 2070H. The word can be accessed in the slave mode by executing a word dump command (see Table 2).

Table 2. 8X9XBH Signature Words

Device	Signature Word
879XBH	896FH
839XBH	896EH
809XBH	Undefined
879XJF	896BH
839XJF	896AH
809XJF	Undefined

10.10 Erasing the EPROM

Initially, and after each erasure, all bits of the 879X are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

Erasing begins upon exposure 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 3000–4000 Å range. Constant exposure to room level fluorescent lighting could erase the typical 879X in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 879X is to be exposed to light for extended periods of time, opaque labels must be placed over the EPROM's window to prevent unintentional erasure.

The recommended erasure procedure for the 879X is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15 Wsec/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 879X should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose an 879X can be exposed to without damage is 7258 Wsec/cm^2 (1 week @ 12000 μW/cm^2). Exposure of the 879X to high intensity UV light for long periods may cause permanent damage.

11.0 QUICK REFERENCE

11.1 Pin Description

On the 48-pin devices the following pins are not bonded out: Port1, Port0 (Analog In) bits 0–3, T2CLK (P2.3), T2RST (P2.4), P2.6, P2.7, CLKOUT, INST, NMI, BUSWIDTH. S-DIP packages do not have INST, CLKOUT, BUSWIDTH or NMI.

PIN DESCRIPTIONS

Symbol	Name and Function
V_{CC}	Main supply voltage (5V).
V_{SS}	Digital circuit ground (0V). Two pins.
V_{PD}	RAM standby supply voltage (5V). This voltage must be present during normal operation. In a Power Down condition (i.e. V_{CC} drops to zero), if $\overline{RESET}$ is activated before V_{CC} drops below spec and V_{PD} continues to be held within spec., the top 16 bytes in the Register File will retain their contents. $\overline{RESET}$ must be held low during the Power Down and should not be brought high until V_{CC} is within spec and the oscillator has stabilized. See Section 2.3.
V_{REF}	Reference voltage for the A/D converter (5V). V_{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0. See Section 8.
ANGND	Reference ground for the A/D converter. Should be held at nominally the same potential as V_{SS}. See Section 8.
V_{PP}	Programming voltage for the EPROM devices. It should be + 12.75V when programming and will float to 5V otherwise. The pin should not be above V_{CC} on ROM or CPU devices. This pin must float in the application circuit on EPROM devices.
XTAL1	Input of the oscillator inverter and of the internal clock generator. See Section 1.5.
XTAL2	Output of the oscillator inverter. See Section 1.5.
CLKOUT	Output of the internal clock generator. The frequency of CLKOUT is 1/3 the oscillator frequency. It has a 33% duty cycle. See Section 1.5
$\overline{RESET}$	Reset input to the chip. Input low for at least 10XTAL1 cycles to reset the chip. The subsequent low-to-high transition re-synchronizes CLKOUT and commences a 10-state-time sequence in which the PSW is cleared, a byte read from 2018H loads CCR, and a jump to location 2080H is executed. Input high for normal operation. $\overline{RESET}$ has an internal pullup. See Section 13.
BUSWIDTH	Input for buswidth selection. If CCR bit 1 is a one, this pin selects the bus width for the bus cycle in progress. If BUSWIDTH is a 1, a 16-bit bus cycle occurs. If BUSWIDTH is a 0 an 8-bit cycle occurs. If CCR bit 1 is a 0, the bus is always an 8-bit bus. If this pin is left unconnected, it will rise to V_{CC}. See Section 2.7.
NMI	A positive transition causes a vector to external memory location 0000H. External memory from 00H through 0FFH is reserved for Intel development systems.
INST	Output high during an external memory read indicates the read is an instruction fetch. INST is valid throughout the bus cycle.
$\overline{EA}$	Input for memory select (External Access). $\overline{EA}$ equal to a TTL-high causes memory accesses to locations 2000H through 3FFFH to be directed to on-chip ROM/EPROM. $\overline{EA}$ equal to a TTL-low causes accesses to these locations to be directed to off-chip memory. $\overline{EA}$ = + 12.5V causes execution to begin in the Programming mode on EPROM devices. $\overline{EA}$ has an internal pulldown, so it goes to 0 unless driven otherwise.
ALE/$\overline{ADV}$	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a latch to demultiplex the address from the address/data bus. When the pin is $\overline{ADV}$, it goes inactive high at the end of the bus cycle. $\overline{ADV}$ can be used as a chip select for external memory. ALE/$\overline{ADV}$ is activated only during external memory accesses. See Section 2.7.
$\overline{RD}$	Read signal output to external memory. $\overline{RD}$ is activated only during external memory reads.

PIN DESCRIPTIONS (Continued)

Symbol	Name and Function
$\overline{WR}/\overline{WRL}$	Write and Write Low output to external memory, as selected by the CCR. $\overline{WR}$ will go low for every external write, while $\overline{WRL}$ will go low only for external writes where an even byte is being written. $\overline{WR}/\overline{WRL}$ is activated only during external memory writes. See Section 2.7.
$\overline{BHE}/\overline{WRH}$	Bus High Enable or Write High output to external memory, as selected by the CCR. $\overline{BHE}$ = 0 selects the bank of memory that is connected to the high byte of the data bus. A0 = 0 selects the bank of memory that is connected to the low byte of the data bus. Thus accesses to a 16-bit wide memory can be to the low byte only (A0 = 0, $\overline{BHE}$ = 1), to the high byte only (A0 = 1, BHE # = 0), or both bytes (A0 = 0, $\overline{BHE}$ = 0). If the $\overline{WRH}$ function is selected, the pin will go low if the bus cycle is writing to an odd memory location. See Section 2.7.
READY	Ready input to lengthen external memory cycles, for interfacing to slow or dynamic memory, or for bus sharing. If the pin is high, CPU operation continues in a normal manner. If the pin is low prior to the falling edge of CLKOUT, the Memory Controller goes into a wait mode until the next positive transition in CLKOUT occurs with READY high. The bus cycle can be lengthened by up to 1 μs. When the external memory is not being used, READY has no effect. Internal control of the number of wait states inserted into a bus cycle held not ready is available through configuration of CCR. READY has a weak internal pullup, so it goes to 1 unless externally pulled low. See Section 2.7.
HSI	Inputs to High Speed Input Unit. Four HSI pins are available: HSI.0, HSI.1, HSI.2, and HSI.3. Two of them (HSI.2 and HSI.3) are shared with the HSO Unit. The HSI pins are also used as inputs by EPROM devices in Programming mode. See Section 6.
HSO	Outputs from High Speed Output Unit. Six HSO pins are available: HSO.0, HSO.1, HSO.2, HSO.3, HSO.4, and HSO.5. Two of them (HSO.4 and HSO.5) are shared with the HSI Unit. See Section 7.
Port 0	8-bit high impedance input-only port. These pins can be used as digital inputs and/or as analog inputs to the on-chip A/D converter. These pins are also a mode input to EPROM devices in the Programming mode. See Section 10.
Port 1	8-bit quasi-bidirectional I/O port. See Section 10.
Port 2	8-bit multi-functional port. Six of its pins are shared with other functions in the 8X9X, the remaining 2 are quasi-bidirectional. These pins are also used to input and output control signals on EPROM devices in Programming Mode. See Section 10.
Ports 3 and 4	8-bit bidirectional I/O ports with open drain outputs. These pins are shared with the multiplexed address/data bus which has strong internal pullups. Ports 3 and 4 are also used as a command, address and data path by EPROM devices operating in the programming mode. See Sections 2.7 and 10.

11.2 Pin List

The following is a list of pins in alphabetical order. Where a pin has two names it has been listed under both names, except for the system bus pins, AD0–AD15, which are listed under Port 3 and Port 4.

Name	68-Pin PLCC	68-Pin PGA	48-Pin DIP	64-Pin SDIP
ACH0/P0.0	6	4	—	4
ACH1/P0.1	5	5	—	3
ACH2/P0.2	7	3	—	5
ACH3/P0.3	4	6	—	2
ACH4/P0.4/MOD.0	11	67	43	9
ACH5/P0.5/MOD.1	10	68	42	8
ACH6/P0.6/MOD.2	8	2	40	6
ACH7/P0.7/MOD.3	9	1	41	7
ALE/$\overline{ADV}$	62	16	34	60
ANGND	12	66	44	10
$\overline{BHE}$/$\overline{WRH}$	41	37	15	39
BUSWIDTH	64	14	—	—
CLKOUT	65	13	—	—
$\overline{EA}$	2	8	39	1
EXTINT/P2.2/$\overline{PROG}$	15	63	47	13
HSI.0	24	54	3	22
HSI.1	25	53	4	23
HSI.2/HSO.4	26	52	5	24
HSI.3/HSO.5	27	51	6	25
HSO.0	28	50	7	26
HSO.1	29	49	8	27
HSO.2	34	44	9	32
HSO.3	35	43	10	33
HSO.4/HSI.2	26	52	5	24
HSO.5/HSI.3	27	51	6	25
INST	63	15	—	—
NMI	3	7	—	—
PWM/P2.5/$\overline{PDO}$	39	39	13	37
PALE/P2.1/RXD	17	61	1	15
$\overline{PROG}$/P2.2/EXTNT	15	63	47	13
PVER/P2.0/TXD	18	60	2	16
P0.0/ACH0	6	4	—	4
P0.1/ACH1	5	5	—	3
P0.2/ACH2	7	3	—	5
P0.3/ACH3	4	6	—	2
P0.4/ACH4/MOD.0	11	67	43	9
P0.5/ACH5/MOD.1	10	68	42	8
P0.6/ACH6/MOD.2	8	2	40	6
P0.7/ACH7/MOD.3	9	1	41	7
P1.0	19	59	—	17
P1.1	20	58	—	18
P1.2	21	57	—	19
P1.3	22	56	—	20
P1.4	23	55	—	21
P1.5	30	48	—	28
P1.6	31	47	—	29
P1.7	32	46	—	30
P2.0/TXD/PVER	18	60	2	16
P2.1/RXD/PALE	17	61	1	15
P2.2/EXTINT	15	63	47	13
P2.3/T2CLK	44	34	—	42
P2.4/T2RST	42	36	—	40
P2.5/PWM/$\overline{PDO}$	39	39	13	37
P2.6	33	45	—	31
P2.7	38	40	—	36
P3.0/AD0 PVAL	60	18	32	58
P3.1/AD1 PVAL	59	19	31	57
P3.2/AD2 PVAL	58	20	30	56
P3.3/AD3 PVAL	57	21	29	55
P3.4/AD4 PVAL	56	22	28	54
P3.5/AD5 PVAL	55	23	27	53
P3.6/AD6 PVAL	54	24	26	52
P3.7/AD7 PVAL	53	25	25	51
P4.0/AD8 PVAL	52	26	24	50
P4.1/AD9 PVAL	51	27	23	49
P4.2/AD10 PVAL	50	28	22	48
P4.3/AD11 PVAL	49	29	21	47
P4.4/AD12 PVAL	48	30	20	46
P4.5/AD13 PVAL	47	31	19	45
P4.6/AD14 PVAL	46	32	18	44
P4.7/AD15 PVAL	45	33	17	43
$\overline{RD}$	61	17	33	59
READY	43	35	16	41
$\overline{RESET}$	16	62	48	14
RXD/P2.1	17	61	1	15
SALE/PVER/P2.0	18	60	2	16
$\overline{SPROG}$/$\overline{PDO}$/P2.5	39	39	13	37
TXD/P2.0/SALE	18	60	2	16
T2CLK/P2.3	44	34	—	42
T2RST/P2.4	42	36	—	40
V_{PP}	37	41	12	35
V_{CC}	1	9	38	64
V_{PD}	14	64	46	12
V_{REF}	13	65	45	11
V_{SS}	68	10	11	34
V_{SS}	36	42	37	63
$\overline{WR}$/$\overline{WRL}$	40	38	14	38
$\overline{WRH}$/$\overline{BHE}$	41	37	15	39
XTAL1	67	11	36	62
XTAL2	66	12	35	61

The following pins are not bonded out in the 48-pin package:

P1.0 through P1.7, P0.0 through P0.3, P2.3, P2.4, P2.6, P2.7 CLKOUT, INST, NMI, $\overline{TEST}$, T2CLK (P2.3), T2RST (P2.4).

11.3 Packaging

The MCS-96 products are available in 48-pin, 64-pin and 68-pin packages, with and without A/D, and with and without on-chip ROM or EPROM. The MCS-96 numbering system shown below this section shows the pinouts for the 48- and 68-pin packages. The 48-pin version is offered in a Dual-In-Line package while the 68-pin versions come in a Plastic Leaded Chip Carrier (PLCC), a Pin Grid Array (PGA) or a Type "B" Leadless Chip Carrier.

The MCS®-96 Family Nomenclature

	Factory Masked ROM			CPU			User Programmable					
							EPROM			OTP		
	68-Pin	64-Pin	48-Pin	68-Pin	64-Pin	48-Pin	68-Pin	64-Pin	48-Pin	68-Pin	64-Pin	48-Pin
ANALOG	8397BH 8397JF	8397BH 8397JF	8395BH 8398	8097BH 8097JF	8097BH 8097JF	8095BH 8098	8797BH		8795BH 8798	8797BH 8797JF	8797JF 8797BH	8798
NO ANALOG	8396BH			8X9X								

Transistor Count

Device Type	# MOS Gates
839XBH/879XBH	120,000
809XBH	50,000
839XJF/879XJF	203,00
809XJF	72,000

MTBF Calculations*

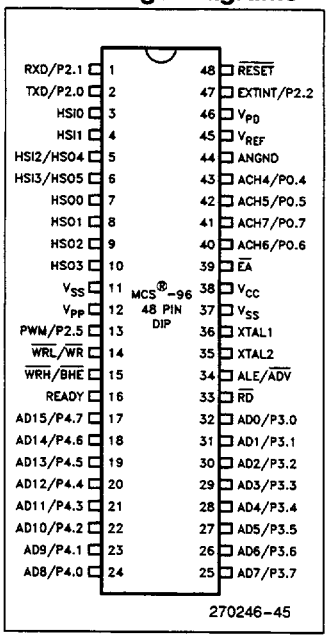
8X9XBH	3.8×10^7 Device Hours @ 55°C
8X9XBH	1.7×10^7 Device Hours @ 70°C
8X9XJF	5.2×10^6 Device Hours @ 55°C

*MTBF data was obtained through calculations based upon the actual average junction temperatures under stress at 55°C and 70°C ambient.

