8XC196KC/8XC196KD User's Manual

1992
Order Number: 272238-001

intel®

8X196KC/KD USER'S MANUAL

1992

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

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

TABLE OF CONTENTS

Figures

Guide to This Manual 1

CHAPTER 1
GUIDE TO THIS MANUAL

This manual describes the 8XC196KC/KD embedded microcontroller. It is intended for use by both software and hardware designers familiar with the principles of microcontrollers and with the 8XC196KC/KD architecture.

1.1. MANUAL CONTENTS

This manual contains 14 chapters, 3 appendixes, a glossary, and an index. This chapter, Chapter 1, provides an overview of the manual. This section summarizes the contents of the remaining chapters and appendixes. The remainder of this chapter describes notational convertions and terminology used throughout the manual and provides references to related documentation.

Chapter 2 — Introduction to the 8XC196KC/KD — provides an overview of the 8XC196KC/KD hardware and software. The hardware portion compares the two devices and describes the core, internal timing, internal peripherals, and special operating modes. The software portion provides an overview of the MCS®-96 instruction set. It discusses differences between the 8XC196KC/KD instruction set and that of the 8096BH and offers guidelines for program development. (For detailed information about the 8XC196KC/KD instruction set, see Appendix A.)

Chapter 3 — Data Types and Addresses — defines the operand types and addressing modes supported by the MCS-96 architecture. (For additional information about the instruction set, see Chapter 2 and Appendix A.)

Chapter 4 — Memory Partitions — describes the addressable memory space within the 8XC196KC and 8XC196KD. (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Chapter 5 — Interrupts — describes the interrupt control circuitry, priority scheme, and timing for both standard and Peripheral Transaction Server (PTS) interrupts. It also describes the three special interrupts and the five PTS modes. It also explains interrupt programming and control. (For additional information about the instructions and registers discussed in this chapter, see Appendix A and Appendix C.)

Chapter 6 — I/O Ports — describes the five input/output ports of the 8XC196KC/KD, explains how to program the ports, and provides hardware connection hints for quasi-bidirectional ports. (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Chapter 7 — Serial I/O Port — describes the synchronous mode (Mode 0) and the three asynchronous modes (Modes 1, 2, and 3) of the serial I/O port's Universal Asynchronous Receiver and Transmitter (UART) and describes how to program the serial I/O port. (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Chapter 8 — High-Speed Input/Output Unit — describes the HSIO unit, the timer/counter-based I/O system that consists of four individual peripheral modules: Timer 1, Timer 2, the High-Speed Input module, and the High-Speed Output module. It also describes how to program each module. (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Chapter 9 — Analog-to-Digital Converter — provides an overview of the analog-to-digital (A/D) converter and describes how to program the converter, read the conversion results, and interface with external circuitry. (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Chapter 10 — Pulse Width Modulator — provides a functional overview of the Pulse Width Modulator (PWM) modules, describes how to program them, and provides sample circuitry for converting the PWM outputs to analog signals. (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Chapter 11 — Minimum Hardware Considerations — describes options for providing the basic requirements for 8XC196KC/KD operation within a system and discusses other hardware considerations.

Chapter 12 — Special Operating Modes — provides an overview of the Idle, Powerdown, and On-Circuit Emulation (ONCE) modes and describes how to enter and exit each mode. (For descriptions of the instructions discussed in this chapter, see Appendix A; for additional information about the signals and registers, see Appendix B and Appendix C.)

Chapter 13 — Interfacing with External Memory — lists the external memory signals and describes the registers that control the external memory interface and the various external bus modes and features. It discusses bus-width and memory configurations, internal Ready control, the bus-hold protocol, and bus-control modes. Finally, it provides timing information for the system bus.

Chapter 14 — Programming the Nonvolatile Memory — contains procedures and guidelines to help you program the nonvolatile, One-Time-Programmable Read-Only Memory (OTPROM). (For additional information about the signals and registers discussed in this chapter, see Appendix B and Appendix C.)

Appendix A — 8XC196KC/KD Instruction Set Reference — provides reference information for the 8XC196KC/KD instruction set. It describes each instruction; shows the relationships between instructions and Program Status Word (PSW) flags; and lists hexadecimal opcodes, instruction lengths, and execution times. (For additional information about the instruction set, see Chapters 2 and 3. For detailed information about the PSW flags, see Appendix C.)

Appendix B — 8XC196KC/KD Signal Descriptions — provides reference information for the pin functions of the 8XC196KC and 8XC196KD, including descriptions of the pin functions, reset status of the I/O and control pins, and package pin assignments.

Appendix C — 8XC196KC/KD Registers — provides reference information for the registers and interrupts of the 8XC196KC/KD. It describes each register in detail and provides information about interrupts, including sources, vectors, and priorities.

Glossary — defines terms with special meaning used throughout the manual.

Index — lists key topics with page number references.

1.2. NOTATIONAL CONVENTIONS AND TERMINOLOGY

The following notations and terminology are used throughout this manual. The Glossary defines other terms with special meanings.

#	The pound symbol (#) has either of two meanings, depending on the context. When used with a signal name, the symbol means that the signal is active low. When used in an instruction, the symbol prefixes an immediate value in immediate addressing mode.
italics	Italics identify variables and introduce new terminology. The context in which italics are used distinguishes between the two possible meanings. Variables must be replaced with correct values.
Assert and Deassert	The terms *assert* and *deassert* refer to the act of making a signal active (enabled) and inactive (disabled), respectively. The polarity (high/low) is defined by the signal name. Active-low signals are designated by a pound symbol (#) suffix; active-high signals have no suffix. To assert RD# is to drive it low; to assert ALE is to drive it high; to deassert RD# is to drive it high; to deassert ALE is to drive it low.
Device Names	The notation *8XC196KC/KD* is used to refer to both devices when information applies to both. When information differs between devices, the individual names *8XC196KC* and *8XC196KD* are used. For those rare cases in which information differs between different versions of the same device, the version is also used (e.g., *8XC196KC-C* or *8XC196KC (C-Step)*). The data sheets listed in the "Related Documents" section on page 1-5 contain device- and version-specific information.

Instructions	Instruction mnemonics are shown in upper case to avoid confusion. You may use either upper case or lower case.
Numbers	Hexadecimal numbers are represented by a string of hexadecimal digits followed by the character *H*. If the number would otherwise begin with *A* through *F*, a zero prefix is added. (For example, *FF* is shown as *0FFH*.) Decimal and binary numbers are represented by their customary notations. (That is, 255 is a decimal number; 1111 1111 is a binary number.)
Units of Measure	The following abbreviations are used to represent units of measure:

A	amps, amperes
DCV	direct current volts
Kbyte	kilobytes
KΩ	kilo-ohms
mA	milliamps, milliamperes
Mbyte	megabytes
MHz	megahertz
ms	milliseconds
mW	milliwatts
ns	nanoseconds
pF	picofarads
W	watts
V	volts
μA	microamps, microamperes
μF	microfarads
μs	microseconds

Register Bits	Bit locations are indexed by 0–7 (or 0–15), where bit 0 is the least-significant bit and 7 (or 15) is the most-significant bit. An individual bit is represented by the register name, followed by a period and the bit number. For example, WSR.7 is bit 7 of the Window Select Register. In some discussions, bit names are used. For example, the name of WSR.7 is HLDEN.
Register Names	Register names are shown in upper case. For example, TIMER2 is the Timer 2 register; Timer 2 is the timer. If a register name contains a lower case character, it represents more than one register. For example, PWM*x*_CONTROL represents three registers: PWM0_CONTROL, PWM1_CONTROL, and PWM2_CONTROL.
Reserved Bits	Certain bits are described as *reserved* bits. These bits are not used in the 8XC196KC/KD, but they may be used in future implementations. To help ensure that a current software design is compatible with future implementations, reserved bits should be cleared (given a value of "0").
Set and Clear	The terms *set* and *clear* refer to the value of a bit or the act of giving it a value. If a bit is *set*, its value is "1"; *setting* a bit gives it a "1" value. If a bit is *clear*, its value is "0"; *clearing* a bit gives it a "0" value.

Signal Names Signal names are shown in upper case. When several signals share a common name, an individual signal is represented by the signal name, followed by a period and a number. For example, the five HSO signals are named HSO.1, HSO.2, etc. Port pins are represented in the same manner: P0.0, P0.1, P0.2, etc. Exceptions are the PWM*x* pins, ACH*x* pins and AD*x* pins. For compatability with earlier devices, these signal names do not use the period to separate the name and number. A pound symbol (#) appended to a signal name identifies an active-low signal.

1.3. RELATED DOCUMENTS

The following documents contain additional information that is useful in designing systems that incorporate the 8XC196KC/KD microcontroller. These documents are available through Intel Literature. In the U.S. and Canada call 1-800-548-4725 to order.

1.3.1. Software Literature

iC-96 Compiler User's Guide	Order Number 481195
MCS®-96 Macro Assembler User's Guide for DOS Systems	Order Number 122350
MCS®-96 Utilities User's Guide for DOS Systems	Order Number 122355
PL/M-96 User's Guide (for DOS and Intel Systems)	Order Number 481643
PL/M-96 User's Guide for DOS Systems	Order Number 481644
The 8096 Floating-Point Arithmetic Library for DOS Systems	Order Number 122365

1.3.2. Hardware Literature

Embedded Microcontrollers and Processors Handbook (2 vols.)	Order Number 270645
Embedded Applications Handbook	Order Number 270648

1.3.3. Individual Data Sheets

These data sheets are included in the *Embedded Microcontrollers and Processors Handbook* and are also available individually.

8XC196KC Commercial/Express CHMOS Microcontroller	Order Number 270942
8XC196KD Commercial CHMOS Microcontroller	Order Number 272145

1.3.4. Application Notes

These applications notes are included in the *Embedded Applications Handbook.*

AP-125, Designing Microcontroller Systems for Electrically Noisy Environments

AP-406, MCS®-96 Analog Acquisition Primer

These applications notes are included in the *Embedded Microcontrollers and Processors Handbook* and are also available individually.

AP-428, Distributed Motor Control Using the 80C196KB	Order Number 270701
AP-466, Using the 80C196KB	Order Number 272116

1.3.5. Related Software

The following software package is an on-line reference guide and a graphical learning tool.

ApBUILDER	Order Number 272216

Introduction to the 8XC196KC/KD

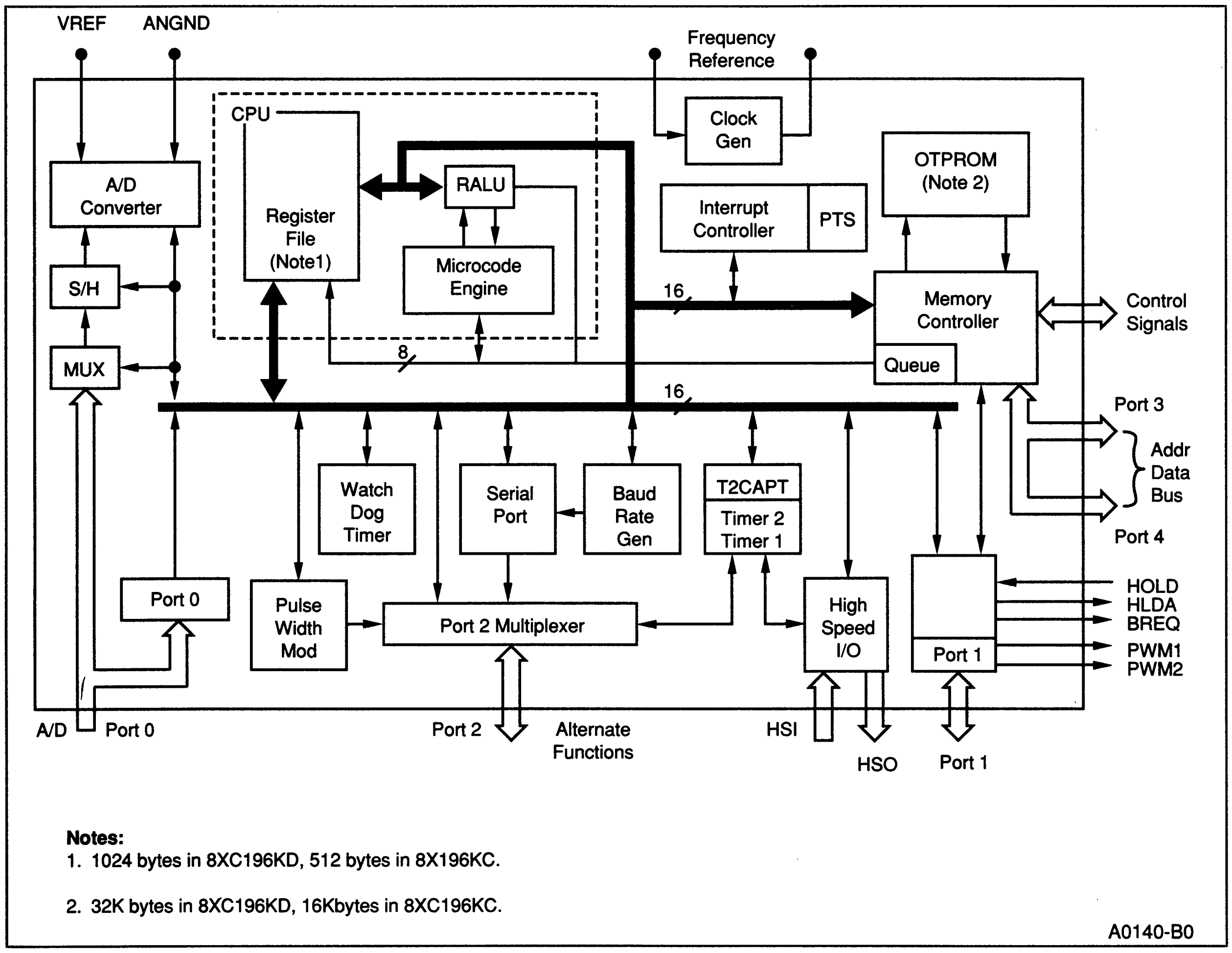
2

CHAPTER 2
INTRODUCTION TO THE 8XC196KC/KD

The 8XC196KC and 8XC196KD are 16-bit CHMOS microcontrollers designed to handle high-speed calculations and fast input/output (I/O) operations. They share a common architecture and instruction set with other members of the MCS-96 family. This chapter provides a high-level overview of both the architecture and software.

Typical applications using the MCS-96 products include closed-loop control and mid-range digital signal processing. Modems, motor-control systems, printers, engine-control systems, photocopiers, anti-lock brakes, air conditioner control systems, disk drives, and medical instrumentation all use MCS-96 products.

Figure 2-1 is a block diagram of the 8XC196KC and 8XC196KD. Each device has a 16-bit-wide Central Processing Unit (CPU) that connects to both an interrupt controller and a memory controller via a 16-bit CPU bus. An extension of this bus connects the CPU to the internal peripheral modules. In addition, an 8-bit CPU bus transfers instruction bytes from the memory controller to the instruction register in the Register Arithmetic-Logic Unit (RALU).



Notes:
1. 1024 bytes in 8XC196KD, 512 bytes in 8X196KC.
2. 32K bytes in 8XC196KD, 16Kbytes in 8XC196KC.

Figure 2-1. 8XC196KC/KD Block Diagram

2.1. COMPARISON OF THE 8XC196KC AND 8XC196KD

The 8XC196KD is a high-speed version of the 8XC196KC with twice as much internal One-Time-Programmable ROM (OTPROM) and RAM (see Table 2-1). The 8XC196KC and 8XC196KD have the same number and types of peripheral modules.

Table 2-1. 8XC196KC and 8XC196KD Comparisons

Feature	8XC196KC	8XC196KD
Addressable Memory Space	64 Kbytes	64 Kbytes
Internal RAM (including SFRs)	512 bytes	1024 bytes
One-Time-Programmable ROM	16 Kbytes	32 Kbytes
Maximum Operating Frequency	16 MHz	20 MHz

2.2. 8XC196KC/KD CORE

The core of the 8XC196KC/KD consists of Central Processing Unit (CPU), a memory controller, and an interrupt controller (see Figure 2-2). The CPU contains a Register/Arithmetic Logic Unit (RALU) and a Register File.

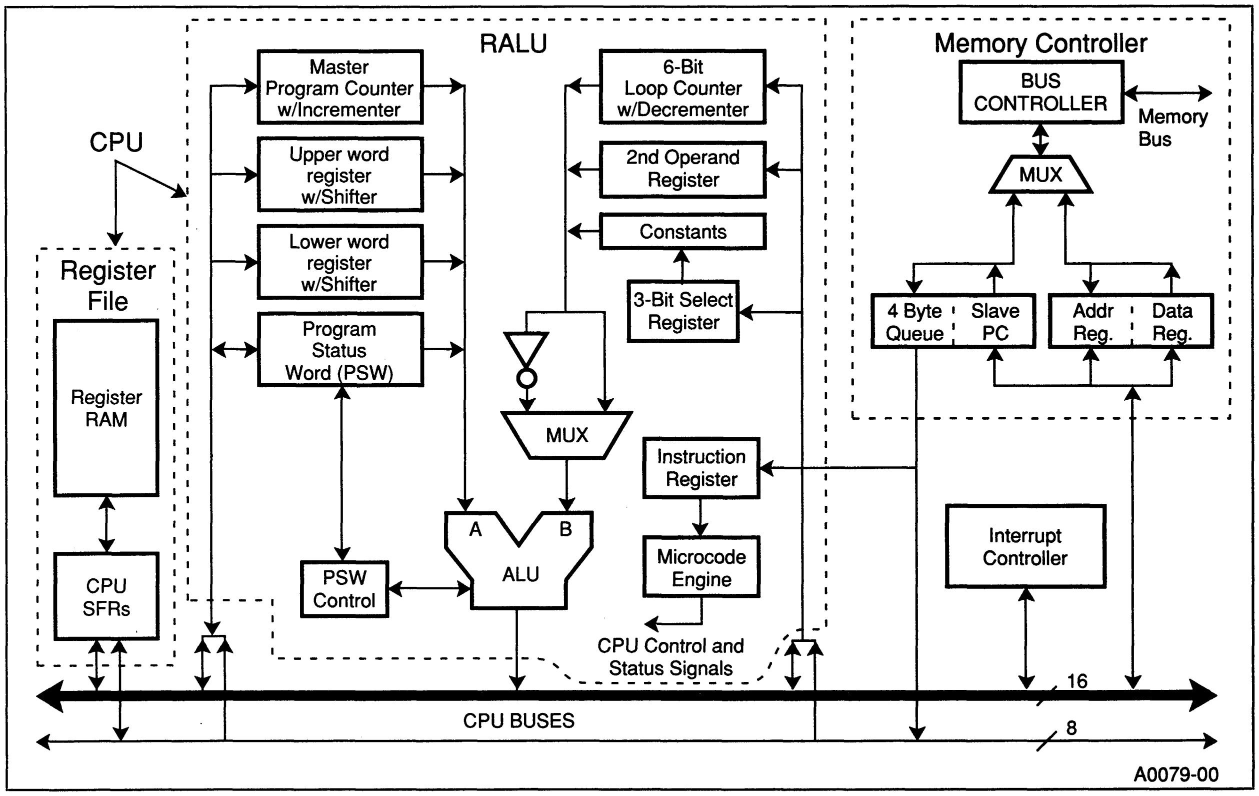


Figure 2-2. Block Diagram of the 8XC196KC/KD Core

2.2.1. CPU Control

The CPU is controlled by the microcode engine, which instructs the RALU to perform operations using bytes, words, or double words from either the 256-byte lower Register File or through a window that directly accesses the upper Register File. CPU instructions move from the four-byte queue in the memory controller into the RALU's instruction register. The microcode engine decodes the instructions and then generates the sequence of events that cause desired functions to occur.

2.2.2. Register File

The Register File is divided into an upper and lower file. The lower Register File contains 24 bytes of Special Function Register (SFR) space and 232 bytes of general-purpose register RAM. The upper Register File contains only general-purpose register RAM (256 bytes in the 8XC196KC and 768 bytes in the 8XC196KD). The general-purpose register RAM can be accessed as bytes, words, or double-words.

The RALU accesses the upper and lower Register Files differently. The lower Register File is always directly accessible via the Register-Direct address mode (see "Addressing Modes" in Chapter 3). The upper Register File is accessible via the Register-Direct address mode only when *vertical windowing* is enabled. Vertical windowing is a technique that maps blocks of the upper Register File into a *window* in the lower Register File. See Chapter 4, "Memory Partitions," for more information about the Register File and windowing.

2.2.3. Register Arithmetic-Logic Unit (RALU)

The RALU contains a 17-bit Arithmetic Logic Unit (ALU), the Program Status Word (PSW), the master program counter (PC), the instruction register, the microcode engine, a constants register, a 3-bit select register, a loop counter, and three temporary registers (the upper-word, lower-word, and second operand registers). All registers, except the three-bit select register, are either 16 or 17 bits (16 bits plus a sign extension) wide. Some of these registers can reduce the ALU's workload by performing simple operations. Words enter the ALU through the A and B inputs.

The RALU speeds up calculations by storing constants (i.e., 0, 1, and 2) in the constants register so that they are readily available when complementing, incrementing, or decrementing bytes or words.

The PSW contains one bit (PSW.1) that globally enables or disables servicing of all maskable interrupts, one bit (PSW.2) that enables or disables the Peripheral Transaction Server (PTS), and six Boolean flags that reflect the state of the user's program. Appendix C provides a detailed description of the PSW.

The PC contains the address of the next instruction and has a built-in incrementer that automatically loads the next sequential address. If a jump, interrupt, call, or return changes the address sequence, the ALU loads the appropriate address into the PC.

The upper and lower word registers are used together for 32-bit instructions and as temporary registers for many instructions. Since they have their own shift logic, the RALU also uses them for operations that require logical shifts, (e.g., 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. Repetitive shifts are counted by the 6-bit loop counter.

The second operand register stores the second operand when the microcode engine executes a two-operand instruction. This includes the multiplier during multiply instructions and the divisor during divide instructions. During subtractions, the output of this register is complemented before it is moved into the B input of the ALU.

2.2.3.1. CODE EXECUTION

The RALU performs most calculations for the 8XC196KC/KD, but it does not use an accumulator. Instead it operates directly on the lower Register File, which essentially provides 256 *accumulators*. Because data does not flow through a single accumulator, the 8XC196KC/KD's code executes faster and more efficiently.

For example, the following 80C186 code multiplies two 16-bit variables (FACTOR_1 and FACTOR_2) and stores the 32-bit result in a third variable (RESULT).

```
MOV AX, FACTOR_1                    ;move factor_1 into a register
MUL AX, FACTOR_2                    ;multiply factor_2 and contents
                                    ;of AX register and store in AX
MOV RESULT, AX                      ;move lower byte into "result"
MOV RESULT+2,DX                     ;move upper byte into "result+2"
```

The following example shows the equivalent code for the 8XC196KC/KD.

```
MUL RESULT, FACTOR_1, FACTOR_2      ;multiply factor_1 & factor_2
                                    ;store answer in result
```

The 8XC196KC/KD can perform this operation in one instruction because it combines a large set of general-purpose registers with a 3-operand instruction format. This format allows a single instruction to specify two source registers and a separate destination register.

2.2.4. Memory Controller

The RALU communicates with all memory, except the Register File, through the memory controller. (It communicates with the upper Register File through the memory controller except when *vertical windowing* is used; see Chapter 4.) The memory controller contains address and data registers, a four-byte queue, a slave program counter (slave PC), and a bus controller.

The bus controller drives the memory bus, which consists of the internal OTPROM bus, the internal RAM bus, and the external address/data bus. The bus controller receives memory-access requests from either the RALU or the four-byte pre-fetch queue; queue requests have priority. This queue is transparent to the RALU and the user.

When the bus controller receives a request from the queue, it fetches the code from the address contained in the slave PC. This increases execution speed since the next instruction byte is available immediately and the processor need not wait for the master PC to send the address to the memory controller. If a jump, interrupt, call, or return changes the address sequence, the master PC loads the new address into the slave PC, the queue is flushed, and processing continues.

NOTE

When using a logic analyzer to debug code, remember that instructions are preloaded into the four-byte queue and are not necessarily executed immediately after they are fetched.

2.2.5. Interrupt Controller

The programmable Interrupt Controller has a hardware priority scheme that can be modified by user software. These interrupts are serviced by user-written interrupt service routines. In addition, the 8XC196KC/KD provides a microcoded hardware interrupt processor, the Peripheral Transaction Server (PTS). The PTS responds to interrupts with a fixed set of actions, such as transferring data, starting an A/D conversion, reading the High-Speed Input module's FIFO, and loading events into the High-Speed Output module. The PTS completes these tasks much more quickly than standard interrupt-driven software service routines can. The PTS can service all interrupts except NMI, Trap, and Unimplemented Opcode. PTS cycles have a higher priority than standard interrupts and may temporarily suspend interrupt service routines. See Chapter 5, "Interrupts" for more information.

2.3. INTERNAL TIMING

The clock generator halves the frequency of the signal on XTAL1 and produces the two internal timing signals, PH1 and PH2. These signals are active when high. The rising edges of PH1 and PH2 generate CLKOUT, the output of the internal clock generator (see Figure 2-3).

The combined period of PH1 and PH2 defines the basic time unit known as a *state time* or *state*. At the maximum 8XC196KD frequency of 20 MHz, one state time equals 100 ns. At the maximum 8XC196KC frequency of 16 MHz, one state time equals 125 ns. Because the 8XC196KC/KD can operate at many frequencies, this manual defines time requirements in terms of state times rather than specific times. Consult the latest data sheet for AC timing specifications.

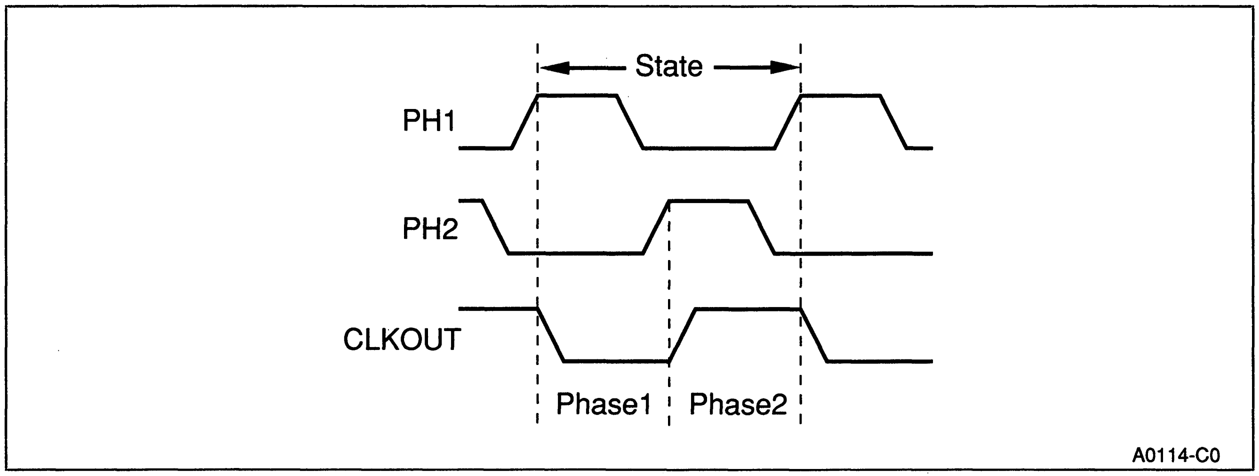


Figure 2-3. Internal Clock Phases

NOTE

A CLKOUT disable bit was added to the IOC3 register in the 8XC196KD and the 8XC196KC (C-Step). Setting this bit disables the CLKOUT signal, which can reduce system noise. This feature is not available in earlier versions of the 8XC196KC. See Appendix C for a description of the IOC3 register.

2.4. INTERNAL PERIPHERALS

The 8XC196KC/KD's internal peripheral modules provide special functions for a variety of applications.

2.4.1. Standard I/O Ports

The 8XC196KC/KD has five 8-bit I/O ports. Some are input-only, some are output-only, some are bidirectional, and some support multiple functions. Port 0 is an input port that is also the analog input for the A/D converter. Port 1 is a quasi-bidirectional port. Port 1 pins are multiplexed with bus control signals and two outputs from the Pulse Width Modulator.

Port 2 contains three types of port lines: quasi-bidirectional, input, and output. Other functions on the 8XC196KC/KD share the Port 2 input and output lines. Ports 3 and 4 are open-drain bidirectional ports that share their pins with the address/data bus. See Chapter 6, "I/O Ports," for more information.

2.4.2. Serial I/O Port

The serial I/O port is an asynchronous/synchronous port that includes a Universal Asynchronous Receiver and Transmitter (UART). The UART has one synchronous mode (Mode 0) and three asynchronous modes (Modes 1, 2 and 3). The asynchronous modes are full duplex, meaning that they can transmit and receive data simultaneously. The receiver on the 8XC196KC/KD is double buffered, so the reception of a second byte may begin before the first byte is read. The transmitter is also double buffered and can generate continual transmissions. See Chapter 7, "Serial I/O Port," for more information.