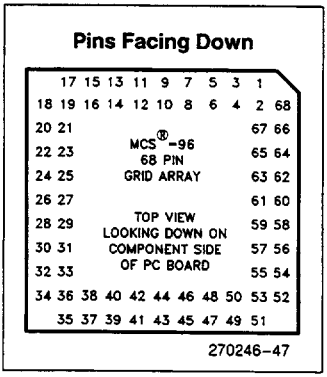
Thermal Characteristics (same for 8X9XBH, 8X9XJF and 8X98)

Package Type	θJa	θJc
PGA	35°C/W	10°C/W
PLCC	37°C/W	13°C/W
LCC	28°C/W	—
Plastic DIP	38°C/W	19°C/W
Ceramic DIP	26°C/W	6.5°C/W

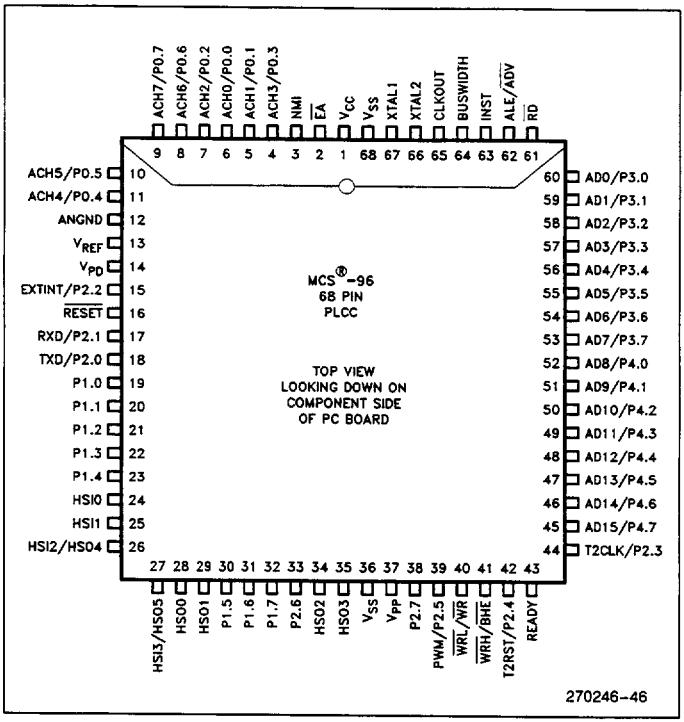
11.4 Package Diagrams



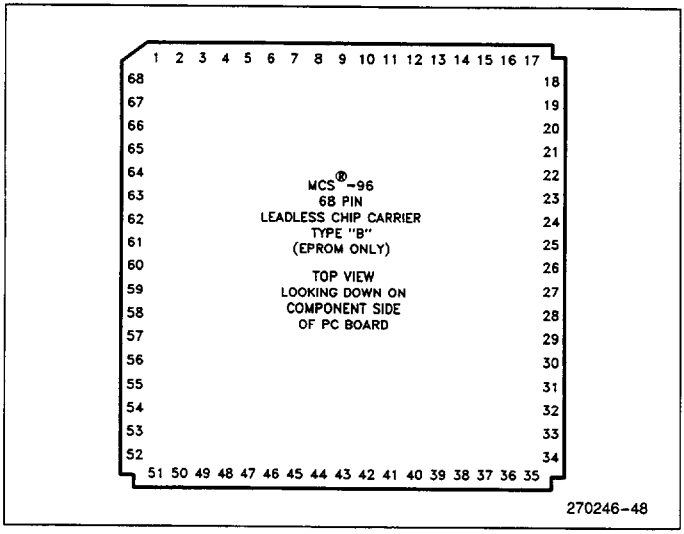
48-Pin Package



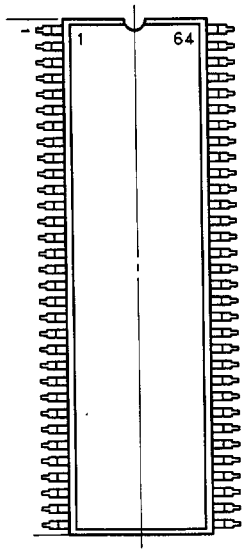
68-Pin Package
(Pin Grid Array - Top View)



68-Pin Package (PLCC - Top View)



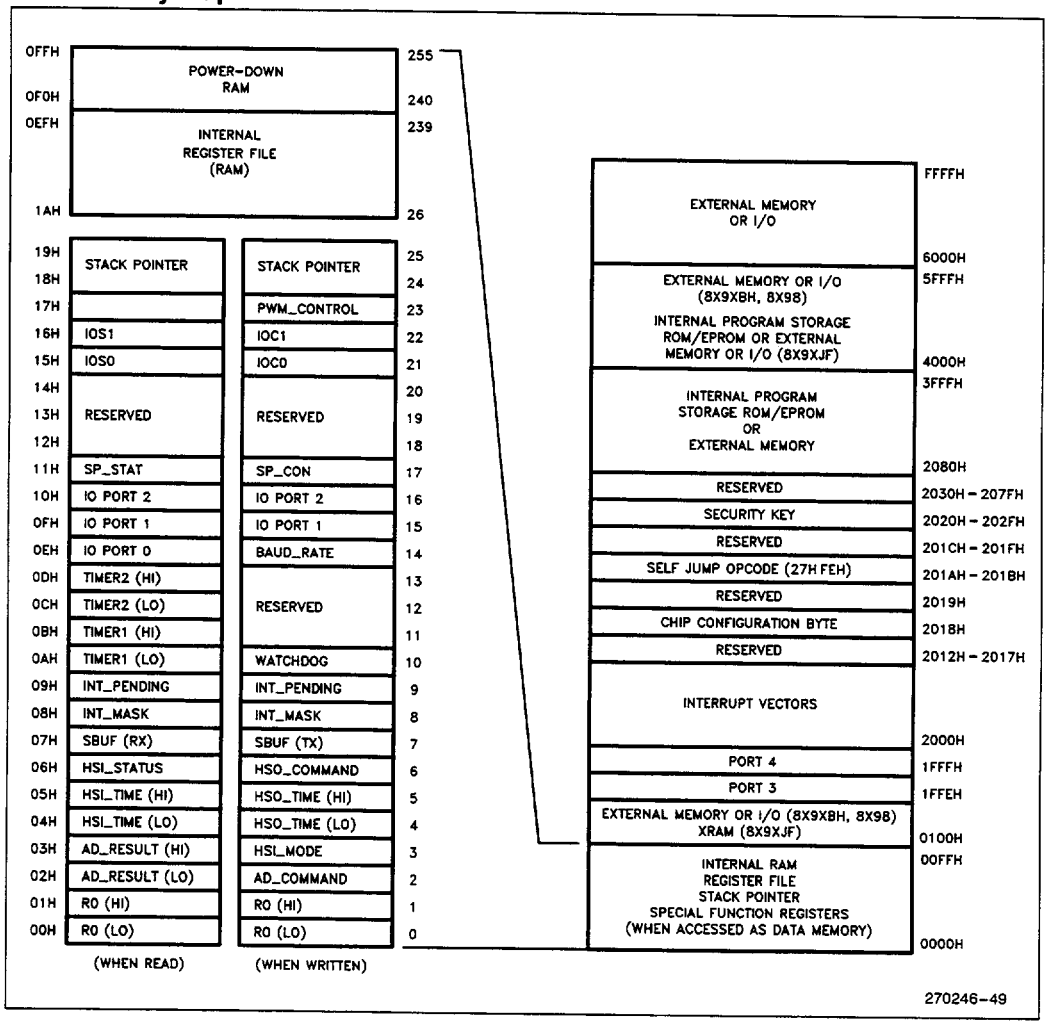
68-Pin Package (LCC - Top View)



270246-65

Shrink-DIP Package

11.5 Memory Map



11.6 Instruction Summary

Mnemonic	Oper-ands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	D ← D + A	✓	✓	✓	✓	↑	—	
ADD/ADDB	3	D ← B + A	✓	✓	✓	✓	↑	—	
ADDC/ADDCB	2	D ← D + A + C	↓	✓	✓	✓	↑	—	
SUB/SUBB	2	D ← D − A	✓	✓	✓	✓	↑	—	
SUB/SUBB	3	D ← B − A	✓	✓	✓	✓	↑	—	
SUBC/SUBCB	2	D ← D − A + C − 1	↓	✓	✓	✓	↑	—	
CMP/CMPB	2	D − A	✓	✓	✓	✓	↑	—	
MUL/MULU	2	D, D + 2 ← D * A	—	—	—	—	—	?	2
MUL/MULU	3	D, D + 2 ← B * A	—	—	—	—	—	?	2
MULB/MULUB	2	D, D + 1 ← D * A	—	—	—	—	—	?	3
MULB/MULUB	3	D, D + 1 ← B * A	—	—	—	—	—	?	3
DIVU	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	✓	↑	—	2
DIVUB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	✓	↑	—	3
DIV	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	?	↑	—	
DIVB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	?	↑	—	
AND/ANDB	2	D ← D and A	✓	✓	0	0	—	—	
AND/ANDB	3	D ← B and A	✓	✓	0	0	—	—	
OR/ORB	2	D ← D or A	✓	✓	0	0	—	—	
XOR/XORB	2	D ← D (excl. or) A	✓	✓	0	0	—	—	
LD/LDB	2	D ← A	—	—	—	—	—	—	
ST/STB	2	A ← D	—	—	—	—	—	—	
LDBSE	2	D ← A; D + 1 ← SIGN(A)	—	—	—	—	—	—	3, 4
LDBZE	2	D ← A; D + 1 ← 0	—	—	—	—	—	—	3, 4
PUSH	1	SP ← SP − 2; (SP) ← A	—	—	—	—	—	—	
POP	1	A ← (SP); SP ← SP + 2	—	—	—	—	—	—	
PUSHF	0	SP ← SP − 2; (SP) ← PSW; PSW ← 0000H | ← 0	0	0	0	0	0	0	
POPF	0	PSW ← (SP); SP ← SP + 2; | ← ✓	✓	✓	✓	✓	✓	✓	
SJMP	1	PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LJMP	1	PC ← PC + 16-bit offset	—	—	—	—	—	—	5
BR [indirect]	1	PC ← (A)	—	—	—	—	—	—	
SCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 16-bit offset	—	—	—	—	—	—	5
RET	0	PC ← (SP); SP ← SP + 2	—	—	—	—	—	—	
J (conditional)	1	PC ← PC + 8-bit offset (if taken)	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D − 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D − 1	↕	↕	↕	↕	↑	—	
NEG/NEGB	1	D ← 0 − D	↕	↕	↕	↕	↑	—	
INC/INCB	1	D ← D + 1	↕	↕	↕	↕	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	↕	↕	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign(D)	↕	↕	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	↕	↕	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb — — — — — — lsb ← 0	↕	?	↕	↕	↑	—	7
SHR/SHRB/SHRL	2	0 → msb — — — — — — lsb → C	↕	?	↕	0	—	↕	7
SHRA/SHRAB/SHRAL	2	msb → msb — — — — — — lsb → C	↕	↕	↕	0	—	↕	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left shift till msb = 1; D ← shift count	↕	?	0	—	—	—	7
TRAP	0	SP ← SP − 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling $\overline{\text{RESET}}$ low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

11.7 Opcode and State Time Listing

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			INDIRECT⑥ NORMAL			INDIRECT⑥ AUTO-INC.		INDEXED⑥ SHORT			INDEXED⑥ LONG	
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE① TIMES	BYTES	STATE① TIMES	OPCODE	BYTES	STATE① TIMES⑧	BYTES	STATE① TIMES⑧
ARITHMETIC INSTRUCTIONS																	
ADD	2	64	3	4	65	4	5	66	3	6/11	3	7/12	67	4	6/11	5	7/12
ADD	3	44	4	5	45	5	6	46	4	7/12	4	8/13	47	5	7/12	6	8/13
ADDB	2	74	3	4	75	3	4	76	3	6/11	3	7/12	77	4	6/11	5	7/12
ADDB	3	54	4	5	55	4	5	56	4	7/12	4	8/13	57	5	7/12	6	8/13
ADDC	2	A4	3	4	A5	4	5	A6	3	6/11	3	7/12	A7	4	6/11	5	7/12
ADDCB	2	B4	3	4	B5	3	4	B6	3	6/11	3	7/12	B7	4	6/11	5	7/12
SUB	2	68	3	4	69	4	5	6A	3	6/11	3	7/12	6B	4	6/11	5	7/12
SUB	3	48	4	5	49	5	6	4A	4	7/12	4	8/13	4B	5	7/12	6	8/13
SUBB	2	78	3	4	79	3	4	7A	3	6/11	3	7/12	7B	4	6/11	5	7/12
SUBB	3	58	4	5	59	4	5	5A	4	7/12	4	8/13	5B	5	7/12	6	8/13
SUBC	2	A8	3	4	A9	4	5	AA	3	6/11	3	7/12	AB	4	6/11	5	7/12
SUBCB	2	B8	3	4	B9	3	4	BA	3	6/11	3	7/12	BB	4	6/11	5	7/12
CMP	2	88	3	4	89	4	5	8A	3	6/11	3	7/12	8B	4	6/11	5	7/12
CMPB	2	98	3	4	99	3	4	9A	3	6/11	3	7/12	9B	4	6/11	5	7/12
MULU	2	6C	3	25	6D	4	26	6E	3	27/32	3	28/33	6F	4	27/32	5	28/33
MULU	3	4C	4	26	4D	5	27	4E	4	28/33	4	29/34	4F	5	28/33	6	29/34
MULUB	2	7C	3	17	7D	3	17	7E	3	19/24	3	20/25	7F	4	19/24	5	20/25
MULUB	3	5C	4	18	5D	4	18	5E	4	20/25	4	21/26	5F	5	20/25	6	21/26
MUL	2	②	4	29	②	5	30	②	4	31/36	4	32/37	②	5	31/36	6	32/37
MUL	3	②	5	30	②	6	31	②	5	32/37	5	33/38	②	6	32/37	7	33/38
MULB	2	②	4	21	②	4	21	②	4	23/28	4	24/29	②	5	23/28	6	24/29
MULB	3	②	5	22	②	5	22	②	5	24/29	5	25/30	②	6	24/29	7	25/30
DIVU	2	8C	3	25	8D	4	26	8E	3	28/32	3	29/33	8F	4	28/32	5	29/33
DIVUB	2	9C	3	17	9D	3	17	9E	3	20/24	3	21/25	9F	4	20/24	5	21/25
DIV	2	②	4	29	②	5	30	②	4	32/36	4	33/37	②	5	32/36	6	33/37
DIVB	2	②	4	21	②	4	21	②	4	24/28	4	25/29	②	5	24/28	6	25/29

270246-63

NOTES:

*Long indexed and Indirect + instructions have identical opcodes with Short indexed and Indirect modes, respectively. The second byte of instructions using any Indirect or indexed addressing mode specifies the exact mode used. If the second byte is even, use Indirect or Short indexed. If it is odd, use Indirect + or Long indexed. In all cases the second byte of the instruction always specifies an even (word) location for the address referenced.