2.4.3. The High-Speed Input/Output (HSIO) Unit

The HSIO unit contains four individual peripheral modules: Timer 1, Timer 2, High-Speed Input, High-Speed Output. Together, these modules form a flexible timer/counter-based I/O system. See Chapter 8, "High-Speed Input/Output Unit," for more information.

2.4.3.1. TIMER 1 AND TIMER 2

Timer 1 is a free-running timer that is incremented every eight state times. It is the time base for the High-Speed Input module and optionally for the High-Speed Output module.

Timer 2 counts both positive and negative input transitions. It can be used as the time base for the High-Speed Output module, as an up/down counter, or as an extra timer.

2.4.3.2. HIGH-SPEED INPUT (HSI)

The HSI module can record times of external events with an eight-state-time resolution. It can monitor four independently configurable inputs and capture the value of Timer 1 when an event takes place. The four types of events that can trigger captures include rising edges, falling edges, rising or falling edges, or every eighth rising edge. The HSI module can store up to eight entries (Timer 1 values): seven in the seven-level FIFO and one in the HSI holding register.

2.4.3.3. HIGH-SPEED OUTPUT (HSO)

The HSO module can trigger events at specified times based on Timer 1 or Timer 2. These programmable events include starting an A/D conversion, resetting Timer 2, generating up to four software timers, and setting or clearing one or more of the six HSO output lines. The HSO unit stores pending events and the specified times in a Content-Addressable Memory (CAM) file. This file stores up to eight commands. Each command specifies the action time, the nature of the action, whether an interrupt is to occur, and whether Timer 1 or Timer 2 is the reference timer.

2.4.4. Analog-to-Digital Converter

The analog-to-digital (A/D) converter converts an analog input voltage to a digital equivalent. Resolution is either 8 or 10 bits; sample and convert times are programmable. Automated A/D conversions and result storage are facilitated by the A/D Scan Mode of the PTS. The main components of the A/D Converter are a sample and hold, an 8-channel multiplexer, and an 8-bit or 10-bit *successive approximation* analog-to-digital converter. See Chapter 9, "Analog-to-Digital Converter," for more information.

2.4.5. Pulse Width Modulator (PWM)

The 8XC196KC/KD has three PWM modules. The output waveform from each is a variable duty cycle pulse that occurs every 256 or 512 state times as programmed. Several types of motors require a PWM waveform for most efficient operation. When filtered, the PWM waveform will produce a DC level that can change in 256 steps by varying the duty cycle. See Chapter 10, "Pulse Width Modulator," for more information.

2.4.6. Watchdog Timer

The Watchdog Timer is an internal timer that resets the device if the software fails to operate properly. See Chapter 11, "Minimum Hardware Considerations," for more information.

2.5. SPECIAL OPERATING MODES

In addition to the normal execution mode, the 8XC196KC/KD operates in several special-purpose modes. Idle mode and Powerdown mode conserve power when the device is inactive, ONCE mode electrically isolates the 8XC196KC/KD from the system, and several other modes provide programming options for nonvolatile memory. See Chapter 12, "Special Operating Modes," for more information about Idle, Powerdown, and ONCE modes and Chapter 13, "Programming the Nonvolatile Memory," for details about programming options.

2.5.1. Reducing Power Consumption

In Idle mode, the CPU stops executing instructions, but the peripheral clocks remain active. Power consumption drops to about 40% of normal execution mode consumption. Either a hardware reset or any enabled interrupt source will bring the device out of Idle mode.

In Powerdown mode, all internal clocks are frozen at logic state zero and the oscillator is shut off. Internal RAM and most peripherals retain their data if V_{CC} is maintained. Power consumption drops into the µW range.

2.5.2. Testing the Printed Circuit Board

ONCE mode electrically isolates the 8XC196KC/KD from the system. By invoking ONCE mode, you can test the printed circuit board while the 8XC196KC/KD is soldered onto the board.

2.5.3. Programming the 8XC196KC/KD

The 8XC196KC/KD supports Auto Programming Mode, Slave Programming Mode, and Run-Time Programming.

- Auto Programming Mode enables the 8XC196KC/KD to program itself from an external EPROM, without an EPROM programmer.
- Slave Programming Mode supports programming with an EPROM programmer. While using this programming mode, you can program and verify any single word in the OTPROM.
- Run-Time Programming allows you to program individual OTPROM locations during normal code execution, while under complete software control.

2.6. INTRODUCTION TO THE 8XC196KC/KD SOFTWARE

This section provides an overview of the MCS-96 instruction set, discusses differences between the 8XC196KC/KD instruction set and that of the 8096BH, and offers guidelines for program development.

Appendix A provides reference information for the 8XC196KC/KD instruction set. It includes descriptions of the instructions, hexadecimal opcodes, instruction lengths, execution times, and the relationships between Program Status Word (PSW) flags and the instructions. Appendix C provides a detailed description of the PSW and SFRs. Chapter 3 describes data types and addressing modes.

2.6.1. Overview of the MCS®-96 Instruction Set

The MCS-96 instruction set contains a full set of arithmetic and logical operations for the 8- and 16-bit data types (BYTE and SHORT-INTEGER, WORD and INTEGER). It supports the 32-bit data types (DOUBLE-WORD and LONG-INTEGER) only as operands in shift operations, as the dividends of 32-by-16 divide operations, and as products of 16-by-16 multiply operations. The remaining operations on 32-bit variables can be implemented by combinations of 16-bit operations.

For example, the following sequences of 16-bit operations perform a 32-bit addition and a 32-bit subtraction, respectively.

```
ADD AX,CX ; (ADD_2op)
ADDC BX,DX

SUB AX,CX ; (SUB_2op)
SUBC BX,DX
```

The instruction set also supports conversions between the data types. The LDBZE (load byte, zero extended) instruction converts a BYTE to a WORD. CLR (clear) can convert a WORD to a DOUBLE-WORD by clearing (writing zeros to) the upper WORD of the DOUBLE-WORD. LDBSE (load byte, sign extended) converts a SHORT-INTEGER into an INTEGER. EXT (sign extend) converts an INTEGER to a LONG-INTEGER.

The MCS-96 instructions for addition, subtraction, and comparison do not distinguish between unsigned WORDs and signed INTEGERs. However, the conditional jump instructions allow you to treat the results of these operations as signed or unsigned quantities. For example, the CMPB (compare byte) instruction is used to compare both signed and unsigned eight-bit quantities. Following a compare operation, you can use the JH (jump if higher) instruction for unsigned operands or the JGT (jump if greater than) instruction for signed operands.

The hardware does not directly support operations on REAL (floating point) variables. Those operations are supported by the floating point library for the 8XC196KC/KD (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 NORML instruction and by the Sticky Bit (ST) flag in the PSW. The NORML instruction normalizes a 32-bit variable; the Sticky Bit (ST) flag can be used in conjunction with the Carry (C) flag to achieve finer resolution in rounding.

2.6.2. Additions to the MCS®-96 Instruction Set

For users already familiar with the 8096BH, this section briefly describes the instructions that have been added to the standard MCS-96 instruction set to form the 8XC196KC/KD instruction set. Please refer to Appendix A for detailed descriptions.

BMOV	BLOCK MOVE. Moves a block of word data from one location to another in memory. This instruction cannot be interrupted.
BMOVI	INTERRUPTABLE BLOCK MOVE. Moves a block of word data from one location to another in memory. This instruction is identical to BMOV, except that BMOVI can be interrupted.
CMPL	COMPARE LONG. Compares the magnitudes of two double-word operands.

DJNZW	DECREMENT AND JUMP IF NOT ZERO WORD. Decrements the value of the word operand and jumps if the result is other than zero.
DPTS	DISABLE PTS. Clears PSW.2, which disables the Peripheral Transaction Server (PTS).
EPTS	ENABLE PTS. Sets PSW.2, which enables the Peripheral Transaction Server (PTS).
IDLPD	IDLE/POWERDOWN. Causes the device either to enter Idle mode, to enter Powerdown mode, or to execute a reset sequence, depending on the value of the operand.
POPA	POPA. Used instead of POPF, to support the eight added interrupts. It pops two words off the stack, placing the first into the INT_MASK1/WSR register-pair and the second into the PSW/INT_MASK word.
PUSHA	PUSH ALL. Used instead of PUSHF, to support the eight added interrupts. It pushes two words onto the stack: the PSW/INT_MASK word and the word formed by the INT_MASK1/WSR register-pair. It clears the PSW, INT_MASK, and INT_MASK1 registers.
TIJMP	TABLE INDIRECT JUMP. Selects an address from a table of addresses, calculates the destination address, and jumps to that address. The TIJMP instruction can reduce the time required to access a look-up table.
XCH	EXCHANGE WORD. Exchanges the value of the source word operand with that of the destination word operand.
XCHB	EXCHANGE BYTE. Exchanges the value of the source byte operand with that of the destination byte operand.

2.6.3. Instruction Set Differences

For many instructions, execution times are shorter on the 8XC196KC/KD than on the 8096BH. The multiply instructions are nearly twice as fast. For example, a 16-by-16 unsigned multiply operation that took 25 state times on the 8096BH takes only 14 state times on the 8XC196KC/KD. Many zero- and one-operand instructions and most instructions that use external data take one or two fewer state times on the 8XC196KC/KD than on the 8096BH.

Indexed and indirect operations relative to the Stack Pointer (SP) work differently on the 8XC196KC/KD than on the 8096BH. On the 8096BH, the address is calculated based on the value of the SP before it is updated; on the 8XC196KC/KD the updated SP is used. The offset for PUSH [SP], POP [SP], PUSH *nn*[SP], and POP *nn*[SP] instructions may need to be changed by a count of two.

2.6.4. Software Standards and Conventions

For a software project of any size, it is a good idea to modularize the program and to establish standards that control communication between the modules. These standards vary with the needs of the final application. However, all standards must include some mechanism for passing parameters to procedures and returning results from procedures. We recommend that you use 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. It also allows the programmer access to the floating-point arithmetic library (FPAL-96) that PLM-96 uses to operate on REAL variables.

2.6.4.1. USING REGISTERS

The MCS-96 architecture provides a 256-byte lower Register File. Some of these registers are used for register-mapped I/O devices and special functions such as the Zero Register and the Stack Pointer. The remaining bytes in the lower Register File, some 232 of them, are available for your use.

To use these registers effectively, you must have some overall strategy for allocating them. PLM-96 adopts a simple and effective strategy. PLM-96 allocates the eight bytes between addresses 1CH and 23H as temporary storage (calling the starting address of this region PLMREG), and treats the remaining area in the Register File as a segment of memory that is allocated as required.

Special Function Registers (SFRs) can be operated on as BYTEs or WORDs, unless otherwise specified. Use caution when using an SFR as the source of an operand or as the base or index register for indirect or indexed operations. Unexpected results can occur because external events can change SFRs and reading some SFRs clears them. Consider the potential for an SFR to change value, especially when using high-level languages, which do not always allow for SFR-type registers.

2.6.4.2. ADDRESSING 32-BIT OPERANDS

The 32-bit operands (DOUBLE-WORDs and LONG-INTEGERs) are formed by two adjacent 16-bit words in memory. The least significant word of a DOUBLE-WORD is always in the lower address, even when the data is in the stack (which means that the most significant word must be pushed into the stack first). The address of a 32-bit operand is that of its least significant byte.

The hardware supports the 32-bit data types as operands in shift operations, as the dividends of 32-by-16 divide operations, and as products of 16-by-16 multiply operations. For these operations, the 32-bit operand must reside in the internal Register File and must be aligned at an address that is evenly divisible by four.

2.6.4.3. LINKING SUBROUTINES

Parameters are passed to subroutines via the stack. Parameters are pushed into the stack in the order in which they are encountered in the scanning of the source text. The 8-bit parameters (BYTE and SHORT-INTEGER) are pushed into the stack with the high order byte undefined. The 32-bit parameters (LONG-INTEGER, DOUBLE-WORD, and REAL) are pushed onto 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:

```
Stack Image
------------ | ?????? ; param1      |
             | high word of param2 |
             | low word of param2  |
             | param3              |
             | return address      | <-- Stack_Pointer
```

If a procedure returns a value to the calling code (as opposed to modifying more global variables) then the result is returned in the PLMREG variable. 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:

- Procedures can always assume that the eight bytes of Register File memory starting at PLMREG can be used as temporary storage within the body of the procedure.
- Code that calls a procedure must assume that the procedure modifies the eight bytes of Register File memory starting at PLMREG.
- Code that calls a procedure must assume that the procedure modifies the Program Status Word (PSW) condition flags (Z, N, V, VT, C, and ST), because procedures do not save and restore the PSW.
- 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. INTERRUPT procedures do not conform to the rules of normal procedures. Parameters cannot be passed to these procedures and they cannot return results. Since INTERRUPT procedures can execute essentially at any time, they must save and restore the PSW and PLMREG.

2.6.5. Software Protection Features and Guidelines

The 8XC196KC/KD has several features to assist in recovering from hardware and software errors. The Unimplemented Opcode interrupt provides protection from executing unimplemented opcodes. The hardware reset instruction (RST) can cause a reset if the program counter goes out of bounds. The RST instruction opcode is 0FFH, so the processor will reset itself if it reads in bus lines that have been pulled high. The Watchdog Timer (WDT) can also reset the device in the event of a hardware or software error.

We recommend that you fill unused areas of code with NOPs and periodic jumps to an error routine or RST instruction. This is particularly important in the code surrounding lookup tables, since executing lookup tables will cause undesired results. Wherever space allows, each table should be surrounded by seven NOPs (because the longest 8XC196KC/KD instruction has seven bytes) and a RST or a jump to an error routine. Since RST is a one-byte instruction, the NOPs are unnecessary 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.

When using the WDT for software protection, we recommend that you reset the WDT from only one place in code, reducing the chance of an undesired WDT reset. 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 10 milliseconds will provide protection only for catastrophic failures.

Data Types and Addresses

3

CHAPTER 3
DATA TYPES AND ADDRESSES

This chapter defines the operand types and addressing modes supported by the MCS-96 architecture.

Chapter 2, "Introduction to the 8XC196KC/KD," provides an overview of the MCS-96 instruction set. It discusses differences between the 8XC196KC/KD instruction set and that of the 8096BH and offers guidelines for program development. Appendix A provides reference information for the 8XC196KC/KD instruction set. It includes descriptions of the instructions, hexadecimal opcodes, instruction lengths, execution times, and the relationships between Program Status Word (PSW) flags and the instructions. Appendix C provides a detailed description of the PSW.

3.1. OPERAND TYPES

The MCS-96 architecture supports a variety of data types likely to be useful in a control application. Where appropriate, this discussion uses the names adopted by the PLM-96 programming language. The name of an operand type is shown in all capitals, to avoid confusion. For example, a *BYTE* is an unsigned eight-bit variable, while a *byte* is an eight-bit unit of data of any type.

The following data types are available on the 8XC196KC/KD:

- BIT
- BYTE
- SHORT-INTEGER
- WORD
- INTEGER
- DOUBLE-WORD
- LONG-INTEGER

Table 3-1 provides an overview of these operand types. The remainder of this section discusses each one in detail.

Table 3-1. Operand Type Definitions

Operand Type	No. of Bits	Signed	Possible Values	Addressing Restrictions
BIT	1	No	True or False	As components of bytes
BYTE	8	No	0 through 255	None
SHORT-INTEGER	8	Yes	–128 through +127	None
WORD	16	No	0 through 65535	Even byte address
INTEGER	16	Yes	–32,768 through +32,767	Even byte address
DOUBLE-WORD *	32	No	0 through 4,294,967,295	Even byte address in on-chip Register File; evenly divisible by four **
LONG-INTEGER *	32	Yes	–2,147,483,648 through +2,147,483,647	Even byte address in on-chip Register File; evenly divisible by four **

* The 32-bit operands are supported only as the operand in shift operations, as the dividend in 32-by-16 divide operations, and as the product of 16-by-16 multiply operations.

** For consistency with Intel-provided software, you should adopt the PLM-96 conventions for addressing 32-bit operands. For more information, refer to "Software Standards and Conventions" in Chapter 2.

3.1.1. BIT Operand

A BIT is a single-bit operand that can take on the Boolean values, "true" and "false." In addition to the normal support for bits as components of BYTE and WORD operands, the 8XC196KC/KD provides a means for directly testing 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.

3.1.2. BYTE Operand

A BYTE is an unsigned, 8-bit variable that can take on values from 0 through 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; bit 0 is the least-significant bit. There are no alignment restrictions for BYTEs, so they may be placed anywhere in the MCS-96 address space.

3.1.3. SHORT-INTEGER Operand

A SHORT-INTEGER is an 8-bit, signed variable that can take on values from –128 through +127. Arithmetic operations that generate results outside the range of a SHORT-INTEGER set the overflow flags in the PSW. The numeric result is the same as the result of 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.

3.1.4. WORD Operand

A WORD is an unsigned, 16-bit variable that can take on values from 0 through 65535. Arithmetic and relational operators can be applied to WORD operands, but the result must be interpreted in modulo 65536 arithmetic. Logical operations on WORDs are applied bitwise. Bits within WORDs are labeled from 0 to 15; bit 0 is 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 that of its least-significant byte (the even byte address). WORD operations to odd addresses are not guaranteed to operate in a consistent manner.

3.1.5. INTEGER Operand

An INTEGER is a 16-bit, signed variable that can take on values from –32,768 through +32,767. Arithmetic operations that generate results outside the range of an INTEGER set the overflow flags in the PSW. The numeric result is the same as the result of the equivalent operation on WORD variables.

INTEGERs must be aligned at even byte boundaries in the MCS-96 address space. The least-significant byte of the INTEGER is in the even byte address, and the most-significant byte is in the next higher (odd) address. The address of an INTEGER is that of its least-significant byte (the even byte address). INTEGER operations to odd addresses are not guaranteed to operate in a consistent manner.

3.1.6. DOUBLE-WORD Operand

A DOUBLE-WORD is an unsigned, 32-bit variable that can take on values from 0 through 4,294,967,295. The MCS-96 architecture directly supports DOUBLE-WORD operands only as the operand in shift operations, as the dividend in 32-by-16 divide operations, and as the product of 16-by-16 multiply operations. For these operations, a DOUBLE-WORD variable must reside in the on-chip Register File and must be aligned at an address that is evenly divisible by four. The address of a DOUBLE-WORD is that of its least-significant byte (the even byte address). The least-significant word of the DOUBLE-WORD is always in the lower address, even when the data is in the stack. This means that the most-significant word must be pushed into the stack first.

DOUBLE-WORD operations that are not directly supported can be easily implemented with two WORD operations.

For consistency with Intel-provided software, you should adopt the PLM-96 conventions for addressing DOUBLE-WORD operands. (The PLM-96 conventions are discussed in "Software Standards and Conventions" in Chapter 2.)

3.1.7. LONG-INTEGER Operand

A LONG-INTEGER is a 32-bit, signed variable that can take on values from –2,147,483,648 through +2,147,483,647. The MCS-96 architecture directly supports DOUBLE-WORD operands only as the operand in shift operations, as the dividend in 32-by-16 divide operations, and as the product of 16-by-16 multiply operations. For these operations, a LONG-INTEGER variable must reside in the on-chip Register File and must be aligned at an address that is evenly divisible by four. The address of a LONG-INTEGER is that of its least-significant byte (the even byte address).

LONG-INTEGER operations that are not directly supported can be easily implemented with two INTEGER operations.

For consistency with Intel-provided software, you should adopt the PLM-96 conventions for addressing LONG-INTEGER operands. (The PLM-96 conventions are discussed in "Software Standards and Conventions" in Chapter 2.)

3.2. ADDRESSING MODES

Six basic addressing modes are used to access operands within the address space of the 80C196KC/KD:

- Register-Direct
- Indirect
- Indirect with Auto-Increment
- Immediate
- Short-Indexed
- Long-Indexed

Two other useful modes are Zero Register addressing and Stack Pointer Register addressing. Zero Register addressing combines the ZERO_REG with Long-Indexed addressing, allowing direct access to any location in memory. Stack Pointer Register addressing combines the SP with Indirect addressing to access the top of the stack and with Short-Indexed addressing to access data within the stack.

This section describes the addressing modes as they are handled by the hardware. An understanding of these details will help programmers to take full advantage of the architecture. The assembly language hides some of the details of how these addressing modes work. The "Assembly Language Addressing Mode Selections" section (see page 3-8) describes how the assembly language handles direct and indexed addressing modes.

3.2.1. Register-Direct Addressing

The Register-Direct addressing mode directly accesses a register from the 256-byte on-chip lower Register File. With windowing, this mode can also directly access the additional SFRs or the upper Register File (see Chapter 4, "Memory Partitions"). The register is selected by an 8-bit field within the instruction, and the register address 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 of Register-Direct Addressing:

```
ADD AX,BX,CX ; AX <-- BX + CX (ADD_3op)
MUL AX,BX    ; AX <-- AX * BX (MUL_2op)
INCB CL      ; CL <-- CL + 1
```

Definition of Temporary Registers:

AX, BX, and CX are 16-bit registers. CL is the low byte of CX.

3.2.2. Indirect Addressing

The Indirect addressing mode accesses 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 MCS-96 address space, including the Register File. An 8-bit field within the instruction selects the register that contains the indirect address. An instruction can contain only one indirect reference; any additional operands must be Register-Direct references.

Examples of Indirect Addressing:

```
LD AX,[AX]        ; AX <-- MEM_WORD(AX)
ADDB AL,BL,[CX]   ; AL <-- BL + MEM_BYTE(CX) (ADDB_3op)
POP [AX]          ; MEM_WORD(AX) <-- MEM_WORD(SP)
                  ; SP <-- SP + 2
```

Definition of Temporary Registers:

AX, CX are 16-bit registers. AL is the low byte of AX. CL is the low byte of CX.

3.2.3. Indirect with Auto-Increment Addressing

Indirect with Auto-Increment addressing mode is the same as the Indirect mode, except that the WORD variable that contains the indirect address is incremented after it is used to address the operand. The least-significant bit of a WORD register distinguishes between indirect addressing with or without auto-increment. If the instruction operates on a BYTE or

SHORT-INTEGER, the indirect address variable is incremented by one. If the instruction operates on a WORD or INTEGER, the indirect address variable is incremented by two.

Examples of Indirect with Auto-Increment Addressing:

```
LD AX,[BX]+        ; AX <-- MEM_WORD(BX)
                   ; BX <-- BX + 2
ADDB AL,BL,[CX]+ ; AL <-- BL + MEM_BYTE(CX)
                   ; CX <-- CX + 1 (ADDB_3op)
PUSH [AX]+         ; SP <-- SP - 2
                   ; MEM_WORD(SP) <-- MEM_WORD(AX)
                   ; AX <-- AX + 2
```

Definition of Temporary Registers:

AX, BX, CX are 16-bit registers. AL is the low byte of AX. BL is the low byte of BX.

3.2.4. Immediate Addressing

Immediate addressing mode allows an operand to be taken directly from a field in the instruction. For operations on BYTE or SHORT-INTEGER operands, this is an 8-bit field. For operations on WORD or INTEGER operands, it is a 16-bit field. An instruction can contain only one Immediate reference; any additional operands must be Register-Direct references.

Examples of Immediate Addressing:

```
ADD AX,#340      ; AX <-- AX + 340 (ADD_2op)
PUSH #1234H      ; SP <-- SP - 2
                 ; MEM_WORD(SP) <-- 1234H
DIVB AX,#10      ; AL <-- AX/10
                 ; AH <-- AX MOD 10
```

Definition of Temporary Registers:

AX is a 16-bit register. AL is the low byte and AH is the high byte of AX.

3.2.5. Short-Indexed Addressing

In Short-Indexed addressing mode, the address of one of the operands is calculated from two 8-bit fields. One 8-bit field in the instruction selects a WORD variable in the Register File, which contains an address. The second 8-bit field in the instruction stream is sign-extended and summed with the WORD variable to form the effective address of the operand. 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; any remaining operands must be Register-Direct references.

Examples of Short-Indexed Addressing:

```
LD AX,12[BX]          ; AX <-- MEM_WORD(BX+12)
MULB AX,BL,3[CX]      ; AX <-- BL * MEM_BYTE(CX+3)
                      ; (MULB_3op)
```

Definition of Temporary Registers:

AX, BX, CX are 16-bit registers. BL is the low byte of BX.

3.2.6. Long-Indexed Addressing

The Long-Indexed 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; any remaining operands must be Register-Direct references.

Examples of Long-Indexed Addressing:

```
AND AX,BX,TABLE[CX]     ; AX <-- BX AND MEM_WORD(TABLE+CX)
                        ; (AND_3op)
ST AX,TABLE[BX]         ; MEM_WORD(TABLE+BX) <-- AX
ADDB AL,BL,LOOKUP[CX]   ; AL <-- BL + MEM_BYTE(LOOKUP+CX)
                        ; (ADDB_3op)
```

Definition of Temporary Registers:

AX, BX, CX are 16-bit registers. AL is the low byte of AX. BL is the low byte of BX.

3.2.7. Zero Register Addressing

The first two bytes in the Register File constitute the Zero Register (ZERO_REG). These bytes are fixed at zero by the 80C196KC/KD hardware. In addition to providing a fixed source of the constant zero for calculations and comparisons, the Zero Register can be used as the WORD variable in a Long-Indexed reference. This combination of register selection and addressing mode allows any location in memory to be addressed directly. Since this mode uses indexed addressing, accesses are slower than register-direct accesses.

Examples of Zero Register Addressing:

```
ADD AX,1234[ZERO_REG]    ; AX <-- AX + MEM_WORD(1234) (ADD_2op)
POP 5678[ZERO_REG]       ; MEM_WORD(5678) <-- MEM_WORD(SP)
                         ; SP <-- SP + 2
```

Definition of Temporary Registers:

AX is a 16-bit register.

3.2.8. Stack Pointer Register Addressing

Bytes 18H and 19H of the lower Register File contain the system Stack Pointer (SP), which is addressed at 18H. Besides providing for convenient manipulation of the Stack Pointer, SP can also be used as the WORD variable in an Indirect reference to access the top of the stack or in a Short-Indexed reference to access data within the stack.

Examples of Stack Pointer Register Addressing:

```
PUSH [SP]    ; Duplicate TOP_OF_STACK
LD AX,2[SP] ; AX <-- NEXT_TO_TOP
```

Definition of Temporary Registers:

AX is a 16-bit register.

3.3. ASSEMBLY LANGUAGE ADDRESSING MODE SELECTIONS

The MCS-96 assembly language simplifies the choice of addressing modes. These features simplify the programming task and should be used wherever possible.

3.3.1. Direct Addressing

The assembly language chooses between Register-Direct and Zero Register addressing depending on the memory location of the operand. The programmer can simply refer to the operand by its symbolic name. If the operand is in the lower Register File, the assembly language chooses a Register-Direct reference. If the operand is elsewhere in memory, it chooses a Long-Indexed reference.

3.3.2. Indexed Addressing

The assembly language chooses between Short-Indexed and Long-Indexed addressing depending on the value of the index expression. If the value can be expressed in eight bits, the assembly language chooses a Short-Indexed reference. If the value is greater than eight bits, it chooses a Long-Indexed reference.

Memory Partitions 4

CHAPTER 4
MEMORY PARTITIONS

This chapter describes the addressable memory space within the 8XC196KC and 8XC196KD. Both devices have 64 Kbytes of addressable memory space, most of which is available for either program or data memory. Each memory location holds one byte.

Figure 4-1 shows the major memory partitions. The addresses differ because the 8XC196KD has twice as much RAM and One-Time-Programmable Read-Only Memory (OTPROM, a version of EPROM) space as the 8XC196KC.