① Number of state times shown for internal/external operands.

② The opcodes for signed multiply and divide are the opcodes for the unsigned functions with an "FE" appended as a prefix.

⑧ State times shown for 16-bit bus.

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			INDIRECT⑧					INDEXED⑧				
								NORMAL			AUTO-INC.		SHORT			LONG	
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE① TIMES	BYTES	STATE① TIMES	OPCODE	BYTES	STATE① TIMES⑥	BYTES	STATE① TIMES⑥
LOGICAL INSTRUCTIONS																	
AND	2	60	3	4	61	4	5	62	3	6/11	3	7/12	63	4	6/11	5	7/12
AND	3	40	4	5	41	5	6	42	4	7/12	4	8/13	43	5	7/12	6	8/13
ANDB	2	70	3	4	71	3	4	72	3	6/11	3	7/12	73	4	6/11	5	7/12
ANDB	3	50	4	5	51	4	5	52	4	7/12	4	8/13	53	5	7/12	6	8/13
OR	2	80	3	4	81	4	5	82	3	6/11	3	7/12	83	4	6/11	5	7/12
ORB	2	90	3	4	91	3	4	92	3	6/11	3	7/12	93	4	6/11	5	7/12
XOR	2	84	3	4	85	4	5	86	3	6/11	3	7/12	87	4	6/11	5	7/12
XORB	2	94	3	4	95	3	4	96	3	6/11	3	7/12	97	4	6/11	5	7/12
DATA TRANSFER INSTRUCTIONS																	
LD	2	A0	3	4	A1	4	5	A2	3	6/11	3	7/12	A3	4	6/11	5	7/12
LDB	2	B0	3	4	B1	3	4	B2	3	6/11	3	7/12	B3	4	6/11	5	7/12
ST	2	C0	3	4	—	—	—	C2	3	7/11	3	8/12	C3	4	7/11	5	8/12
STB	2	C4	3	4	—	—	—	C6	3	7/11	3	8/12	C7	4	7/11	5	8/12
LDBSE	2	BC	3	4	BD	3	4	BE	3	6/11	3	7/12	BF	4	6/11	5	7/12
LDBZE	2	AC	3	4	AD	3	4	AE	3	6/11	3	7/12	AF	4	6/11	5	7/12
STACK OPERATIONS (internal stack)																	
PUSH	1	C8	2	8	C9	3	8	CA	2	11/15	2	12/16	CB	3	11/15	4	12/16
POP	1	CC	2	12	—	—	—	CE	2	14/18	2	14/18	CF	3	14/18	4	14/18
PUSHF	0	F2	1	8													
POPF	0	F3	1	9													
STACK OPERATIONS (external stack)																	
PUSH	1	C8	2	12	C9	3	12	CA	2	15/19	2	16/20	CB	3	15/19	4	16/20
POP	1	CC	2	14	—	—	—	CE	2	16/20	2	16/20	CF	3	16/20	4	16/20
PUSHF	0	F2	1	12													
POPF	0	F3	1	13													

JUMPS AND CALLS							
MNEMONIC	OPCODE	BYTES	STATES	MNEMONIC	OPCODE	BYTES	STATES
LJMP	E7	3	8	LCALL	EF	3	13/16⑤
SJMP	20-27④	2	8	SCALL	28-2F④	2	13/16⑤
BR[ ]	E3	2	8	RET	F0	1	12/16⑤
				TRAP③	F7	1	21/24

270246-64

NOTES:
① Number of state times shown for internal/external operands.
③ The assembler does not accept this mnemonic.
④ The least significant 3 bits of the opcode are concatenated with the following 8 bits to form an 11-bit, 2's complement, offset for the relative call or jump.
⑤ State times for stack located internal/external.
⑥ State times shown for 16-bit bus.

CONDITIONAL JUMPS

All conditional jumps are 2 byte instructions. They require 8 state times if the jump is taken, 4 if it is not.(8)

MNEMONIC	OPCODE	MNEMONIC	OPCODE	MNEMONIC	OPCODE	MNEMONIC	OPCODE
JC	DB	JE	DF	JGE	D6	JGT	D2
JNC	D3	JNE	D7	JLT	DE	JLE	DA
JH	D9	JV	DD	JVT	DC	JST	D8
JNH	D1	JNV	D5	JNVT	D4	JNST	D0

JUMP ON BIT CLEAR OR BIT SET

These instructions are 3 byte instructions. They require 9 state times if the jump is taken, 5 if it is not.(8)

	BIT NUMBER							
MNEMONIC	0	1	2	3	4	5	6	7
JBC	30	31	32	33	34	35	36	37
JBS	38	39	3A	3B	3C	3D	3E	3F

LOOP CONTROL

MNEMONIC	OPCODE	BYTES	STATE TIMES
DJNZ	EO	3	5/9 STATE TIME (NOT TAKEN/TAKEN)(8)

SINGLE REGISTER INSTRUCTIONS

MNEMONIC	OPCODE	BYTES	STATES(8)	MNEMONIC	OPCODE	BYTES	STATES(8)
DEC	05	2	4	EXT	06	2	4
DECB	15	2	4	EXTB	16	2	4
NEG	03	2	4	NOT	02	2	4
NEGB	13	2	4	NOTB	12	2	4
INC	07	2	4	CLR	01	2	4
INCB	17	2	4	CLRB	11	2	4

SHIFT INSTRUCTIONS

INSTR MNEMONIC	WORD		INSTR MNEMONIC	BYTE		INSTR MNEMONIC	DBL WD		STATE TIMES(8)
	OP	B		OP	B		OP	B	
SHL	09	3	SHLB	19	3	SHLL	0D	3	7 + 1 PER SHIFT(7)
SHR	08	3	SHRB	18	3	SHRL	0C	3	7 + 1 PER SHIFT(7)
SHRA	0A	3	SHRAB	1A	3	SHRAL	0E	3	7 + 1 PER SHIFT(7)

SPECIAL CONTROL INSTRUCTIONS

MNEMONIC	OPCODE	BYTES	STATES(8)	MNEMONIC	OPCODE	BYTES	STATES(8)
SETC	F9	1	4	DI	FA	1	4
CLRC	F8	1	4	EI	FB	1	4
CLRVT	FC	1	4	NOP	FD	1	4
RST(6)	FF	1	166	SKIP	00	2	4

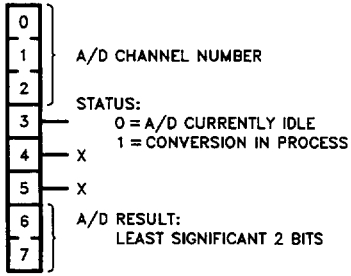
NORMALIZE

MNEMONIC	OPCODE	BYTES	STATE TIMES
NORML	0F	3	11 + 1 PER SHIFT

NOTES:
6. This instruction takes 2 states to pull $\overline{\text{RESET}}$ low, then holds it low for at least one state time to initiate a reset. The reset takes 13 states, at which time the program restarts at location 2080H.
7. Execution will take at least 8 states, even for 0 shift.
8. State times shown for 16-bit bus.

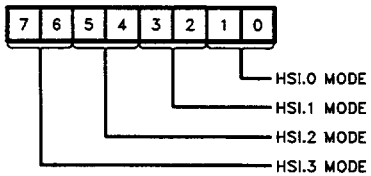
11.8 SFR Summary

A/D Result LO (02H)



270246-50

HSI_Mode (03H)

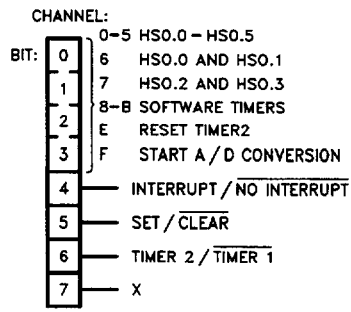


WHERE EACH 2-BIT MODE CONTROL FIELD DEFINES ONE OF 4 POSSIBLE MODES:

00 8 POSITIVE TRANSITIONS
01 EACH POSITIVE TRANSITION
10 EACH NEGATIVE TRANSITION
11 EVERY TRANSITION (POSITIVE AND NEGATIVE)

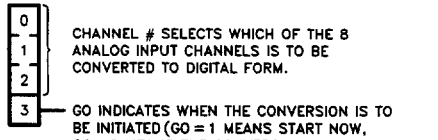
270246-51

HSO Command (06H)



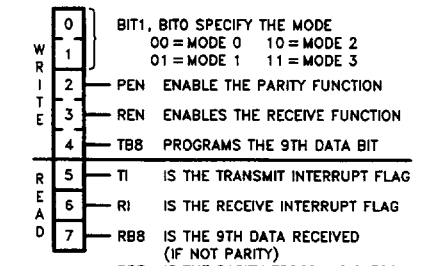
270246-52

A/D Command (02H)



270246-53

SPCON/SPSTAT (11H)



270246-54

Baud Rate Calculations

Using XTAL1:

$$\text{Mode 0: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{XTAL1 frequency}}{4*(B+1)};\ B \neq 0$$

$$\text{Others: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{XTAL1 frequency}}{64*(B+1)}$$

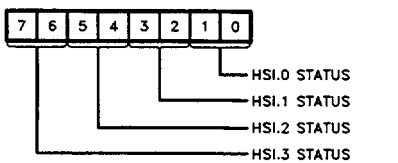
Using T2CLK:

$$\text{Mode 0: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{T2CLK frequency}}{B};\ B \neq 0$$

$$\text{Others: } \frac{\text{Baud}}{\text{Rate}} = \frac{\text{T2CLK frequency}}{16*B};\ B \neq 0$$

Note that B cannot equal 0, except when using XTAL1 in other than Mode 0.

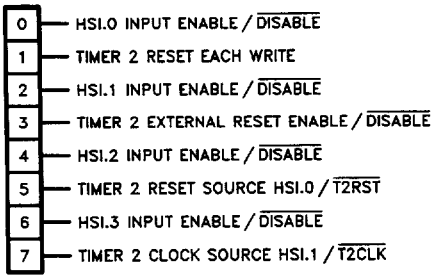
HSI_Status (06H)



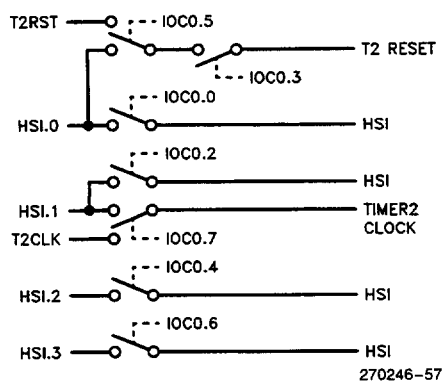
WHERE FOR EACH 2-BIT STATUS FIELD THE LOWER BIT INDICATES WHETHER OR NOT AN EVENT HAS OCCURED ON THIS PIN AND THE UPPER BIT INDICATES THE CURRENT STATUS OF THE PIN.