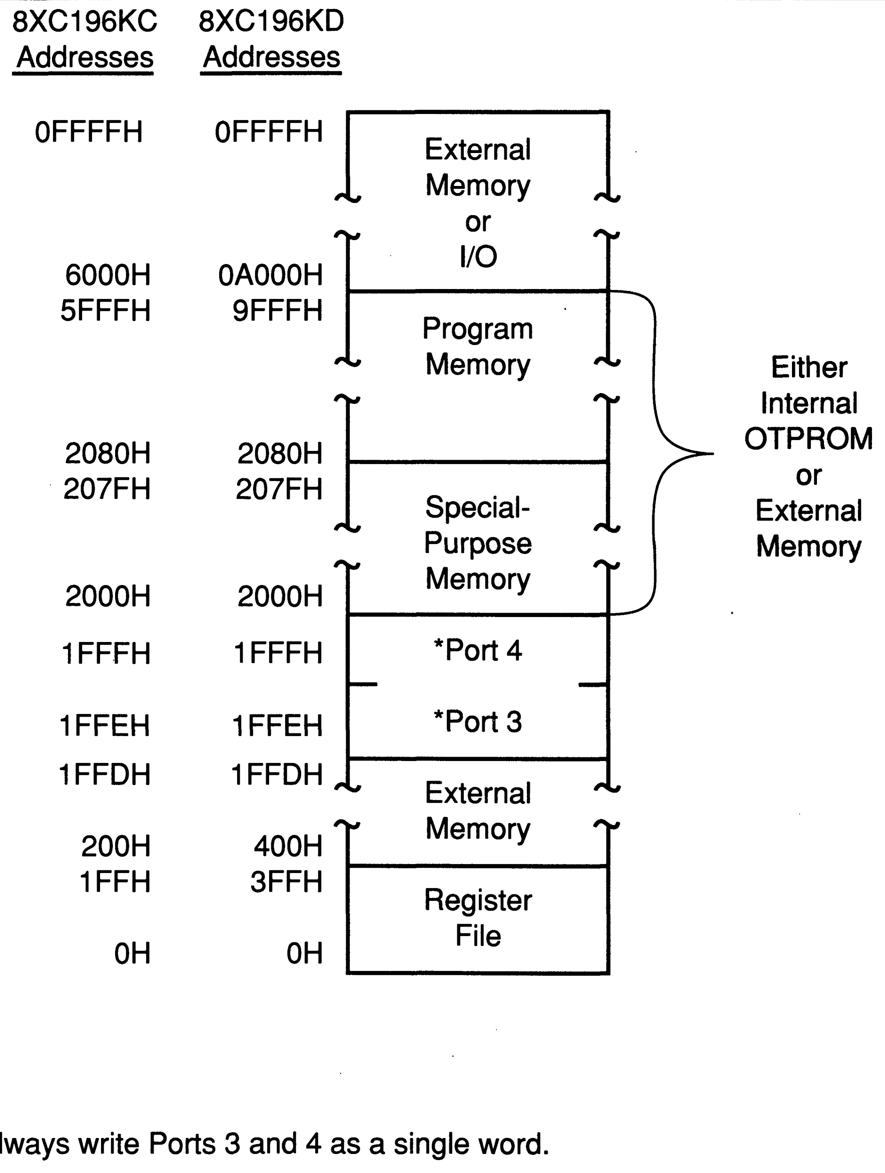


Figure 4-1. Top-Level Memory Map

Table 4-1 compares the 8XC196KC and 8XC196KD memory addresses and provides beginning and ending addresses in both hexadecimal and decimal. The sections that follow describe each memory partition.

Table 4-1. 8XC196KC/KD Top-Level Memory Addresses

	8XC196KC Address Range		8XC196KD Address Range	
Description	**Hexadecimal**	**Decimal**	**Hexadecimal**	**Decimal**
External Memory or I/O	6000H–0FFFFH	24576–65635	0A000H–0FFFFH	40960–65535
Program Memory (Note 1)	2080H–5FFFH	8320–24575	2080H–9FFFH	8320–40956
Special-Purpose Memory (Note 1)	2000H–207FH	8192–8319	2000H–207FH	8192–8319
Ports 3 and 4 (Note 2)	1FFEH–1FFFH	8190–8191	1FFEH–1FFFH	8190–8191
External Memory	200H–1FFDH	512–8189	400H–1FFDH	1024–8189
Register File (includes SFRs)	0H–1FFH	0–511	0H–3FFH	0–1023

NOTES:

1. Located in either internal OTPROM or external memory.
2. Ports 3 and 4 are word-addressable only.

4.1. EXTERNAL MEMORY PARTITIONS

The top of the memory map and the partition directly above the Register File are always assigned to external memory or I/O (see Figure 4-1 or Table 4-1). The 8XC196KC/KD can interface with a variety of external memory devices. See Chapter 13, "Interfacing with External Memory," for details.

4.2. PORTS 3 AND 4

Ports 3 and 4 are read and written as a word at memory location 1FFEH (Port 3 is the low byte; Port 4 is the high byte). These ports function as either I/O ports, the system address/data bus, or a combination of both, depending upon the value of the EA# signal at reset. See Chapter 6, "I/O Ports," for more information.

4.3. PROGRAM AND SPECIAL-PURPOSE MEMORY

Program memory and special-purpose memory occupy a 16-Kbyte memory partition in the 8XC196KC and a 32-Kbyte memory partition in the 8XC196KD (see Figure 4-1 or Table 4-1). These partitions can reside in either internal OTPROM or external memory. Chapter 14, "Programming the Nonvolatile Memory," provides information about programming the OTPROM.

4.3.1. Selecting Internal or External Memory Mapping

An access to the program memory or special-purpose memory address range is directed either to internal OTPROM or external memory, but not both. The value of the External Access (EA#) pin during the rising edge of the RESET# signal determines whether the access is internal or external. This value is latched into the device and cannot be changed unless the device is reset. If EA# is low, the internal OTPROM is inaccessible and all accesses to this address range are directed to external memory. If EA# is high, all accesses to this address range are directed to the internal OTPROM.

NOTE

The EA# input must be tied low for CPU-only devices to function properly.

4.3.2. Program Memory

Program memory is located at addresses 2080H–5FFFH in the 8XC196KC and addresses 2080H–9FFFH in the 8XC196KD. When the device is reset, the CPU fetches and then executes the instruction from location 2080H. (See Chapter 11, "Minimum Hardware Considerations," for more information about reset.)

NOTE

Since the default value in ROM/OTPROM locations is 0FFH, we recommend that you write 0FFH to unused program memory locations.

4.3.3. Special-Purpose Memory

Special-purpose memory is located at addresses 2000H–207FH (see Figure 4-2). It contains several reserved memory locations, vectors for both the Peripheral Transaction Server (PTS) and standard interrupts, a security key, and the Chip Configuration Byte (CCB). Table 4-2 lists the special-purpose memory addresses and provides beginning and ending addresses in both hexadecimal and decimal.

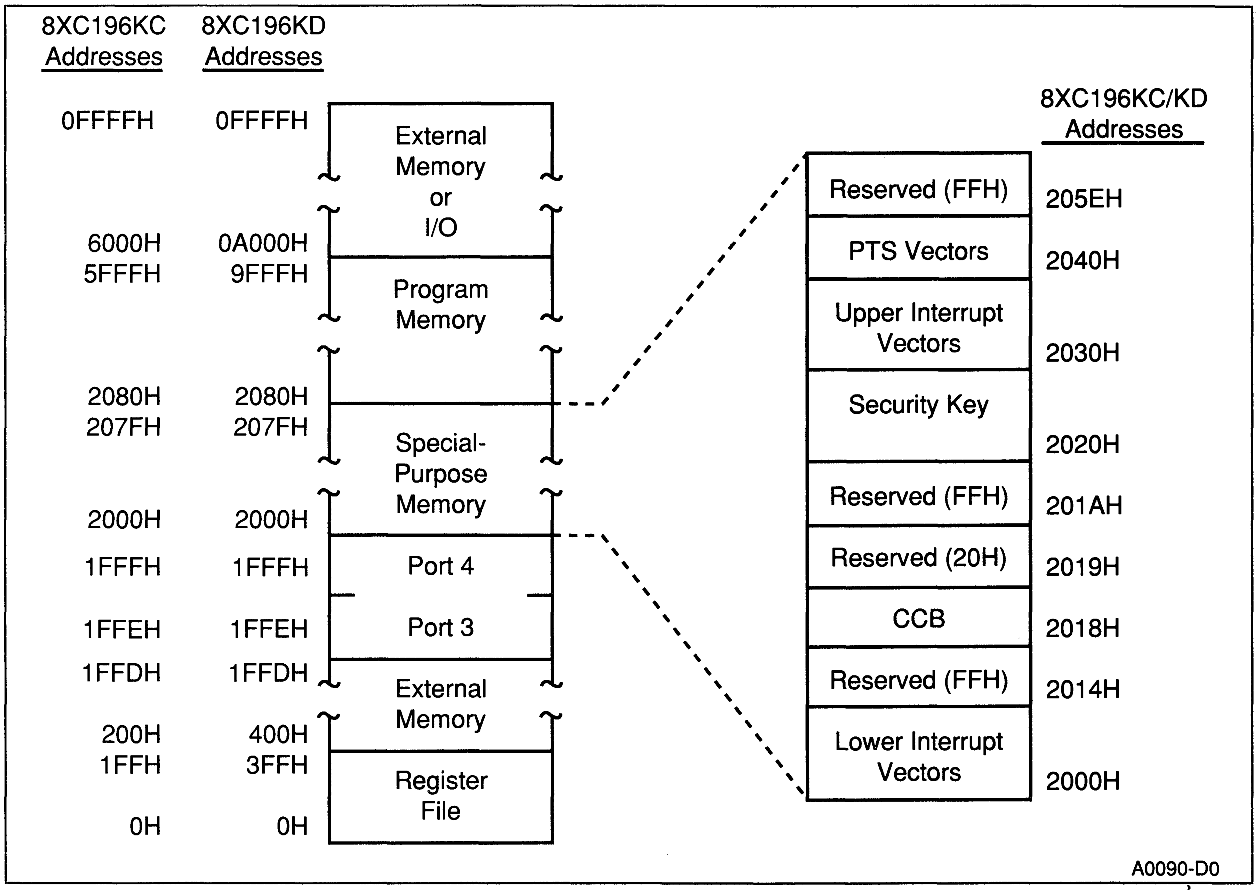


Figure 4-2. 8XC196KC/KD Special-Purpose Memory Map

4.3.3.1. RESERVED MEMORY LOCATIONS

Several memory locations are reserved for testing or future product use. The user program must not read or write these reserved memory locations. The contents and/or function of these locations may change in future revisions, so software that uses reserved locations may not function properly.

When programming the OTPROM, always write 20H to reserved address 2019H and 0FFH to all other reserved addresses.

4.3.3.2. INTERRUPT VECTORS

The Upper and Lower Interrupt Vectors contain the addresses of the interrupt service routines. The Peripheral Transaction Server (PTS) vectors contain the addresses of the PTS control blocks. See Chapter 5, "Interrupts," for more information.

Table 4-2. 8XC196KC/KD Special-Purpose Memory Addresses

Description	8XC196KC/KD Address Range	
	Hexadecimal	Decimal
Reserved (must contain 0FFH)	205EH–207FH	8286–8319
PTS Vectors	2040H–205DH	8256–8285
Upper Interrupt Vectors	2030H–203FH	8240–8255
Security Key	2020H–202FH	8224–8239
Reserved (must contain 0FFH)	201AH–201FH	8218–8223
Reserved (must contain 20H)	2019H	8217
CCB	2018H	8216
Reserved (must contain 0FFH)	2014H–2017H	8212–8215
Lower Interrupt Vectors	2000H–2013H	8192–8211

4.3.3.3. SECURITY KEY

The security key protects the 8XC196KC/KD against unauthorized reading and writing of OTPROM. If the security lock bits in the Chip Configuration Register (CCR.6 and CCR.7) are cleared, each byte of the 16-byte programmable security code is compared to a corresponding byte in external memory. If the external data does not match the security key, the device does not enter into the requested programming mode. See Chapter 14, "Programming the Nonvolatile Memory," for more information about the security key.

4.3.3.4. CHIP CONFIGURATION BYTE

The Chip Configuration Byte (CCB) is the first byte fetched from memory after a device reset. The reset sequence loads the CCB into an internal register called the Chip Configuration Register (CCR). The CCR controls internal memory protection, internal READY mode, bus control signals, bus-width options, and Powerdown mode (see Appendix C, "8XC196KC/KD Registers").

4.4. REGISTER FILE

The Register File is divided into an upper and a lower Register File (see Figure 4-3). The upper Register File contains general-purpose register RAM. The lower Register File contains general-purpose register RAM, the Stack Pointer (SP), and CPU Special Function Registers (SFRs). Table 4-3 compares the 8XC196KC and 8XC196KD Register File memory addresses and provides beginning and ending addresses in both hexadecimal and decimal.

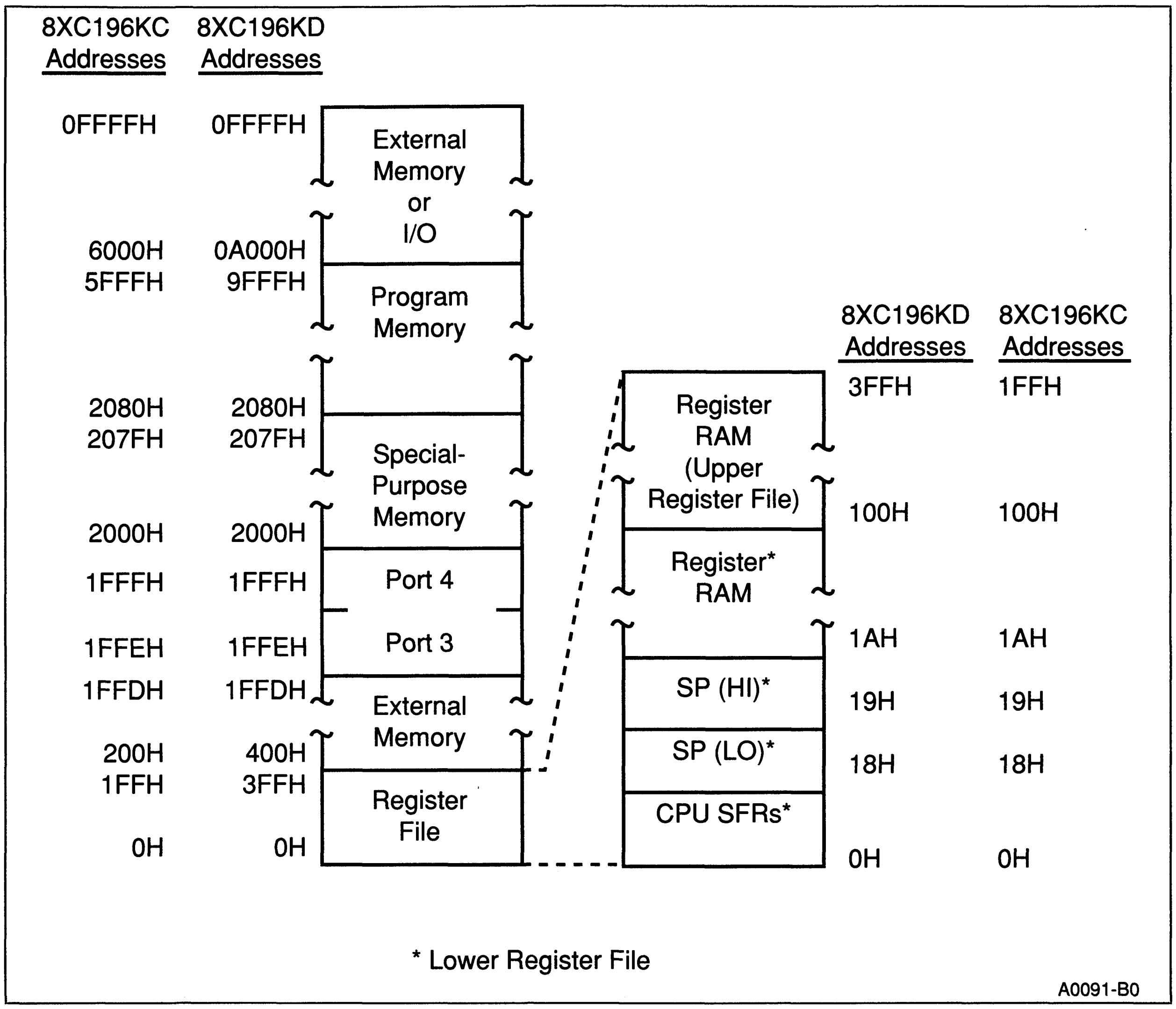


Figure 4-3. Register File Memory Map

NOTE

The Register File must not contain code. An attempt to execute instructions from Register File memory locations (0H–01FFH in the 8XC196KC or 0H–03FFH in the 8XC196KD) causes the memory controller to automatically fetch instructions from external memory. This section of external memory is reserved for use by Intel development tools.

Table 4-3. 8XC196KC/KD Register File Memory Addresses

	8XC196KC Address Range		8XC196KD Address Range	
Description	Hexadecimal	Decimal	Hexadecimal	Decimal
Upper Register File (Note 1)	100H–1FFH	256–511	100H–3FFH	256–1023
General-purpose register RAM (Note 2)	1AH–0FFH	26–255	1AH–0FFH	26–255
Stack Pointer (SP) (Note 2)	18H–19H	24–25	18H–19H	24–25
Special Function Registers (SFRs) (Note 2)	0H–17H	0–23	0H–17H	0–23

NOTES:

1. Contains general-purpose register RAM (accessed by indirect or indexed addressing unless vertical windowing is used).
2. Located in the lower Register File (accessed by either register-direct, indirect, or indexed addressing).

4.4.1. General-Purpose Register RAM

Locations 1AH–0FFH contain general-purpose register RAM. The stack pointer locations, 18H and 19H, can also be used as general-purpose register RAM when stack operations are not being performed. The RALU can access this memory directly, using register-direct addressing. The upper Register File also contains general-purpose register RAM (100H–1FFH in the 8XC196KC and 100H–3FFH in the 8XC196KD). The RALU normally accesses this memory using indirect or indexed addressing. *Vertical windowing* is a technique that enables the RALU to use register-direct addressing to access the RAM in the upper Register File. (See Chapter 3, "Data Types and Addresses," for a description of differences between register-direct and indexed addressing.) Vertical windowing provides fast context switching of interrupt tasks and faster program execution. See "Vertical Windowing" on page 4-12.

4.4.2. Stack Pointer (SP)

Memory locations 18H and 19H contain the stack pointer (SP). The SP contains the address of the stack. The SP must point to a word (even) address that is two bytes greater than the desired starting address. Before the CPU executes a subroutine call or interrupt service routine, it decrements the SP twice and then copies (PUSHes) the address of the next instruction from the program counter onto the stack. It then loads the address of the subroutine or interrupt service routine into the program counter. After it completes the subroutine or interrupt service routine, the CPU executes a return from subroutine instruction (RET) and loads (POPs) the contents of the top of the stack (that is, the return address) into the program counter.

Subroutines may be nested. That is, each subroutine may call other subroutines. The CPU pushes the contents of the program counter onto the stack each time it executes a subroutine call. The stack grows downward as entries are added. The only limit to the nesting depth is the amount of available memory. As the CPU returns from each nested subroutine, it pops the address off the top of the stack and the next return address moves to the top of the stack.

When stack operations are not being performed, the SP locations may be used as general-purpose register RAM.

4.4.2.1. INITIALIZING THE STACK POINTER

The user program must load a word-aligned (i.e., even) address into the stack pointer. Select an address that is two bytes greater than the desired starting address because the stack pointer is automatically decremented before the CPU pushes the first byte of the return address onto the stack. Remember that the stack grows downward, so allow sufficient room for the maximum number of stack entries. The stack may be located in either the internal Register File or external RAM.

The following example initializes the top of the 8XC196KD's upper Register File as the stack.

```
LD SP, #400H                  :Load stack pointer
```

4.4.3. Special Function Registers

Locations 00H–17H provide access to the CPU Special Function Registers (SFRs) through three *horizontal windows* (HWindow 0, 1, and 15). (See "Horizontal Windowing" on page 4-8.) The RALU has direct control of all peripheral modules, except Ports 3 and 4, through the SFRs. Appendix C, "8XC196KC/KD Registers," describes each register in detail.

When using an SFR as a base or index register for indirect or indexed operations, be aware that the contents of the SFRs are not always predictable. External events can change the contents of SFRs, and some SFRs are cleared when read.

The functions of many SFRs change depending upon whether they are being read from or written to. For this reason, never use an SFR as an operand in a read-modify-write instruction (e.g., XORB, AD_RESULT).

Do not use reserved SFRs; write zeros to them or leave them in their default state. When read, reserved bits and SFRs will return undefined values.

4.5. HORIZONTAL WINDOWING

The 8XC196KC/KD uses a *horizontal windowing* scheme that swaps three 24-byte memory blocks (HWindow 0, 1, and 15) within the lowest 24 bytes of the lower Register File (see Figure 4-4). Each HWindow provides read or write access to a unique combination of SFRs. Some registers are accessed as a single byte; others are accessed as a word (two bytes). Some registers, such as the Window Select Register (WSR, 14H), are accessible in all three HWindows. Others must be written in one HWindow and read in another. Appendix C, "8XC196KC/KD Registers," provides detailed descriptions of each register.

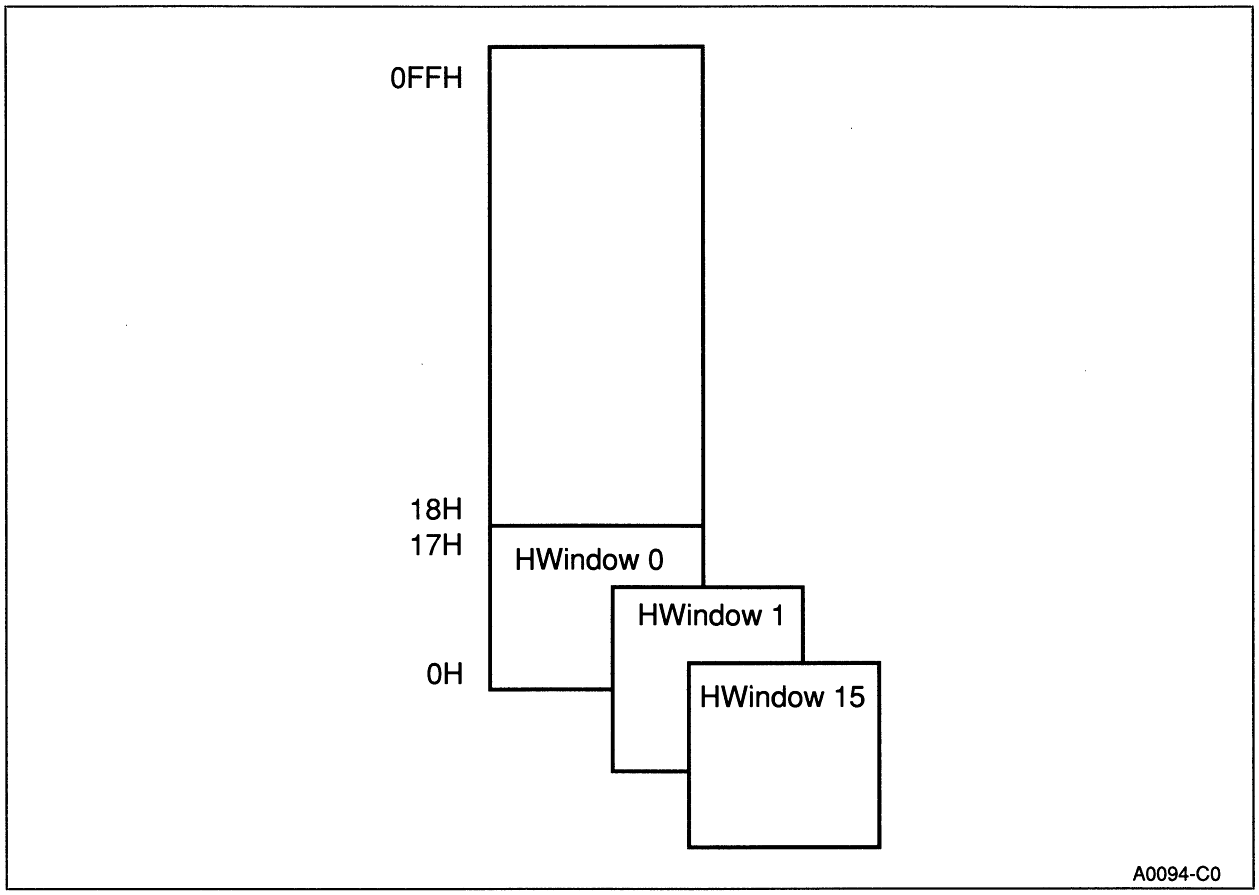


Figure 4-4. Horizontal Windowing

4.5.1. Selecting an HWindow

The Window Select Register (WSR, 14H) provides access to HWindows and VWindows (see "Vertical Windowing" on page 4-12). To select an HWindow, write the number of the desired window into WSR.0–WSR.3 and clear WSR.4–WSR.6. Only HWindows 0, 1, and 15 are available. All other HWindows are reserved. Table 4-4 shows the appropriate WSR contents for each HWindow selection. See Appendix C, "8XC196KC/KD Registers," for a complete description of the WSR.

Table 4-4. HWindow Selections

HWindow	WSR Contents
0	X000 0000B = 00H
1	X000 0001B = 01H
15	X000 1111B = 0FH

4.5.2. HWindow 0

HWindow 0 is the default window. It provides read access to 19 registers and write access to 21 registers (see Figure 4-5). Some registers (e.g., INT_MASK1) can be both read and written within HWindow 0. Others (e.g., IOS1) can be either read or written, but not both, within HWindow 0. For these registers, select HWindow 15 to perform the complementary function.

4.5.3. HWindow 1

HWindow 1 provides read and write access to 12 registers (see Figure 4-5). Most support those features that differentiate the 8XC196KC and 8XC196KD from other, earlier members of the MCS-96 family. Some registers are also accessible in both HWindow 0 and HWindow 15.

	HWINDOW 0 (Read)	HWINDOW 0 (Write)		HWINDOW 1 (Read/Write)
17H	IOS2	PWM0_CONTROL	17H	PWM2_CONTROL
16H	IOS1	IOC1	16H	PWM1_CONTROL
15H	IOS0	IOC0	15H	Reserved
14H	WSR	WSR	14H	WSR
13H	INT_MASK1	INT_MASK1	13H	INT_MASK1
12H	INT_PEND1	INT_PEND1	12H	INT_PEND1
11H	SP_STAT	SP_CON	11H	Reserved
10H	IOPORT2	IOPORT2	10H	Reserved
0FH	IOPORT1	IOPORT1	0FH	Reserved
0EH	IOPORT0	BAUD_RATE	0EH	Reserved
0DH	TIMER2 (HI)	TIMER2 (HI)	0DH	Reserved
0CH	TIMER2 (LO)	TIMER2 (LO)	0CH	IOC3
0BH	TIMER1 (HI)	IOC2	0BH	Reserved
0AH	TIMER1 (LO)	WATCHDOG	0AH	Reserved
09H	INT_PEND	INT_PEND	09H	INT_PEND
08H	INT_MASK	INT_MASK	08H	INT_MASK
07H	SBUF (RX)	SBUF (TX)	07H	PTSSRV (HI)
06H	HSI_STATUS	HSO_COMMAND	06H	PTSSRV (LO)
05H	HSI_TIME (HI)	HSO_TIME (HI)	05H	PTSSEL (HI)
04H	HSI_TIME (LO)	HSO_TIME (LO)	04H	PTSSEL (LO)
03H	AD_RESULT (HI)	HSI_MODE	03H	AD_TIME
02H	AD_RESULT (LO)	AD_COMMAND	02H	Reserved
01H	ZERO_REG (HI)	ZERO_REG (HI)	01H	ZERO_REG (HI)
00H	ZERO_REG (LO)	ZERO_REG (LO)	00H	ZERO_REG (LO)

Figure 4-5. HWindow 0 and HWindow 1

4.5.4. HWindow 15

HWindow 15 provides access to the same registers that HWindow 0 does, except for bytes 0CH–10H. Bytes 0CH–0DH access the 16-bit TIMER2 register in HWindow 0, but they access the 16-bit T2CAPTURE register in HWindow 15. Bytes 0EH–10H access the IOPORT0, IOPORT1, and IOPORT2 registers in HWindow 0, but they are reserved in HWindow 15. Those registers that are read-only in HWindow 0 are write-only in HWindow 15, and vice versa.

	HWINDOW 15 (Read)	HWINDOW 15 (Write)
17H	PWM0_CONTROL	IOS2
16H	IOC1	IOS1
15H	IOC0	IOS0
14H	WSR	WSR
13H	INT_MASK1	INT_MASK1
12H	INT_PEND1	INT_PEND1
11H	SP_CON	SP_STAT
10H	Reserved	Reserved
0FH	Reserved	Reserved
0EH	Reserved	Reserved
0DH	T2CAPTURE (HI)	T2CAPTURE (HI)
0CH	T2CAPTURE (LO)	T2CAPTURE (LO)
0BH	IOC2	TIMER1 (HI)
0AH	WATCHDOG	TIMER1 (LO)
09H	INT_PEND	INT_PEND
08H	INT_MASK	INT_MASK
07H	SBUF (TX)	SBUF (RX)
06H	HSO_COMMAND	HSI_STATUS
05H	HSO_TIME (HI)	HSI_TIME (HI)
04H	HSO_TIME (LO)	HSI_TIME (LO)
03H	HSI_MODE	AD_RESULT (HI)
02H	AD_COMMAND	AD_RESULT (LO)
01H	ZERO_REG (HI)	ZERO_REG (HI)
00H	ZERO_REG (LO)	ZERO_REG (LO)

Figure 4-6. HWindow 15

4.6. VERTICAL WINDOWING

Vertical windowing maps sections of the upper Register File into the upper locations of the lower Register File, in 32-, 64-, or 128-byte increments known as VWindows. The 8XC196KC has sixteen 32-byte VWindows, eight 64-byte VWindows, and four 128-byte VWindows. Since the 8XC196KD has twice as much RAM as the 8XC196KC, it also has twice as many VWindows.