270246-55

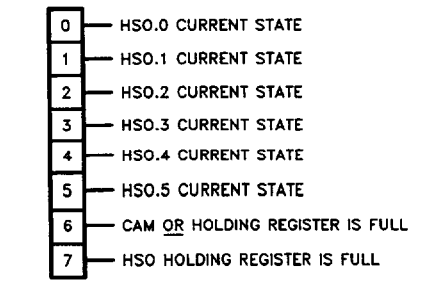
IOC0 (15H)



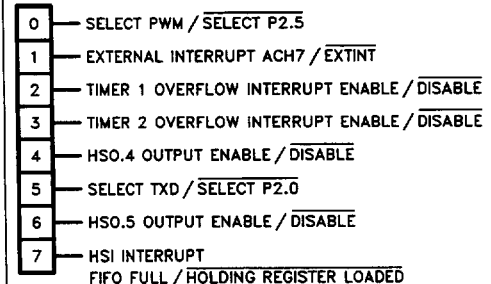
IOC0 (15H)



IOS0 (15H)

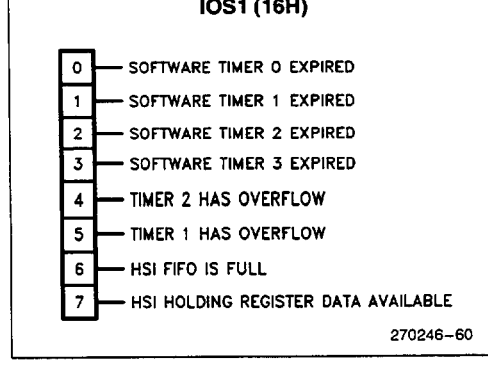


IOC1 (16H)

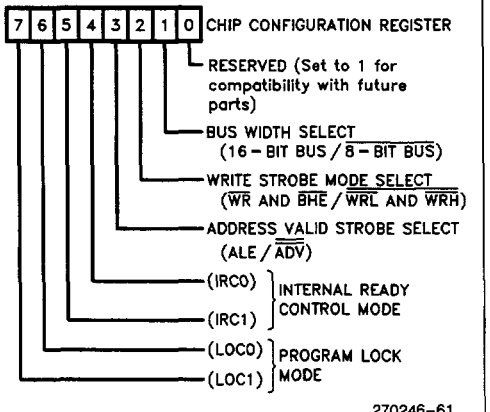


Vector	Vector Location (High Byte)	Vector Location (Low Byte)	Priority
Software Trap	2011H	2010H	Not Applicable
Extint	200FH	200EH	7 (Highest)
Serial Port	200DH	200CH	6
Software Timers	200BH	200AH	5
HSI.0	2009H	2008H	4
High Speed Outputs	2007H	2006H	3
HSI Data Available	2005H	2004H	2
A/D Conversion Complete	2003H	2002H	1
Timer Overflow	2001H	2000H	0 (Lowest)

IOS1 (16H)



Chip Configuration



Internal Ready Control

IRC1	IRC0	Description
0	0	Limit to 1 Wait State
0	1	Limit to 2 Wait States
1	0	Limit to 3 Wait States
1	1	Disable Internal Ready Control

Program Lock Modes

LOC1	LOC0	Protection
0	0	Read and Write Protected
0	1	Read Protected
1	0	Write Protected
1	1	No Protection

Programming Function PMODE Values

PMODE	Programming Mode
0–4	Reserved
5	Slave Programming
6–0BH	Reserved
0CH	Auto Programming Mode
0DH	Program Configuration Byte
0EH–0FH	Reserved

Slave Programming Mode Commands

P4.7	P4.6	Action
0	0	Word Dump
0	1	Data Verify
1	0	Data Program
1	1	Reserved

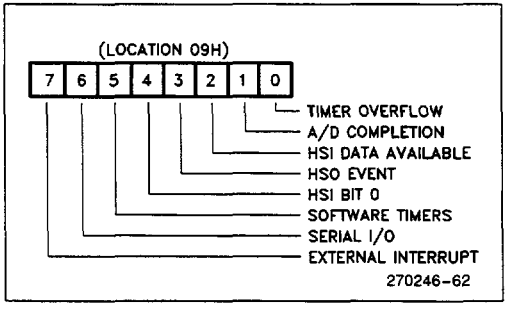
8X9XBH Signature Word

Device	Signature Word
879XBH	896FH
839XBH	896EH
809XBH	Undefined

Port 2 Pin Functions

Port	Function	Alternate Function
P2.0	Output	TXD (Serial Port Transmit)
P2.1	Input	RXD (Serial Port Receive)
P2.2	Input	EXTINT (External Interrupt)
P2.3	Input	T2CLK (Timer 2 Clock)
P2.4	Input	T2RST (Timer 2 Reset)
P2.5	Output	PWM (Pulse Width Modulation)

Interrupt Pending Register



PSW Register

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Z	N	V	VT	C	—	I	ST	<Interrupt Mask Reg>							

8X9X Quick Reference

3

intel®

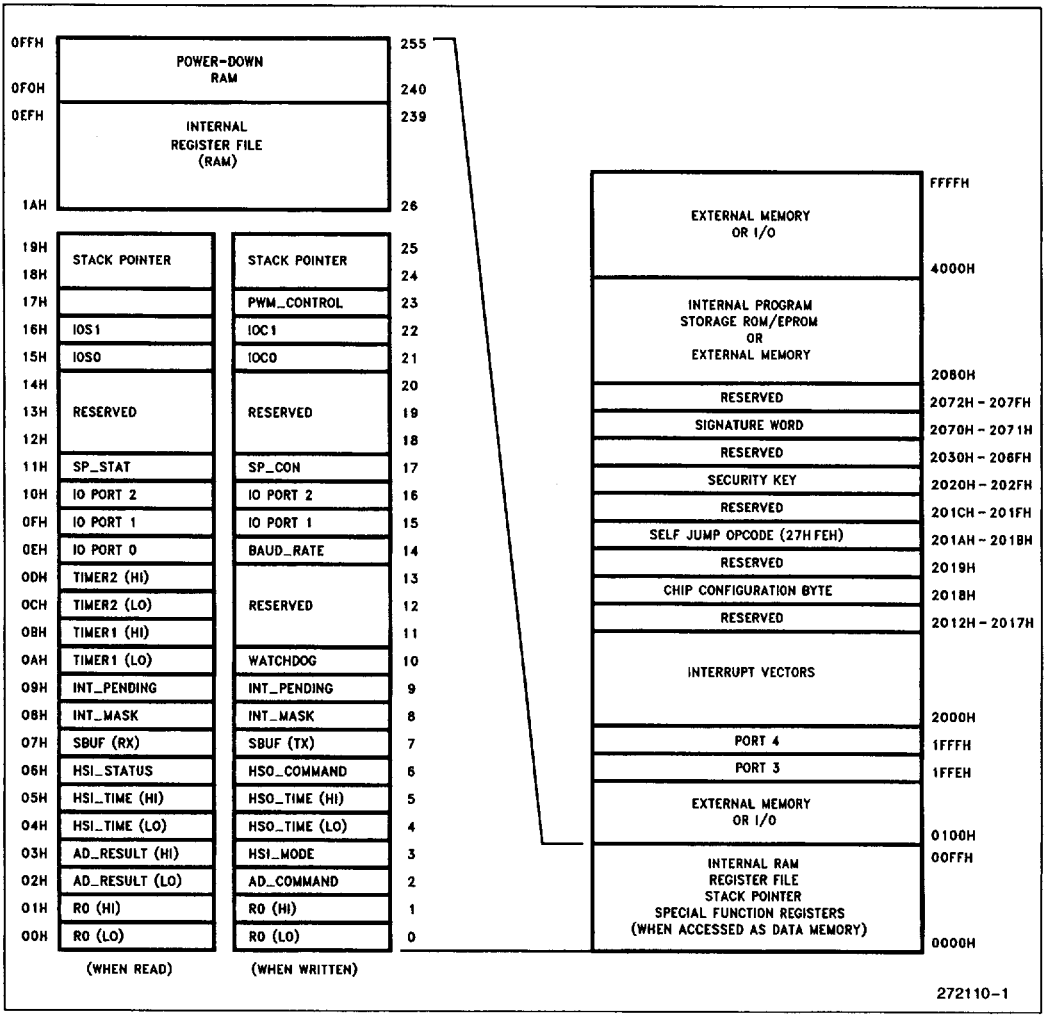
8X9X
Quick Reference

August 1992

Order Number: 272110-002

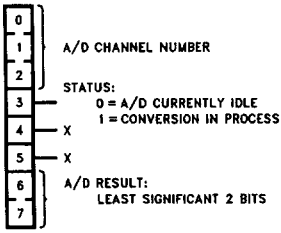
8X9X QUICK REFERENCE

1.0 MEMORY AND SFR MAP



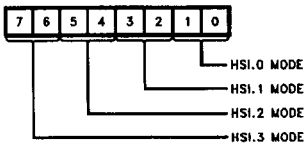
2.0 SFR BIT SUMMARY

A/D Result LO (02H)



272110-2

HSI_Mode (03H)

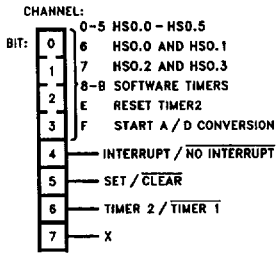


WHERE EACH 2-BIT MODE CONTROL FIELD DEFINES ONE OF 4 POSSIBLE MODES:

00 8 POSITIVE TRANSITIONS
01 EACH POSITIVE TRANSITION
10 EACH NEGATIVE TRANSITION
11 EVERY TRANSITION (POSITIVE AND NEGATIVE)

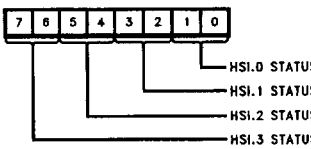
272110-3

HSO Command (06H)



272110-4

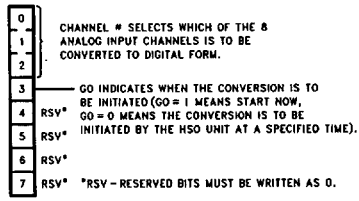
HSI_Status (06H)



WHERE FOR EACH 2-BIT STATUS FIELD THE LOWER BIT INDICATES WHETHER OR NOT AN EVENT HAS OCCURRED ON THIS PIN AND THE UPPER BIT INDICATES THE CURRENT STATUS OF THE PIN.

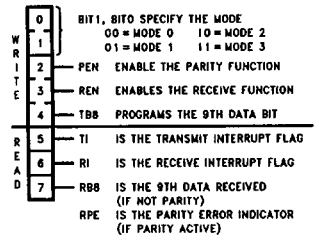
272110-5

A/D Command (02H)



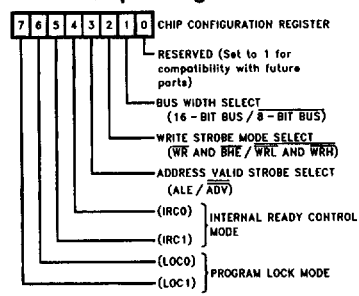
272110-6

SPCON/SPSTAT (11H)



272110-7

Chip Configuration



272110-8

Internal Ready Control

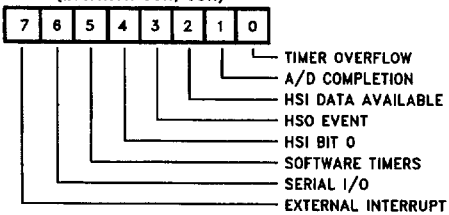
IRC1	IRC0	Description
0	0	Limit to 1 Wait State
0	1	Limit to 2 Wait States
1	0	Limit to 3 Wait States
1	1	Disable Internal Ready Control

Program Lock Modes

LOC1	LOC0	Protection
0	0	Read and Write Protected
0	1	Read Protected
1	0	Write Protected
1	1	No Protection

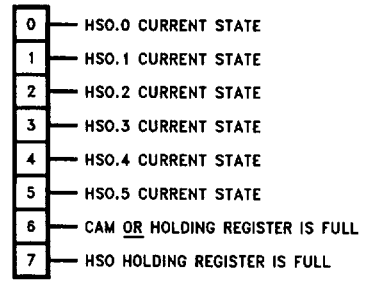
Interrupt Pending/Mask Register

(LOCATION 08H, 09H)



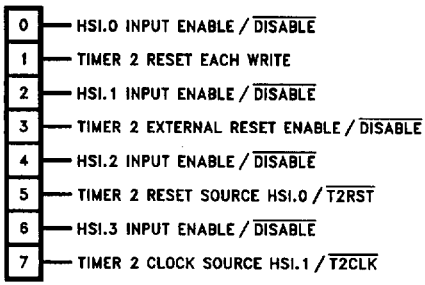
272110-9

IOS0 (15H)



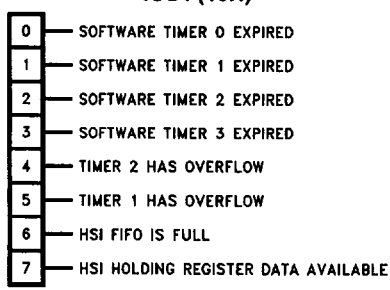
272110-12

IOC0 (15H)



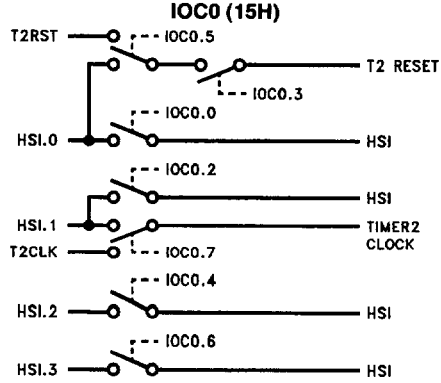
272110-10

IOS1 (16H)



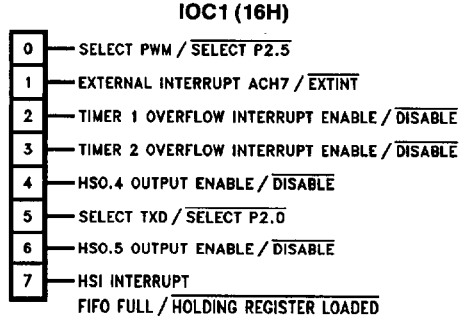
272110-13

IOC0 (15H)



272110-11

IOC1 (16H)



272110-14

PSW Register

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Z	N	V	VT	C	—	I	ST	<Interrupt Mask Reg>							