Figure 4-7 is an example of a 128-byte VWindow.

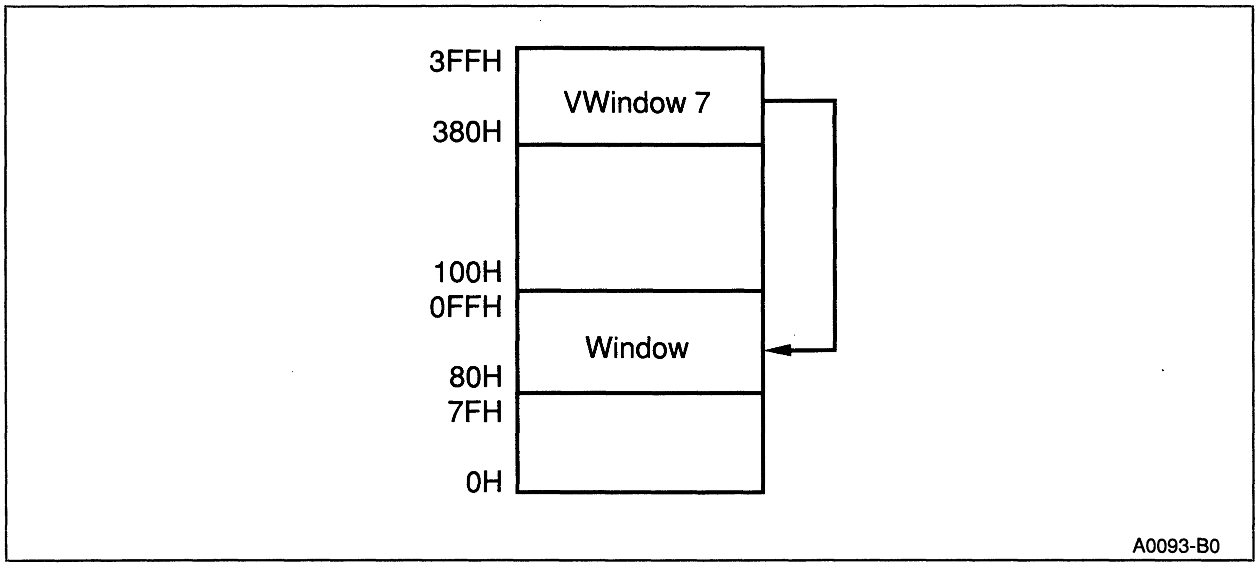


Figure 4-7. Vertical Windowing

4.6.1. Selecting a VWindow

The Window Select Register (WSR, 14H) provides access to HWindows and VWindows. Set WSR.4, WSR.5, or WSR.6 to select a 128-, 64-, or 32-byte VWindow respectively (see Figure 4-8). Write the VWindow number into lower WSR bits. Appendix C, "8XC196KC/KD Registers," provides a complete description of the WSR, with tables that list the appropriate WSR contents for each VWindow selection. (To select the vertical window shown in Figure 4-7, load 17H into the WSR.)

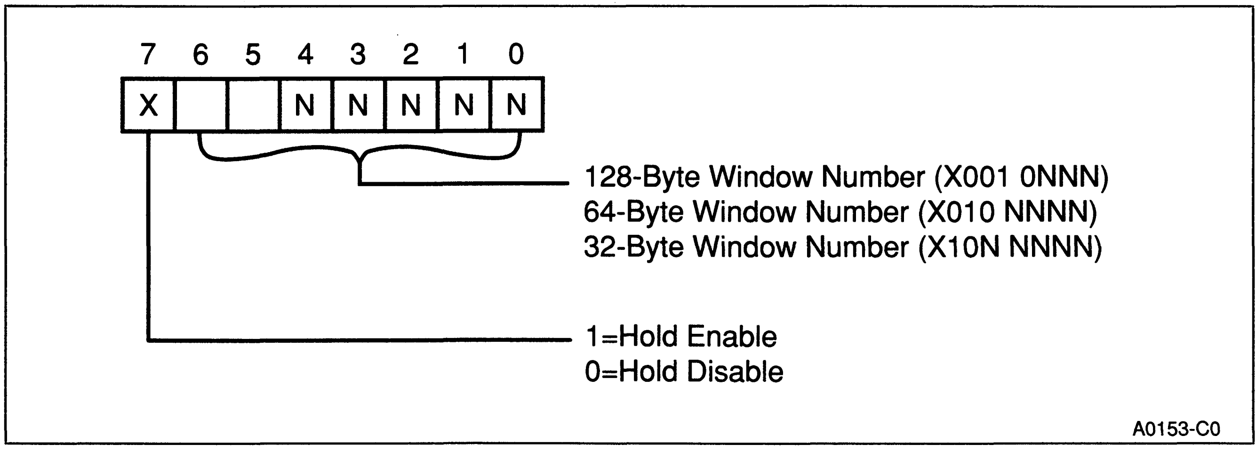

7 6 5 4 3 2 1 0
X N N N N N
128-Byte Window Number (X001 0NNN)
64-Byte Window Number (X010 NNNN)
32-Byte Window Number (X10N NNNN)
1=Hold Enable
0=Hold Disable
A0153-C0

Figure 4-8. Window Select Register Bit Settings

4.6.2. Vertical Windowing and Addressing Modes

When vertical windowing is enabled:

- a register-direct instruction that uses an address within the lower Register File window actually accesses the VWindow in the upper Register File;
- an indirect or indexed instruction that uses either an address within the lower Register File window or the VWindow accesses the actual location in memory.

NOTE

Indirect shift operations will not perform correctly if WSR = X100 0000.

The following sample code illustrates the difference between register-direct and indexed addressing when using vertical windowing.

```
PUSHA                   ;pushes the contents of WSR onto the stack
LDB WSR, #17H           ;select VWindow 7, a 128-byte block
                        ;The next instruction uses register-direct addr
ADD 40H, 80H            ;mem_word(40H)←mem_word(40H) + mem_word(380H)
                        ;
                        ;The next two instructions use indirect addr
ADD 40H, 80H[0]         ;mem_word(40H)←mem_word(40H) + mem_word(80H + 0)
                        ;
ADD 40H, 380H[0]        ;mem_word(40H)←mem_word(40H) + mem_word(380H + 0)
                        ;
POPA                    ;reloads the previous contents into WSR
```

Interrupts 5

CHAPTER 5
INTERRUPTS

A microcontroller's primary function is to provide real-time control of an instrument or device. The interrupt control circuitry within a microcontroller permits real-time events to control program flow. When an event generates an interrupt, the CPU services the interrupt before executing the next instruction. An internal peripheral, an external signal, or an instruction can request an interrupt. In the simplest case, the 8XC196KC/KD receives the request, performs the service, and returns to the task that was interrupted. This chapter describes the interrupt control circuitry, priority scheme, and timing. It also describes the three special interrupts and explains interrupt programming and control.

5.1. INTERRUPT PROCESSING

The 8XC196KC/KD provides two interrupt service options: software interrupt service routines via the Interrupt Controller and microcoded hardware interrupt processing via the Peripheral Transaction Server (PTS). You can select either option for each of the maskable interrupts. (See "Selecting Either PTS or Standard Interrupt Service" on page 5-10.) The nonmaskable interrupts (NMI, Software Trap, and Unimplemented Opcode) are always serviced by interrupt service routines. Figure 5-1 illustrates the interrupt processing flow.

5.1.1. Interrupt Controller

The Interrupt Controller services interrupts with software interrupt service routines. When the hardware detects an interrupt, it generates and executes a special interrupt call. This pushes the contents of the program counter onto the stack and then loads it with the contents of the appropriate interrupt vector. The Upper and Lower Interrupt Vectors in Special-Purpose memory (see Chapter 4, "Memory Partitions") contain the addresses of the interrupt service routines. The CPU executes the interrupt service routine. Upon completion of the service routine, the program counter is reloaded from the stack and program execution continues.

5.1.2. Peripheral Transaction Server (PTS)

The Peripheral Transaction Server (PTS) is a microcoded hardware interrupt handler. It can be used in place of a standard interrupt service routine for each of the maskable interrupts. The PTS services interrupts with less overhead; it does not modify the stack or the PSW, and it allows normal instruction flow to continue. For these reasons, the PTS can service an interrupt in the time required to execute a single instruction.

The PTS operates in five special microcoded modes that enable the PTS to complete the specific tasks in much less time than an interrupt service routine. See the "PTSCON" section on page 5-16 for a description of the PTS modes.

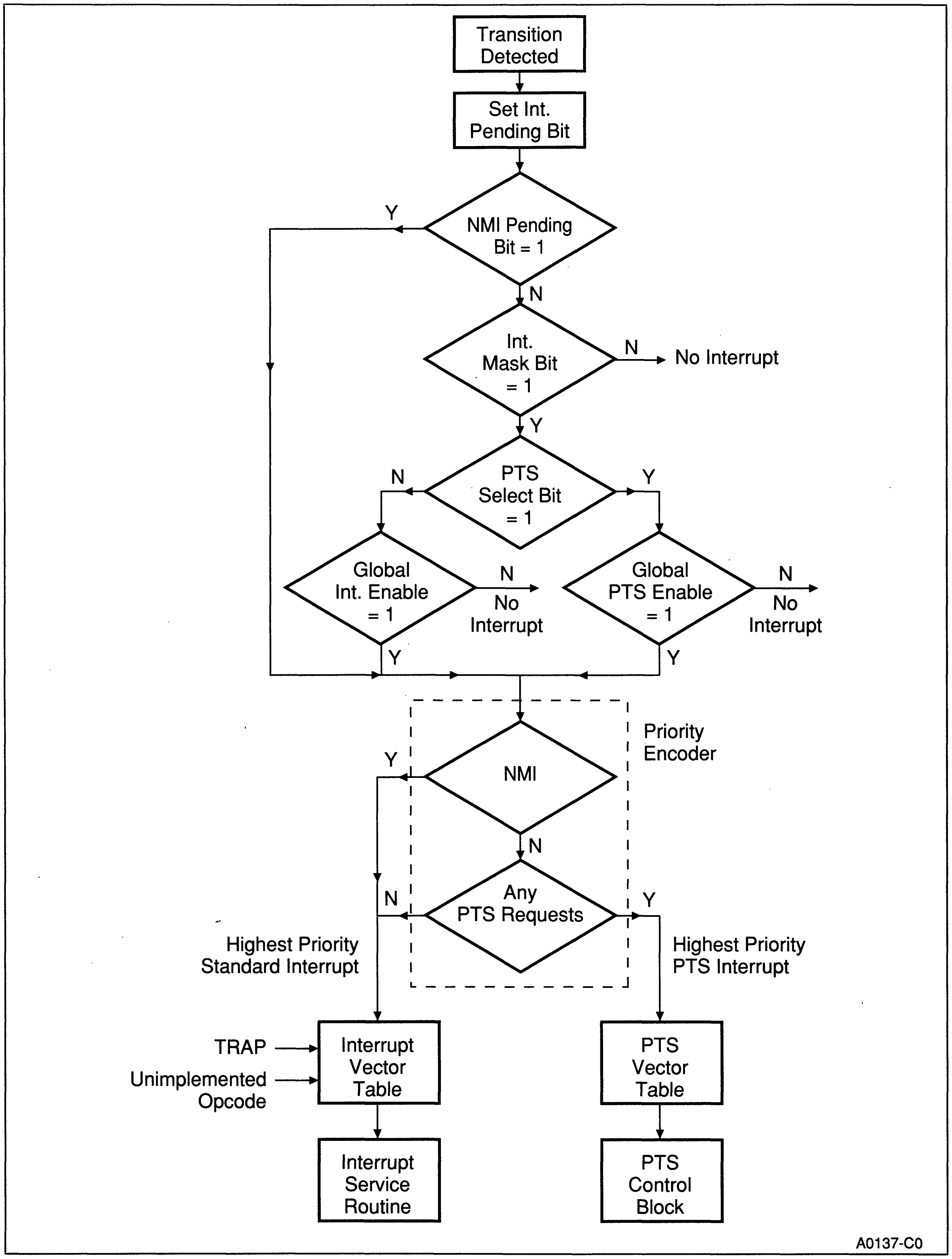


Figure 5-1. 8XC196KC/KD Flow Diagram for PTS and Standard Interrupts

Each PTS interrupt requires a block of data called the PTS Control Block (PTSCB). When a PTS interrupt occurs, the priority encoder selects the appropriate vector and fetches the PTS Control Block (PTSCB). The PTSCB determines the mode, the total number of transfers (if applicable), the total number of cycles that will execute before the PTS requires servicing, and the source and/or destination of data transfers (if applicable). Each PTS interrupt generates one PTS cycle. Figure 5-2 shows the PTS cycle flow.