3.0 PIN DEFINITION TABLE

Pin Name	68L PLCC	68L PGA or LCC	64L SDIP	48L DIP
ACH0	6	4	4	
ACH1	5	5	3	
ACH2	7	3	5	
ACH3	4	6	2	
ACH4	11	67	9	43
ACH5	10	68	8	42
ACH6	8	2	6	40
ACH7	9	1	7	41
AD0	60	18	58	32
AD1	59	19	57	31
AD2	58	20	56	30
AD3	57	21	55	29
AD4	56	22	54	28
AD5	55	23	53	27
AD6	54	24	52	26
AD7	53	25	51	25
AD8	52	26	50	24
AD9	51	27	49	23
AD10	50	28	48	22
AD11	49	29	47	21
AD12	48	30	46	20
AD13	47	31	45	19
AD14	46	32	44	18
AD15	45	33	43	17
$\overline{ADV}$	62	16	60	34
ALE	62	16	60	34
ANGND	12	66	10	44
$\overline{BHE}$	41	37	39	15
BUSWIDTH	64	14		
CLKOUT	65	13		
$\overline{EA}$	2	8	1	39
EXTINT	15	63	13	47
HSI.0	24	54	22	3
HSI.1	25	53	23	4
HSI.2	26	52	24	5
HSI.3	27	51	25	6

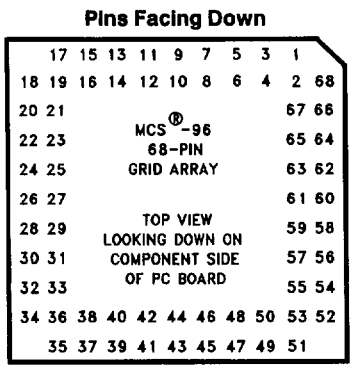
Pin Name	68L PLCC	68L PGA or LCC	64L SDIP	48L DIP
HSO.0	28	50	26	7
HSO.1	29	49	27	8
HSO.2	34	44	32	9
HSO.3	35	43	33	10
HSO.4	26	52	24	5
HSO.5	27	51	25	6
INST	63	15		
NMI	3	7		
P0.0	6	4	4	
P0.1	5	5	3	
P0.2	7	3	5	
P0.3	4	6	2	
P0.4	11	67	9	43
P0.5	10	68	8	42
P0.6	8	2	6	40
P0.7	9	1	7	41
P1.0	19	59	17	
P1.1	20	58	18	
P1.2	21	57	19	
P1.3	22	56	20	
P1.4	23	55	21	
P1.5	30	48	28	
P1.6	31	47	29	
P1.7	32	46	30	
P2.0	18	60	16	2
P2.1	17	61	15	1
P2.2	15	63	13	47
P2.3	44	34	42	
P2.4	42	36	40	
P2.5	39	39	37	13
P2.6	33	45	31	
P2.7	38	40	36	
P3.0	60	18	58	32
P3.1	59	19	57	31
P3.2	58	20	56	30
P3.3	57	21	55	29

3.0 PIN DEFINITION TABLE (Continued)

Pin Name	68L PLCC	68L PGA or LCC	64L SDIP	48L DIP
P3.4	56	22	54	28
P3.5	55	23	53	27
P3.6	54	24	52	26
P3.7	53	25	51	25
P4.0	52	26	50	24
P4.1	51	27	49	23
P4.2	50	28	48	22
P4.3	49	29	47	21
P4.4	48	30	46	20
P4.5	47	31	45	19
P4.6	46	32	44	18
P4.7	45	33	43	17
PALE	17	61	15	1
$\overline{PDO}$	39	39	37	13
PMOD.0	11	67	9	43
PMOD.1	10	68	8	42
PMOD.2	8	2	6	40
PMOD.3	9	1	7	41
$\overline{PROG}$	15	63	13	47
PVER	18	60	16	2
PWM	39	39	37	13

Pin Name	68L PLCC	68L PGA or LCC	64L SDIP	48L DIP
$\overline{RD}$	61	17	59	33
READY	43	35	41	16
$\overline{RESET}$	16	62	14	48
RXD	17	61	15	1
SALE	18	60	16	2
$\overline{SPROG}$	39	39	37	13
T2CLK	44	34	42	
T2RST	42	36	40	
TXD	18	60	16	2
V_{CC}	1	9	64	38
V_{PD}	14	64	12	46
V_{PP}	37	41	35	12
V_{REF}	13	65	11	45
V_{SS1}	68	10	34	11
V_{SS2}	36	42	63	37
$\overline{WR}$	40	38	38	14
$\overline{WRL}$	40	38	38	14
$\overline{WRH}$	41	37	39	25
XTAL1	67	11	62	36
XTAL2	66	12	61	35

4.0 PACKAGE PIN ASSIGNMENT

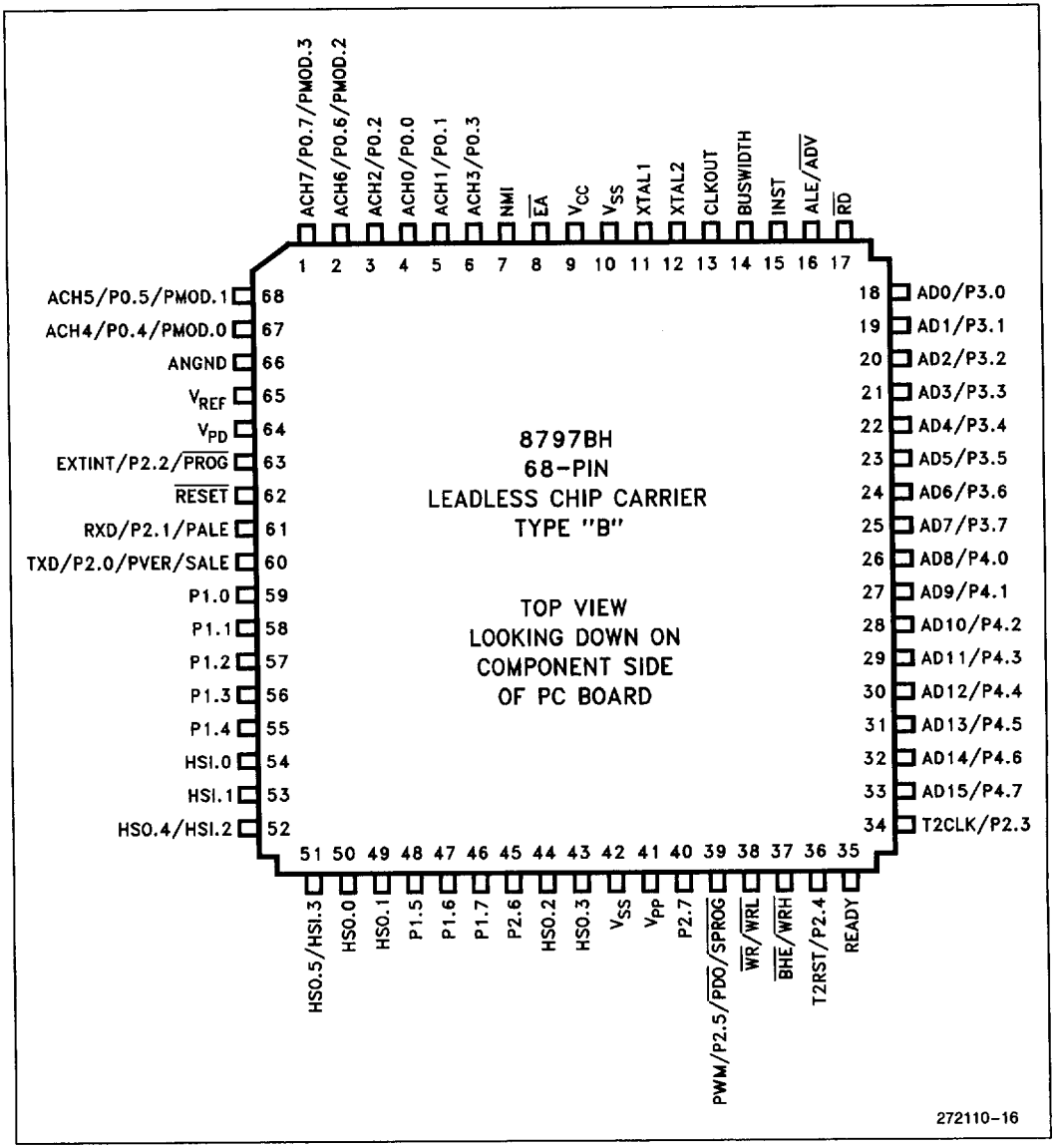


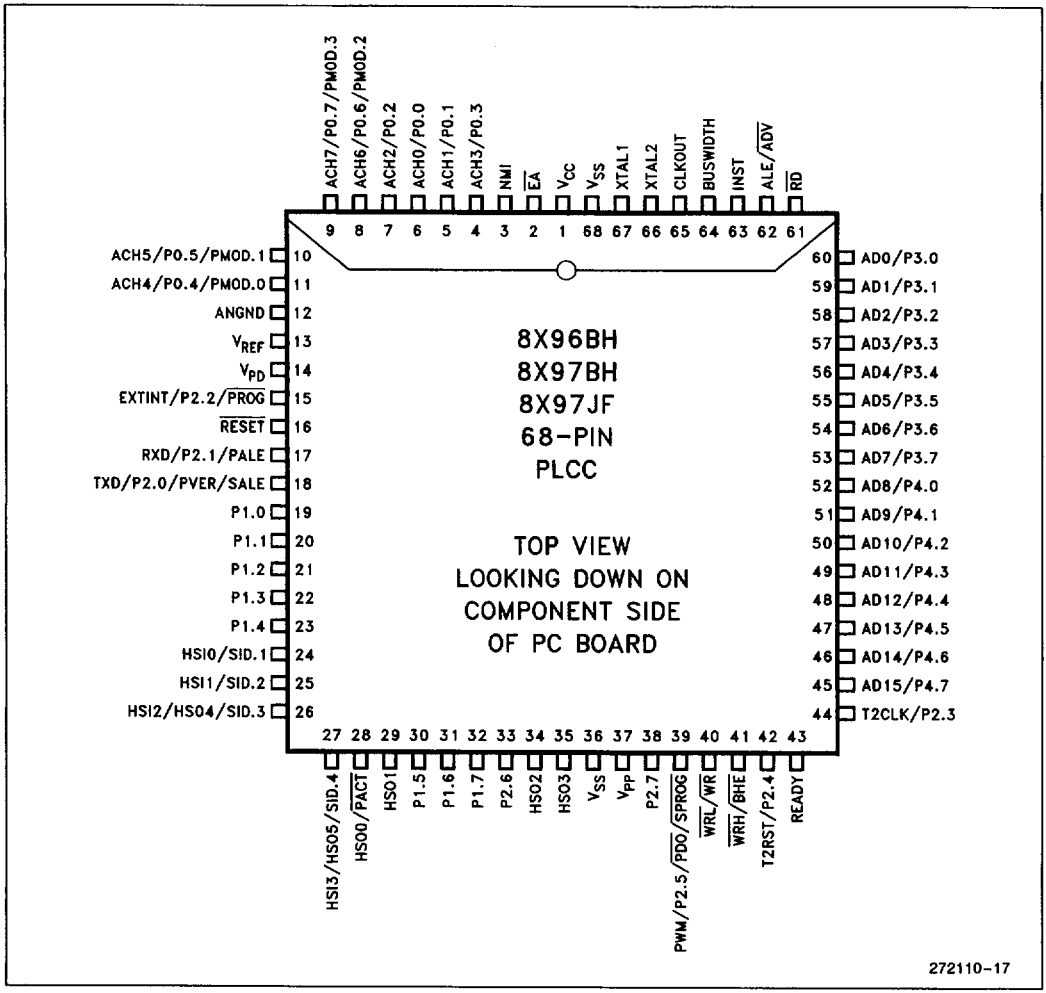
272110-15

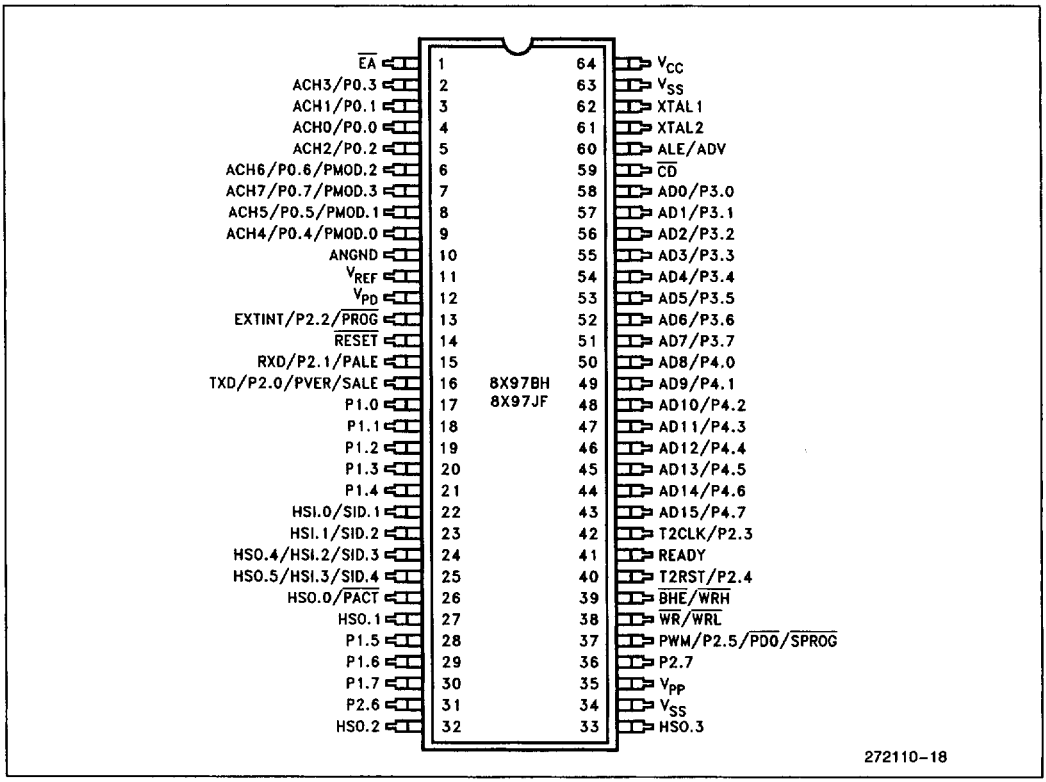
PGA	Description
1	ACH7/P0.7/PMOD.3
2	ACH6/P0.6/PMOD.2
3	ACH2/P0.2
4	ACH0/P0.0
5	ACH1/P0.1
6	ACH3/P0.3
7	NMI
8	$\overline{EA}$
9	V_{CC}
10	V_{SS}
11	XTAL1
12	XTAL2
13	CLKOUT
14	BUSWIDTH
15	INST
16	ALE/$\overline{ADV}$
17	$\overline{RD}$
18	AD0/P3.0
19	AD1/P3.1
20	AD2/P3.2
21	AD3/P3.3
22	AD4/P3.4
23	AD5/P3.5

PGA	Description
24	AD6/P3.6
25	AD7/P3.7
26	AD8/P4.0
27	AD9/P4.1
28	AD10/P4.2
29	AD11/P4.3
30	AD12/P4.4
31	AD13/P4.5
32	AD14/P4.6
33	AD15/P4.7
34	T2CLK/P2.3
35	READY
36	T2RST/P2.4
37	$\overline{BHE}$/$\overline{WRH}$
38	$\overline{WR}$/$\overline{WRL}$
39	PWM/P2.5/$\overline{PDO}$/$\overline{SPROG}$
40	P2.7
41	V_{PP}
42	V_{SS}
43	HSO.3
44	HSO.2
45	P2.6
46	P1.7

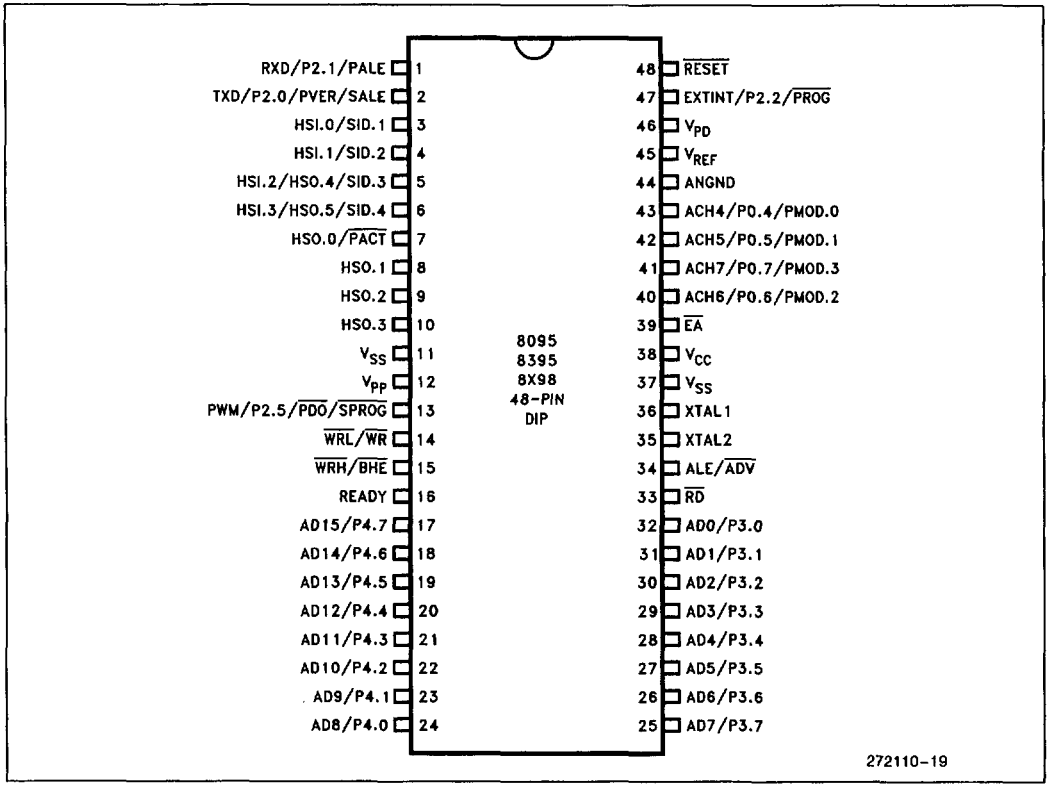
PGA	Description
47	P1.6
48	P1.5
49	HSO.1
50	HSO.0
51	HSO.5/HSI.3
52	HSO.4/HSI.2
53	HSI.1
54	HSI.0
55	P1.4
56	P1.3
57	P1.2
58	P1.1
59	P1.0
60	TXD/P2.0/PVER/SALE
61	RXD/P2.1/PALE
62	$\overline{RESET}$
63	EXTINT/P2.2/$\overline{PROG}$
64	V_{PD}
65	V_{REF}
66	ANGND
67	ACH4/P0.4/PMOD.0
68	ACH5/P0.5/PMOD.1


8797BH
68-PIN
LEADLESS CHIP CARRIER
TYPE "B"
TOP VIEW
LOOKING DOWN ON
COMPONENT SIDE
OF PC BOARD
1 ACH7/P0.7/PMOD.3
2 ACH6/P0.6/PMOD.2
3 ACH2/P0.2
4 ACH0/P0.0
5 ACH1/P0.1
6 ACH3/P0.3
7 NMI
8 EA
9 VCC
10 VSS
11 XTAL1
12 XTAL2
13 CLKOUT
14 BUSWIDTH
15 INST
16 ALE/ADV
17 RD
18 AD0/P3.0
19 AD1/P3.1
20 AD2/P3.2
21 AD3/P3.3
22 AD4/P3.4
23 AD5/P3.5
24 AD6/P3.6
25 AD7/P3.7
26 AD8/P4.0
27 AD9/P4.1
28 AD10/P4.2
29 AD11/P4.3
30 AD12/P4.4
31 AD13/P4.5
32 AD14/P4.6
33 AD15/P4.7
34 T2CLK/P2.3
35 READY
36 T2RST/P2.4
37 BHE/WRH
38 WR/WRL
39 PWM/P2.5/PDO/SPROG
40 P2.7
41 VPP
42 VSS
43 HSO.3
44 HSO.2
45 P2.6
46 P1.7
47 P1.6
48 P1.5
49 HSO.1
50 HSO.0
51 HSO.5/HSI.3
52 HSO.4/HSI.2
53 HSI.1
54 HSI.0
55 P1.4
56 P1.3
57 P1.2
58 P1.1
59 P1.0
60 TXD/P2.0/PVER/SALE
61 RXD/P2.1/PALE
62 RESET
63 EXTINT/P2.2/PROG
64 VPD
65 VREF
66 ANGND
67 ACH4/P0.4/PMOD.0
68 ACH5/P0.5/PMOD.1
272110-16


ACH7/P0.7/PMOD.3
ACH6/P0.6/PMOD.2
ACH2/P0.2
ACH0/P0.0
ACH1/P0.1
ACH3/P0.3
NMI
EA
VCC
VSS
XTAL1
XTAL2
CLKOUT
BUSWIDTH
INST
ALE/ADV
RD
9 8 7 6 5 4 3 2 1 68 67 66 65 64 63 62 61
ACH5/P0.5/PMOD.1 10
ACH4/P0.4/PMOD.0 11
ANGND 12
VREF 13
VPD 14
EXTINT/P2.2/PROG 15
RESET 16
RXD/P2.1/PALE 17
TXD/P2.0/PVER/SALE 18
P1.0 19
P1.1 20
P1.2 21
P1.3 22
P1.4 23
HSI0/SID.1 24
HSI1/SID.2 25
HSI2/HS04/SID.3 26
60 AD0/P3.0
59 AD1/P3.1
58 AD2/P3.2
57 AD3/P3.3
56 AD4/P3.4
55 AD5/P3.5
54 AD6/P3.6
53 AD7/P3.7
52 AD8/P4.0
51 AD9/P4.1
50 AD10/P4.2
49 AD11/P4.3
48 AD12/P4.4
47 AD13/P4.5
46 AD14/P4.6
45 AD15/P4.7
44 T2CLK/P2.3
8X96BH
8X97BH
8X97JF
68-PIN
PLCC
TOP VIEW
LOOKING DOWN ON
COMPONENT SIDE
OF PC BOARD
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
HSI3/HS05/SID.4
HS00/PACT
HSO1
P1.5
P1.6
P1.7
P2.6
HSO2
HSO3
VSS
VPP
P2.7
PWM/P2.5/PDO/SPROG
WRL/WR
WRH/BHE
T2RST/P2.4
READY
272110-17



Shrink-DIP Package


RXD/P2.1/PALE 1
TXD/P2.0/PVER/SALE 2
HSI.0/SID.1 3
HSI.1/SID.2 4
HSI.2/HSO.4/SID.3 5
HSI.3/HSO.5/SID.4 6
HSO.0/PACT 7
HSO.1 8
HSO.2 9
HSO.3 10
VSS 11
VPP 12
PWM/P2.5/PDO/SPROG 13
WRL/WR 14
WRH/BHE 15
READY 16
AD15/P4.7 17
AD14/P4.6 18
AD13/P4.5 19
AD12/P4.4 20
AD11/P4.3 21
AD10/P4.2 22
AD9/P4.1 23
AD8/P4.0 24
8095
8395
8X98
48-PIN
DIP
48 RESET
47 EXTINT/P2.2/PROG
46 VPD
45 VREF
44 ANGND
43 ACH4/P0.4/PMOD.0
42 ACH5/P0.5/PMOD.1
41 ACH7/P0.7/PMOD.3
40 ACH6/P0.6/PMOD.2
39 EA
38 VCC
37 VSS
36 XTAL1
35 XTAL2
34 ALE/ADV
33 RD
32 AD0/P3.0
31 AD1/P3.1
30 AD2/P3.2
29 AD3/P3.3
28 AD4/P3.4
27 AD5/P3.5
26 AD6/P3.6
25 AD7/P3.7
272110-19

5.0 PIN DESCRIPTIONS

Symbol	Name and Function
V_{CC}	Main supply voltage (5V).
V_{SS}	Digital circuit ground (0V). There are two V_{SS} pins, both of which must be connected.
V_{PD}	RAM standby supply voltage (5V). This voltage must be present during normal operation. In a Power Down condition (i.e., V_{CC} drops to zero), if $\overline{RESET}$ is activated before V_{CC} drops below spec and V_{PD} continues to be held within spec., the top 16 bytes in the Register File will retain their contents.
V_{REF}	Reference voltage for the A/D converter (5V). V_{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0. Must be connected to use A/D or Port 0.
ANGND	Reference ground for the A/D converter. Must be held at nominally the same potential as V_{SS}.
V_{PP}	Programming voltage for the EPROM devices. It should be +12.75V for programming and will float to 5V otherwise. The pin should not be above V_{CC} for ROM and CPU devices. This pin must be left floating in the application circuit for EPROM devices.
XTAL1	Input of the oscillator inverter and of the internal clock generator.
XTAL2	Output of the oscillator inverter.
CLKOUT*†	Output of the internal clock generator. The frequency of CLKOUT is 1/3 the oscillator frequency. It has a 33% duty cycle.
$\overline{RESET}$	Reset input to the chip. Input low for a minimum 10 XTAL1 cycles to reset the chip. The subsequent low-to-high transition re-synchronizes CLKOUT and commences a 10-state-time RESET sequence.
BUSWIDTH*†	Input for bus width selection. If CCR bit 1 is a one, this pin selects the bus width for the bus cycle in progress. If BUSWIDTH is a 1, a 16-bit bus cycle occurs. If BUSWIDTH is a 0 an 8-bit cycle occurs. If CCR bit 1 is a 0, the bus is always an 8-bit bus. If this pin is left unconnected, it will rise to V_{CC}.
NMI*†	A positive transition causes a vector to external memory location 0000H.
INST*†	Output high during an external memory read indicates the read is an instruction fetch. INST is valid throughout the bus cycle.
$\overline{EA}$	Input for memory select (External Access). $\overline{EA}$ equal to a TTL-high causes memory accesses to locations 2000H through 3FFFH to be directed to on-chip ROM/EPROM. $\overline{EA}$ equal to a TTL-low causes accesses to these locations to be directed to off-chip memory. $\overline{EA}$ = +12.75V causes execution to begin in the Programming Mode.
ALE/$\overline{ADV}$	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a signal to demultiplex the address from the address/data bus. When the pin is $\overline{ADV}$, it goes inactive high at the end of the bus cycle. ALE/$\overline{ADV}$ is activated only during external memory accesses.
$\overline{RD}$	Read signal output to external memory. $\overline{RD}$ is activated only during external memory reads.
$\overline{WR}$/$\overline{WRL}$	Write and Write Low output to external memory, as selected by the CCR. $\overline{WR}$ will go low for every external write, while $\overline{WRL}$ will go low only for external writes where an even byte is being written. $\overline{WR}$/$\overline{WRL}$ is activated only during external memory writes.
$\overline{BHE}$/$\overline{WRH}$	Bus High Enable or Write High output to external memory, as selected by the CCR. $\overline{BHE}$ will go low for external writes to the high byte of the data bus. $\overline{WRH}$ will go low for external writes where an odd byte is being written. $\overline{BHE}$/$\overline{WRH}$ is activated only during external memory writes.

5.0 PIN DESCRIPTIONS (Continued)

Symbol	Name and Function
READY	Ready input to lengthen external memory cycles. If the pin is low prior to the falling edge of CLKOUT, the memory controller goes into a wait mode until the next positive transition in CLKOUT occurs with READY high. When the external memory is not being used, READY has no effect. Internal control of the number of wait states inserted into a bus cycle held not ready is available in the CCR.
HSI	Inputs to High Speed Input Unit. Four HSI pins are available: HSI.0, HSI.1, HSI.2 and HSI.3. Two of them (HSI.2 and HSI.3) are shared with the HSO Unit.
HSO	Outputs from High Speed Output Unit. Six HSO pins are available: HSO.0, HSO.1, HSO.2, HSO.3, HSO.4 and HSO.5. Two of them (HSO.4 and HSO.5) are shared with the HSI Unit.
Port 0‡	8-bit high impedance input-only port. These pins can be used as digital inputs and/or as analog inputs to the on-chip A/D converter.
Port 1†	8-bit quasi-bidirectional I/O port.
Port 2†	8-bit multi-functional port. Six of its pins are shared with other functions in the 8096BH, the remaining 2 are quasi-bidirectional.
Ports 3 and 4	8-bit bidirectional I/O ports with open drain outputs. These pins are shared with the multiplexed address/data bus. Ports 3 and 4 are also used as a command, address and data path by EPROM devices operating in the Programming Mode.
PMODE	Determines the EPROM programming mode.
$\overline{\text{PACT}}$	A low signal in Auto Programming Mode indicates that programming is in progress. A high signal indicates programming is complete.
PVAL	A low signal in Auto Programming Mode indicates that the device was programmed correctly.
SALE	A falling edge of Auto Programming Mode indicates that Ports 3 and 4 contain valid programming address/command information (output from master).
$\overline{\text{SPROG}}$	A falling edge in Auto Programming Mode indicates that Ports 3 and 4 contain valid programming data (output from master).
SID	Assigns a pin of Ports 3 and 4 to each slave to pass programming verification.
PALE	A falling edge in Slave Programming Mode and Auto Configuration Byte Programming Mode indicates that Ports 3 and 4 contain valid programming address/command information (input to slave).
$\overline{\text{PROG}}$	A falling edge in Slave Programming Mode indicates that Ports 3 and 4 contain valid programming data (input to slave).
PVER	A high signal in Slave Programming Mode and Auto Configuration Byte Programming Mode indicates the byte programmed correctly.
PVAL	A high signal in Slave Programming Mode indicates the device was programmed correctly.
$\overline{\text{PDO}}$	A low signal in Slave Programming Mode indicates that the PROG pulse was applied for longer than allowed.

*Not available on Shrink-DIP package.
†Not available on 48-pin device.
‡Port 0.0.1.2.3 not available on 48-pin device.

6.0 OPCODE TABLE

Opcode	Instruction
00	SKIP
01	CLR
02	NOT
03	NEG
04	XCH
05	DEC
06	EXT
07	INC
08	SHR
09	SHL
0A	SHRA
0B	RESERVED**
0C	SHRL
0D	SHLL
0E	SHRAL
0F	NORML
10	RESERVED**
11	CLRB
12	NOTB
13	NEGB
14	XCHB
15	DECB
16	EXTB
17	INCB
18	SHRB
19	SHLB
1A	SHRAB
1B	RESERVED**
1C	RESERVED**
1D	RESERVED**
1E	RESERVED**
1F	RESERVED**
20	SJMP
21	SJMP
22	SJMP
23	SJMP
24	SJMP
25	SJMP
26	SJMP
27	SJMP
28	SCALL
29	SCALL
2A	SCALL
2B	SCALL
2C	SCALL
2D	SCALL
2E	SCALL
2F	SCALL
30	JBC
31	JBC
32	JBC
33	JBC

Opcode	Instruction
34	JBC
35	JBC
36	JBC
37	JBC
38	JBS
39	JBS
3A	JBS
3B	JBS
3C	JBS
3D	JBS
3E	JBS
3F	JBS
40	AND DIRECT (3 OPS)
41	AND IMMEDIATE (3 OPS)
42	AND INDIRECT (3 OPS)
43	AND INDEXED (3 OPS)
44	ADD DIRECT (3 OPS)
45	ADD IMMEDIATE (3 OPS)
46	ADD INDIRECT (3 OPS)
47	ADD INDEXED (3 OPS)
48	SUB DIRECT (3 OPS)
49	SUB IMMEDIATE (3 OPS)
4A	SUB INDIRECT (3 OPS)
4B	SUB INDEXED (3 OPS)
4C	MULU DIRECT (3 OPS)
4D	MULU IMMEDIATE (3 OPS)
4E	MULU INDIRECT (3 OPS)
4F	MULU INDEXED (3 OPS)
50	ANDB DIRECT (3 OPS)
51	ANDB IMMEDIATE (3 OPS)
52	ANDB INDIRECT (3 OPS)
53	ANDB INDEXED (3 OPS)
54	ADDB DIRECT (3 OPS)
55	ADDB IMMEDIATE (3 OPS)
56	ADDB INDIRECT (3 OPS)
57	ADDB INDEXED (3 OPS)
58	SUBB DIRECT (3 OPS)
59	SUBB IMMEDIATE (3 OPS)
5A	SUBB INDIRECT (3 OPS)
5B	SUBB INDEXED (3 OPS)
5C	MULUB DIRECT (3 OPS)
5D	MULUB IMMEDIATE (3 OPS)
5E	MULUB INDIRECT (3 OPS)
5F	MULUB INDEXED (3 OPS)
60	AND DIRECT (2 OPS)
61	AND IMMEDIATE (2 OPS)
62	AND INDIRECT (2 OPS)
63	AND INDEXED (2 OPS)
64	ADD DIRECT (2 OPS)
65	ADD IMMEDIATE (2 OPS)
66	ADD INDIRECT (2 OPS)
67	ADD INDEXED (2 OPS)

6.0 OPCODE TABLE (Continued)

Opcode	Instruction
68	SUB DIRECT (2 OPS)
69	SUB IMMEDIATE (2 OPS)
6A	SUB INDIRECT (2 OPS)
6B	SUB INDEXED (2 OPS)
6C	MULU DIRECT (2 OPS)
6D	MULU IMMEDIATE (2 OPS)
6E	MULU INDIRECT (2 OPS)
6F	MULU INDEXED (2 OPS)
70	ANDB DIRECT (2 OPS)
71	ANDB IMMEDIATE (2 OPS)
72	ANDB INDIRECT (2 OPS)
73	ANDB INDEXED (2 OPS)
74	ADDB DIRECT (2 OPS)
75	ADDB IMMEDIATE (2 OPS)
76	ADDB INDIRECT (2 OPS)
77	ADDB INDEXED (2 OPS)
78	SUBB DIRECT (2 OPS)
79	SUBB IMMEDIATE (2 OPS)
7A	SUBB INDIRECT (2 OPS)
7B	SUBB INDEXED (2 OPS)
7C	MULUB DIRECT (2 OPS)
7D	MULUB IMMEDIATE (2 OPS)
7E	MULUB INDIRECT (2 OPS)
7F	MULUB INDEXED (2 OPS)
80	OR DIRECT
81	OR IMMEDIATE
82	OR INDIRECT
83	OR INDEXED
84	XOR DIRECT
85	XOR IMMEDIATE
86	XOR INDIRECT
87	XOR INDEXED
88	CMP DIRECT
89	CMP IMMEDIATE
8A	CMP INDIRECT
8B	CMP INDEXED
8C	DIVU DIRECT
8D	DIVU IMMEDIATE
8E	DIVU INDIRECT
8F	DIVU INDEXED
90	ORB DIRECT
91	ORB IMMEDIATE
92	ORB INDIRECT
93	ORB INDEXED
94	XORB DIRECT
95	XORB IMMEDIATE
96	XORB INDIRECT
97	XORB INDEXED
98	CMPB DIRECT
99	CMPB IMMEDIATE
9A	CMPB INDIRECT
9B	CMPB INDEXED

Opcode	Instruction
9C	DIVUB DIRECT
9D	DIVUB IMMEDIATE
9E	DIVUB INDIRECT
9F	DIVUB INDEXED
A0	LD DIRECT
A1	LD IMMEDIATE
A2	LD INDIRECT
A3	LD INDEXED
A4	ADDC DIRECT
A5	ADDC IMMEDIATE
A6	ADDC INDIRECT
A7	ADDC INDEXED
A8	SUBC DIRECT
A9	SUBC IMMEDIATE
AA	SUBC INDIRECT
AB	SUBC INDEXED
AC	LDBZE DIRECT
AD	LDBZE IMMEDIATE
AE	LDBZE INDIRECT
AF	LDBZE INDEXED
B0	LDB DIRECT
B1	LDB IMMEDIATE
B2	LDB INDIRECT
B3	LDB INDEXED
B4	ADDCB DIRECT
B5	ADDCB IMMEDIATE
B6	ADDCB INDIRECT
B7	ADDCB INDEXED
B8	SUBCB DIRECT
B9	SUBCB IMMEDIATE
BA	SUBCB INDIRECT
BB	SUBCB INDEXED
BC	LDBSE DIRECT
BD	LDBSE IMMEDIATE
BE	LDBSE INDIRECT
BF	LDBSE INDEXED
C0	ST DIRECT
C1	RESERVED**
C2	ST INDIRECT
C3	ST INDEXED
C4	STB DIRECT
C5	RESERVED**
C6	STB INDIRECT
C7	STB INDEXED
C8	PUSH DIRECT
C9	PUSH IMMEDIATE
CA	PUSH INDIRECT
CB	PUSH INDEXED
CC	POP DIRECT
CD	RESERVED**
CE	POP INDIRECT
CF	POP INDEXED

6.0 OPCODE TABLE (Continued)

Opcode	Instruction
D0	JNST
D1	JNH
D2	JGT
D3	JNC
D4	JNVT
D5	JNV
D6	JGE
D7	JNE
D8	JST
D9	JH
DA	JLE
DB	JC
DC	JVT
DD	JV
DE	JLT
DF	JE
E0	DJNZ
E1	RESERVED**
E2	RESERVED**
E3	BR (INDIRECT)
E4	RESERVED**
E5	RESERVED**
E6	RESERVED**
E7	LJMP

Opcode	Instruction
E8	RESERVED**
E9	RESERVED**
EA	RESERVED**
EB	RESERVED**
EC	RESERVED**
ED	RESERVED**
EE	RESERVED**
EF	LCALL
F0	RET
F1	RESERVED**
F2	PUSHF
F3	POPF
F4	RESERVED**
F5	RESERVED**
F6	RESERVED**
F7	TRAP
F8	CLRC
F9	SETC
FA	DI
FB	EI
FC	CLRVT
FD	NOP
FE	*DIV/DIVB/MUL/MULB
FF	RST

*Two Byte Instruction
**Opcodes which do not have a corresponding instruction will not generate an interrupt if executed.

7.0 INSTRUCTION SUMMARY

Mnemonic	Oper-ands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	D ← D + A	↕	↕	↕	↕	↑	—	
ADD/ADDB	3	D ← B + A	↕	↕	↕	↕	↑	—	
ADDC/ADDCB	2	D ← D + A + C	↓	↕	↕	↕	↑	—	
SUB/SUBB	2	D ← D − A	↕	↕	↕	↕	↑	—	
SUB/SUBB	3	D ← B − A	↕	↕	↕	↕	↑	—	
SUBC/SUBCB	2	D ← D − A + C − 1	↓	↕	↕	↕	↑	—	
CMP/CMPB	2	D − A	↕	↕	↕	↕	↑	—	
MUL/MULU	2	D, D + 2 ← D * A	—	—	—	—	—	?	2
MUL/MULU	3	D, D + 2 ← B * A	—	—	—	—	—	?	2
MULB/MULUB	2	D, D + 1 ← D * A	—	—	—	—	—	?	3
MULB/MULUB	3	D, D + 1 ← B * A	—	—	—	—	—	?	3
DIVU	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	↕	↑	—	2
DIVUB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	↕	↑	—	3
DIV	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	?	↑	—	
DIVB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	?	↑	—	
AND/ANDB	2	D ← D and A	↕	↕	0	0	—	—	
AND/ANDB	3	D ← B and A	↕	↕	0	0	—	—	
OR/ORB	2	D ← D or A	↕	↕	0	0	—	—	
XOR/XORB	2	D ← D (excl. or) A	↕	↕	0	0	—	—	
LD/LDB	2	D ← A	—	—	—	—	—	—	
ST/STB	2	A ← D	—	—	—	—	—	—	
LDBSE	2	D ← A; D + 1 ← SIGN(A)	—	—	—	—	—	—	3, 4
LDBZE	2	D ← A; D + 1 ← 0	—	—	—	—	—	—	3, 4
PUSH	1	SP ← SP − 2; (SP) ← A	—	—	—	—	—	—	
POP	1	A ← (SP); SP ← SP + 2	—	—	—	—	—	—	
PUSHF	0	SP ← SP − 2; (SP) ← PSW; PSW ← 0000H I ← 0	0	0	0	0	0	0	
POPF	0	PSW ← (SP); SP ← SP + 2; I ← ↕	↕	↕	↕	↕	↕	↕	
SJMP	1	PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LJMP	1	PC ← PC + 16-bit offset	—	—	—	—	—	—	5
BR [indirect]	1	PC ← (A)	—	—	—	—	—	—	
SCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 16-bit offset	—	—	—	—	—	—	5
RET	0	PC ← (SP); SP ← SP + 2	—	—	—	—	—	—	

7.0 INSTRUCTION SUMMARY (Continued)

Mnemonic	Oper-ands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
J (conditional)	1	PC ← PC + 8-bit offset (if taken)	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D − 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D − 1	↕	↕	↕	↕	↑	—	
NEG/NEGB	1	D ← 0 − D	↕	↕	↕	↕	↑	—	
INC/INCB	1	D ← D + 1	↕	↕	↕	↕	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	↕	↕	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign (D)	↕	↕	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	↕	↕	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb — — — — — — lsb ← 0	↕	?	↕	↕	↑	—	7
SHR/SHRB/SHRL	2	0 → msb — — — — — — lsb → C	↕	?	↕	0	—	↕	7
SHRA/SHRAB/SHRAL	2	msb → msb — — — — — — lsb → C	↕	↕	↕	0	—	↕	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	

7.0 INSTRUCTION SUMMARY (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left shift till msb = 1; D ← shift count	✓	?	0	—	—	—	7
TRAP	0	SP ← SP − 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling $\overline{\text{RESET}}$ low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

8.0 OPCODES, INSTRUCTION LENGTH AND STATE TIMES

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			INDIRECT⑥					INDEXED⑥				
								NORMAL			AUTO-INC.		SHORT			LONG	
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE① TIMES	BYTES	STATE① TIMES	OPCODE	BYTES	STATE① TIMES⑥	BYTES	STATE① TIMES⑥
ARITHMETIC INSTRUCTIONS																	
ADD	2	64	3	4	65	4	5	66	3	6/11	3	7/12	67	4	6/11	5	7/12
ADD	3	44	4	5	45	5	6	46	4	7/12	4	8/13	47	5	7/12	6	8/13
ADDB	2	74	3	4	75	3	4	76	3	6/11	3	7/12	77	4	6/11	5	7/12
ADDB	3	54	4	5	55	4	5	56	4	7/12	4	8/13	57	5	7/12	6	8/13
ADDC	2	A4	3	4	A5	4	5	A6	3	6/11	3	7/12	A7	4	6/11	5	7/12
ADDCB	2	B4	3	4	B5	3	4	B6	3	6/11	3	7/12	B7	4	6/11	5	7/12
SUB	2	68	3	4	69	4	5	6A	3	6/11	3	7/12	6B	4	6/11	5	7/12
SUB	3	48	4	5	49	5	6	4A	4	7/12	4	8/13	4B	5	7/12	6	8/13
SUBB	2	78	3	4	79	3	4	7A	3	6/11	3	7/12	7B	4	6/11	5	7/12
SUBB	3	58	4	5	59	4	5	5A	4	7/12	4	8/13	5B	5	7/12	6	8/13
SUBC	2	A8	3	4	A9	4	5	AA	3	6/11	3	7/12	AB	4	6/11	5	7/12
SUBCB	2	B8	3	4	B9	3	4	BA	3	6/11	3	7/12	BB	4	6/11	5	7/12
CMP	2	88	3	4	89	4	5	8A	3	6/11	3	7/12	8B	4	6/11	5	7/12
CMPB	2	98	3	4	99	3	4	9A	3	6/11	3	7/12	9B	4	6/11	5	7/12
MULU	2	6C	3	25	6D	4	26	6E	3	27/32	3	28/33	6F	4	27/32	5	28/33
MULU	3	4C	4	26	4D	5	27	4E	4	28/33	4	29/34	4F	5	28/33	6	29/34
MULUB	2	7C	3	17	7D	3	17	7E	3	19/24	3	20/25	7F	4	19/24	5	20/25
MULUB	3	5C	4	18	5D	4	18	5E	4	20/25	4	21/26	5F	5	20/25	6	21/26
MUL	2	②	4	29	②	5	30	②	4	31/36	4	32/37	②	5	31/36	6	32/37
MUL	3	②	5	30	②	6	31	②	5	32/37	5	33/38	②	6	32/37	7	33/38
MULB	2	②	4	21	②	4	21	②	4	23/28	4	24/29	②	5	23/28	6	24/29
MULB	3	②	5	22	②	5	22	②	5	24/29	5	25/30	②	6	24/29	7	25/30
DIVU	2	8C	3	25	8D	4	26	8E	3	28/32	3	29/33	8F	4	28/32	5	29/33
DIVUB	2	9C	3	17	9D	3	17	9E	3	20/24	3	21/25	9F	4	20/24	5	21/25
DIV	2	②	4	29	②	5	30	②	4	32/36	4	33/37	②	5	32/36	6	33/37
DIVB	2	②	4	21	②	4	21	②	4	24/28	4	25/29	②	5	24/28	6	25/29

272110-20

NOTES:

*Long indexed and Indirect + instructions have identical opcodes with Short indexed and Indirect modes, respectively. The second byte of instructions using any Indirect or indexed addressing mode specifies the exact mode used. If the second byte is even, use Indirect or Short indexed. If it is odd, use Indirect + or Long indexed. In all cases the second byte of the instruction always specifies an even (word) location for the address referenced.

① Number of state times shown for internal/external operands.

② The opcodes for signed multiply and divide are the opcodes for the unsigned functions with an "FE" appended as a prefix.

⑥ State times shown for 16-bit bus.

8.0 OPCODES, INSTRUCTION LENGTH AND STATE TIMES (Continued)

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			INDIRECT⓪					INDEXED⓪				
								NORMAL			AUTO-INC.		SHORT			LONG	
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE① TIMES	BYTES	STATE① TIMES	OPCODE	BYTES	STATE① TIMES⑥	BYTES	STATE① TIMES⑥
LOGICAL INSTRUCTIONS																	
AND	2	60	3	4	61	4	5	62	3	6/11	3	7/12	63	4	6/11	5	7/12
AND	3	40	4	5	41	5	6	42	4	7/12	4	8/13	43	5	7/12	6	8/13
ANDB	2	70	3	4	71	3	4	72	3	6/11	3	7/12	73	4	6/11	5	7/12
ANDB	3	50	4	5	51	4	5	52	4	7/12	4	8/13	53	5	7/12	6	8/13
OR	2	80	3	4	81	4	5	82	3	6/11	3	7/12	83	4	6/11	5	7/12
ORB	2	90	3	4	91	3	4	92	3	6/11	3	7/12	93	4	6/11	5	7/12
XOR	2	84	3	4	85	4	5	86	3	6/11	3	7/12	87	4	6/11	5	7/12
XORB	2	94	3	4	95	3	4	96	3	6/11	3	7/12	97	4	6/11	5	7/12
DATA TRANSFER INSTRUCTIONS																	
LD	2	A0	3	4	A1	4	5	A2	3	6/11	3	7/12	A3	4	6/11	5	7/12
LDB	2	B0	3	4	B1	3	4	B2	3	6/11	3	7/12	B3	4	6/11	5	7/12
ST	2	C0	3	4	—	—	—	C2	3	7/11	3	8/12	C3	4	7/11	5	8/12
STB	2	C4	3	4	—	—	—	C6	3	7/11	3	8/12	C7	4	7/11	5	8/12
LDBSE	2	BC	3	4	BD	3	4	BE	3	6/11	3	7/12	BF	4	6/11	5	7/12
LDBZE	2	AC	3	4	AD	3	4	AE	3	6/11	3	7/12	AF	4	6/11	5	7/12
STACK OPERATIONS (internal stack)																	
PUSH	1	C8	2	8	C9	3	8	CA	2	11/15	2	12/16	CB	3	11/15	4	12/16
POP	1	CC	2	12	—	—	—	CE	2	14/18	2	14/18	CF	3	14/18	4	14/18
PUSHF	0	F2	1	8													
POPF	0	F3	1	9													
STACK OPERATIONS (external stack)																	
PUSH	1	C8	2	12	C9	3	12	CA	2	15/19	2	16/20	CB	3	15/19	4	16/20
POP	1	CC	2	14	—	—	—	CE	2	16/20	2	16/20	CF	3	16/20	4	16/20
PUSHF	0	F2	1	12													
POPF	0	F3	1	13													

JUMPS AND CALLS							
MNEMONIC	OPCODE	BYTES	STATES	MNEMONIC	OPCODE	BYTES	STATES
LJMP	E7	3	8	LCALL	EF	3	13/16⑤
SJMP	20-27④	2	8	SCALL	28-2F④	2	13/16⑤
BR[ ]	E3	2	8	RET	F0	1	12/16⑤
				TRAP③	F7	1	21/24

272110-21

NOTES:
① Number of state times shown for internal/external operands.
③ The assembler does not accept this mnemonic.
④ The least significant 3 bits of the opcode are concatenated with the following 8 bits to form an 11-bit, 2's complement, offset for the relative call or jump.
⑤ State times for stack located internal/external.
⑥ State times shown for 16-bit bus.

8.0 OPCODES, INSTRUCTION LENGTH AND STATE TIMES (Continued)

CONDITIONAL JUMPS

All conditional jumps are 2 byte instructions. They require 8 state times if the jump is taken, 4 if it is not.[8]

MNEMONIC	OPCODE	MNEMONIC	OPCODE	MNEMONIC	OPCODE	MNEMONIC	OPCODE
JC	DB	JE	DF	JGE	D6	JGT	D2
JNC	D3	JNE	D7	JLT	DE	JLE	DA
JH	D9	JV	DD	JVT	DC	JST	D8
JNH	D1	JNV	D5	JNVT	D4	JNST	D0

JUMP ON BIT CLEAR OR BIT SET

These instructions are 3 byte instructions. They require 9 state times if the jump is taken, 5 if it is not.[8]

	BIT NUMBER							
MNEMONIC	0	1	2	3	4	5	6	7
JBC	30	31	32	33	34	35	36	37
JBS	38	39	3A	3B	3C	3D	3E	3F

LOOP CONTROL

MNEMONIC	OPCODE	BYTES	STATE TIMES
DJNZ	EO	3	5/9 STATE TIME (NOT TAKEN/TAKEN)[8]

SINGLE REGISTER INSTRUCTIONS

MNEMONIC	OPCODE	BYTES	STATES[8]	MNEMONIC	OPCODE	BYTES	STATES[8]
DEC	05	2	4	EXT	06	2	4
DECB	15	2	4	EXTB	16	2	4
NEG	03	2	4	NOT	02	2	4
NEGB	13	2	4	NOTB	12	2	4
INC	07	2	4	CLR	01	2	4
INCB	17	2	4	CLRB	11	2	4

SHIFT INSTRUCTIONS

INSTR MNEMONIC	WORD		INSTR MNEMONIC	BYTE		INSTR MNEMONIC	DBL WD		STATE TIMES[8]
	OP	B		OP	B		OP	B	
SHL	09	3	SHLB	19	3	SHLL	0D	3	7 + 1 PER SHIFT[7]
SHR	08	3	SHRB	18	3	SHRL	0C	3	7 + 1 PER SHIFT[7]
SHRA	0A	3	SHRAB	1A	3	SHRAL	0E	3	7 + 1 PER SHIFT[7]

8.0 OPCODES, INSTRUCTION LENGTH AND STATE TIMES (Continued)

SPECIAL CONTROL INSTRUCTIONS

MNEMONIC	OPCODE	BYTES	STATES(8)	MNEMONIC	OPCODE	BYTES	STATES(8)
SETC	F9	1	4	DI	FA	1	4
CLRC	F8	1	4	EI	FB	1	4
CLRVT	FC	1	4	NOP	FD	1	4
RST(6)	FF	1	166	SKIP	00	2	4

NORMALIZE

MNEMONIC	OPCODE	BYTES	STATE TIMES
NORML	0F	3	11 + 1 PER SHIFT

NOTES:
6. This instruction takes 2 states to pull $\overline{\text{RESET}}$ low, then holds it low for at least one state time to initiate a reset. The reset takes 13 states, at which time the program restarts at location 2080H.
7. Execution will take at least 8 states, even for 0 shift.
8. State times shown for 16-bit bus.

9.0 INTERRUPT TABLE

Vector	Vector Location		Priority
	(High Byte)	(Low Byte)	
Software Trap	2011H	2010H	Not Applicable
Extint	200FH	200EH	7 (Highest)
Serial Port	200DH	200CH	6
Software Timers	200BH	200AH	5
HSI.0	2009H	2008H	4
High Speed Outputs	2007H	2006H	3
HSI Data Available	2005H	2004H	2
A/D Conversion Complete	2003H	2002H	1
Timer Overflow	2001H	2000H	0 (Lowest)

10.0 FORMULAS

Baud Rate Calculations

Using XTAL1:

Mode 0: $\text{Baud Rate} = \frac{\text{XTAL1 frequency}}{4 \cdot (B + 1)}; B \neq 0$

Others: $\text{Baud Rate} = \frac{\text{XTAL1 frequency}}{64 \cdot (B + 1)}$

Using T2CLK:

Mode 0: $\text{Baud Rate} = \frac{\text{T2CLK frequency}}{B}; B \neq 0$

Others: $\text{Baud Rate} = \frac{\text{T2CLK frequency}}{16 \cdot B}; B \neq 0$

Note that B cannot equal 0, except when using XTAL1 in other than Mode 0.

8X9XBH Signature Word

Device	Signature Word
879XBH	896FH
839XBH	896EH
809XBH	Undefined

11.0 RESET STATUS

Register	RESET Value
Port 1	XXXXXXXXB
Port 2	XX0XXXX1B
Port 3	11111111B
Port 4	11111111B
PWM Control	00H
Serial Port (Transmit)	undefined
Serial Port (Receive)	undefined
Baud Rate Register	undefined
Serial Port Control	XXXX0XXXB
Serial Port Status	X00XXXXXB
A/D Command	undefined
A/D Result	undefined
Interrupt Pending	undefined
Interrupt Mask	00000000B
Timer 1	0000H
Timer 2	0000H
WDT	0000H
HSI Mode	XXXXXXXXB
HSI Status	undefined
IOS0	00000000B
IOS1	00000000B
IOC0	X0X0X0X0B
IOC1	X0X0XXX1B
HSI FIFO	empty
HSI CAM	empty
HSO SFR	000000B
PSW	0000H
Stack Pointer	undefined
Program Counter	2080H
Port 1	weak pullups
Port 2.6, Port 2.7	weak pullups
Ports 3 and 4	floating
HSO.0, HSO.1, HSO.2, HSO.3	low
HSO.4, HSO.5	floating
$\overline{RD}$	high
$\overline{WR}/\overline{WRL}$	high
ALE/$\overline{ADV}$	high
$\overline{BHE}/\overline{WRH}$	high
INST	low