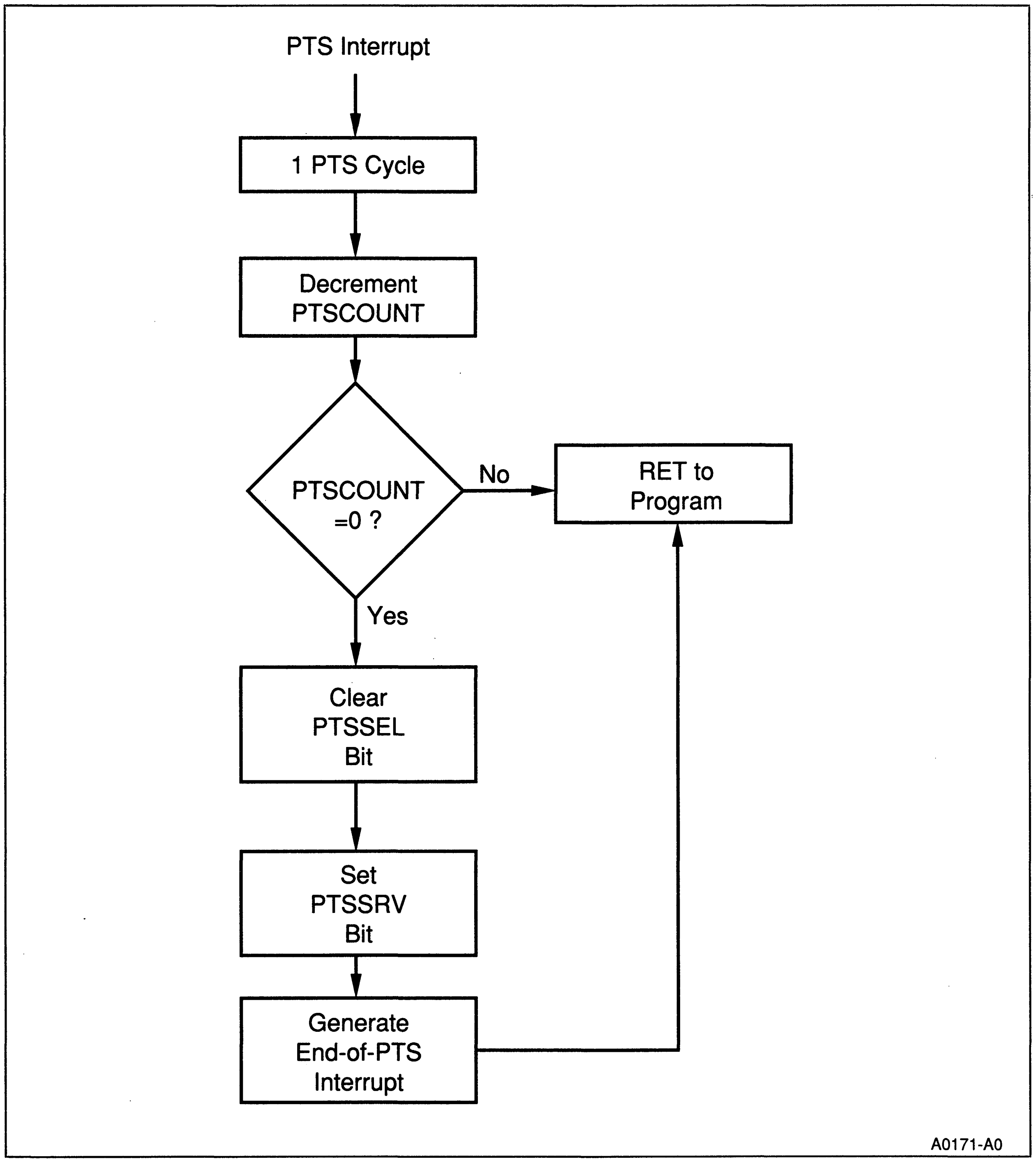


Figure 5-2. PTS Cycle Flow Diagram

5.2. INTERRUPT PRIORITIES

The Unimplemented Opcode and Software Trap interrupts are not prioritized; they go directly to the Interrupt Controller for servicing. The Interrupt Controller selects the corresponding vector location in Special-Purpose memory (see Chapter 4, "Memory Partitions"). The vector contains the starting address of the interrupt service routine.

The Priority Encoder determines the priority of all other pending interrupt requests. Table 5–1 shows the default interrupt priorities (15 is highest and 1 is lowest). NMI has the highest priority of all prioritized interrupts. If an NMI is pending, the Priority Encoder selects it as the highest priority request, and the Interrupt Controller selects the corresponding vector location in Special-Purpose memory (see Chapter 4). The vector contains the starting address of the corresponding interrupt service routine.

Any PTS interrupt request has a higher priority than all maskable standard interrupt requests. If no NMI request is pending, the Priority Encoder determines the highest priority PTS interrupt service request, and the Interrupt Controller selects the corresponding PTS vector location in Special-Purpose memory. The vector contains the starting address of the corresponding PTS control block (PTSCB).

If no NMI or PTS request is pending, the Priority Encoder determines the highest priority standard interrupt request, and the Interrupt Controller selects the corresponding vector location in Special-Purpose memory. The vector contains the starting address of the corresponding interrupt service routine.

5.2.1. Modifying Interrupt Priorities

The software can modify the default priorities of maskable interrupts by controlling the interrupt mask registers (INT_MASK and INT_MASK1). For example, you can specify which interrupts, if any, can interrupt an interrupt service routine or PTS cycle. The following code shows one way to prevent all interrupts, except EXTINT (priority 7), from interrupting a Receive interrupt service routine (priority 9).

```
SERIAL_RI_ISR:
     pusha                              ; Save PSW, INT_MASK, INT_MASK1, & WSR
     di                                 ; Disable all interrupts except EXTINT
     ldb int_mask1, #00100000B          ;
     ei                                 ; Enable interrupt servicing
                                        ;
                                        ; Service the RI interrupt
                                        ;
     popa                               ; Restore PSW, INT_MASK, INT_MASK1, &
                                        ; WSR regs
     ret
```

Note that location 2032H in the interrupt vector table would be loaded with the value of the label SERIAL_RI_ISR, and that the Receive interrupt must be enabled for this routine to execute.

Table 5-1. 8XC196KC/KD Interrupt Vector Sources, Locations, and Priorities

Number	Interrupt Vector	Source(s)	Interrupt Vector Location	PTS Vector Location	Priority (1)
Special	Unimplemented Opcode	Unimplemented Opcode	2012H	—	—
Special	Software Trap	TRAP Instruction	2010H	—	—
INT15	NMI **(2)**	NMI	203EH	—	15
Each of the following, maskable interrupts can be assigned to the PTS. Any PTS interrupt has priority over all other maskable interrupts.					
INT14	HSI FIFO Full	HSI FIFO Full	203CH	205CH	14
INT13	EXTINT1 **(2)**	P2.2	203AH	205AH	13
INT12	Timer 2 Overflow	Timer 2 Overflow	2038H	2058H	12
INT11	Timer 2 Capture **(2)**	Timer 2 Capture	2036H	2056H	11
INT10	HSI FIFO 4	HSI FIFO Fourth Entry	2034H	2054H	10
INT09	Receive	RI Flag **(3)**	2032H	2052H	9
INT08	Transmit	TI Flag **(3)**	2030H	2050H	8
INT07	EXTINT **(2)**	P2.2 or P0.7	200EH	204EH	7
INT06	Serial Port	RI Flag and TI Flag **(4)**	200CH	204CH	6
INT05	Software Timer	Software Timer 0–3 Timer 2 Reset A/D Conversion Start	200AH	204AH	5
INT04	HSI.0 Pin **(2)**	HSI.0	2008H	2048H	4
INT03	High Speed Outputs	HSO.0–HSO.5	2006H	2046H	3
INT02	HSI Data Available	HSI FIFO Full or HSI Holding Reg. Loaded	2004H	2044H	2
INT01	A/D Conversion Complete	A/D Conversion Complete	2002H	2042H	1
INT00	Timer Overflow	Timer 1 or Timer 2	2000H	2040H	0

NOTES:

1. The Unimplemented Opcode and Software Trap interrupts are not prioritized. They go directly to the Interrupt Controller for servicing. NMI has the highest priority of all prioritized interrupts. Any PTS interrupt has priority over all other maskable interrupts.
2. These interrupts can be configured to function as independent, external interrupts.
3. If the Serial interrupt is masked and the Receive and Transmit interrupts are enabled, the RI flag and TI flag generate separate Receive and Transmit interrupts. If 8096BH compatibility is not an issue, this configuration is preferred.
4. If the Receive and Transmit interrupts are masked and the Serial interrupt is enabled, both RI flag and TI flag generate a Serial Port interrupt. This configuration provides compatibility with the 8096BH.

All 8XC196KC/KD interrupt service routines are handled in the following manner:

1. After the hardware detects and prioritizes an interrupt request, it generates and executes a special interrupt call. This pushes the program counter onto the stack and then loads it with the contents of the vector corresponding to the highest priority, pending, unmasked interrupt. The hardware will not allow another interrupt call until after the first instruction of the interrupt service routine is executed.

2. The PUSHA instruction, which is guaranteed to execute, saves the contents of the PSW, INT_MASK1, and Window Select Register (WSR) onto the stack and then clears the PSW and INT_MASK1. In addition to the arithmetic flags, the PSW contains INT_MASK register, the global interrupt enable bit (I), and the PTS enable bit (PSE). Clearing the PSW and INT_MASK1 register effectively masks all maskable interrupts, disables standard interrupt servicing, and disables the PTS. The PUSHA instruction inhibits interrupts calls until after the next instruction executes.

3. The LDB INT_MASK1 instruction enables those interrupts that you choose to allow to interrupt the service routine. In this example, only EXTINT can interrupt the Receive interrupt service routine. By enabling or disabling interrupts, the software establishes its own interrupt servicing priorities.

4. The EI instruction re-enables interrupt processing and inhibits interrupt calls until after the next instruction executes.

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

6. At the end of the service routine, the POPA instruction restores the original contents of the PSW, INT_MASK1, and WSR registers; any changes made to these registers during the interrupt service routine are overwritten. Because interrupt calls cannot occur immediately following a POPA instruction, the last instruction (RET) will execute before another interrupt call can occur. It is quite likely that the POPA instruction will re-enable a pending interrupt. If the Interrupt Controller serviced the pending interrupt before the RET instruction executed, 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 occur at a high frequency.

Notice that the "preamble" and exit code for this routine does not save or restore register RAM. The interrupt service routine is assumed to allocate its own private set of registers from the lower Register File. The availability of 232 bytes of RAM in the lower Register File makes this quite practical. In addition, the RAM in the upper Register File is available via *vertical windowing* (see Chapter 4).

5.3. INTERRUPT TIMING

Five external sources can interrupt the 8XC196KC/KD. The Transition Detector samples the interrupt inputs and latches the interrupt when a low-to-high transition occurs. The interrupt input must be held high for the minimum pulse width to ensure recognition. Some interrupts are sampled during Phase 1 (CLKOUT low); others are sampled during Phase 2 (CLKOUT high). Table 5-2 lists both the minimum pulse width and the sample clock phase for each external interrupt.

Table 5-2. 8XC196KC/KD Minimum Interrupt Pulse Width and Sample Clock Phase

Interrupt Source	Interrupt Vector(s)	Number	Minimum Pulse Width	Sampled During
HSI.0	HSI.0 Pin	INT04	> 2 state times	Phase 1
NMI	NMI	INT15	> 1 state time	Phase 2
P0.7	EXTINT	INT07	> 2 state times	Phase 1
P2.2	EXTINT	INT07	> 2 state times	Phase 2
	EXTINT1	INT13	> 2 state times	Phase 2
Timer 2 Capture	Timer 2 Capture	INT11	> 2 state times	Phase 2

Table 5-3 describes six instructions that always inhibit interrupts from being acknowledged until after the next instruction is executed.

Table 5-3. Instructions that Inhibit Interrupts

Instruction	Description
DI	Disables interrupts
EI	Enables interrupts following the execution of the next statement.
POPA	Pops two words off the top of the stack and places the first word into the INT_MASK1/WSR register-pair and the second word into the PSW/INT_MASK register pair.
POPF instruction	Pops one word off the top of the stack and places it into the PSW/INT_MASK register pair.
PUSHA	Pushes the PSW, INT_MASK, INT_MASK1, and the Window Select Register (WSR) onto the top of the stack, then clears the PSW, INT_MASK, and INT_MASK1 registers.
PUSHF	Pushes the PSW/INT_MASK register pair onto the top of the stack, then clears it.

Execution of any of the following also inhibits interrupts from being acknowledged until after the next instruction is executed:

- the signed prefix opcode (FE) for the two-byte, signed multiply and divide instructions
- the Unimplemented Opcode interrupt
- the Software Trap interrupt

5.3.1. Interrupt Latency

Latency is the total delay between the time that the interrupt is generated (not acknowledged) and the time that the 8XC196KC/KD begins executing the interrupt service routine or PTS cycle. A delay occurs between the time that the interrupt is detected and the time that it is acknowledged. An interrupt is not acknowledged until the current instruction finishes executing. If the interrupt does not occur at least four state times before the end of the current instruction, it may not be acknowledged until after the next instruction finishes. This additional delay occurs because instructions are prefetched and prepared a few state times before they are executed. Thus, the maximum delay between interrupt generation and acknowledgment is four state times plus the execution time of the next instruction.

When a standard interrupt is acknowledged, the hardware clears the interrupt pending bit and forces a call to the address contained in the corresponding interrupt vector after completing the current instruction. The procedure that gets the vector and forces the call requires 16 state times. If the stack is in external RAM, the call requires an additional 2 state times assuming a zero-wait-state bus. When a PTS interrupt is acknowledged, it immediately vectors to the PTSCB and begins executing the PTS cycle.

5.3.2. Calculating Latency

The maximum latency occurs when the interrupt occurs too late for acknowledgment following the current instruction. The following worst-case calculation assumes that the current instruction does not inhibit interrupts (see Table 5-3). To calculate latency, add the following terms:

- Four state times to allow the current instruction to finish
- The total number of state times for the next instruction; the longest instruction (NORML) takes 39 state times
- The response time (16 state times for an internal stack or 18 for an external stack) for standard interrupts only

5.3.2.1. STANDARD INTERRUPT LATENCY

The maximum delay for a standard interrupt is 61 state times (4 + 39 + 18). This delay time does not include time needed to execute the first instruction in the interrupt service routine. Figure 5-3 illustrates an example of this worst-case scenario.

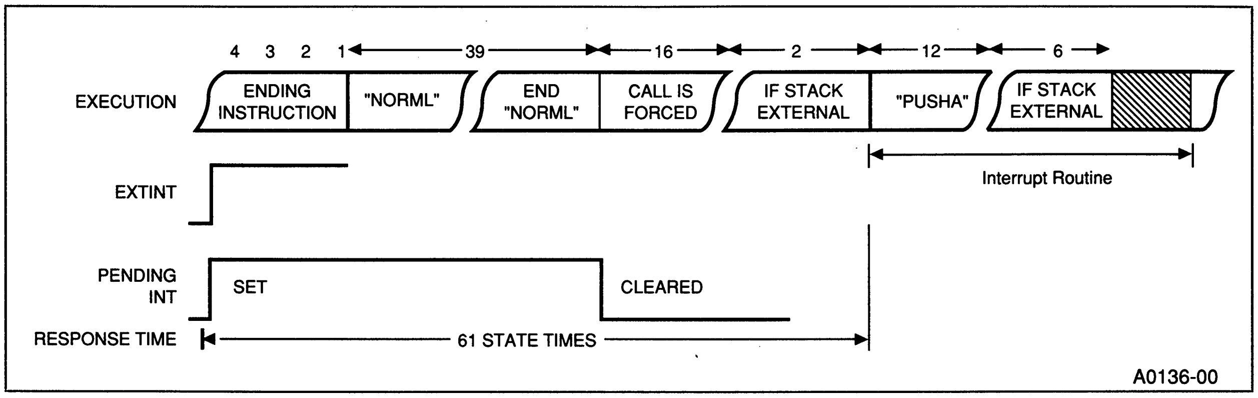


Figure 5-3. Standard Interrupt Response Time

5.3.2.2. PTS INTERRUPT LATENCY

The maximum delay for a PTS interrupt is 43 state times (4 + 39). This delay time does not include the added delay if the PTS is disabled (PSW.2 clear) or if a higher priority PTS request is being serviced. See Table A-11 in Appendix A for the PTS cycle execution times.

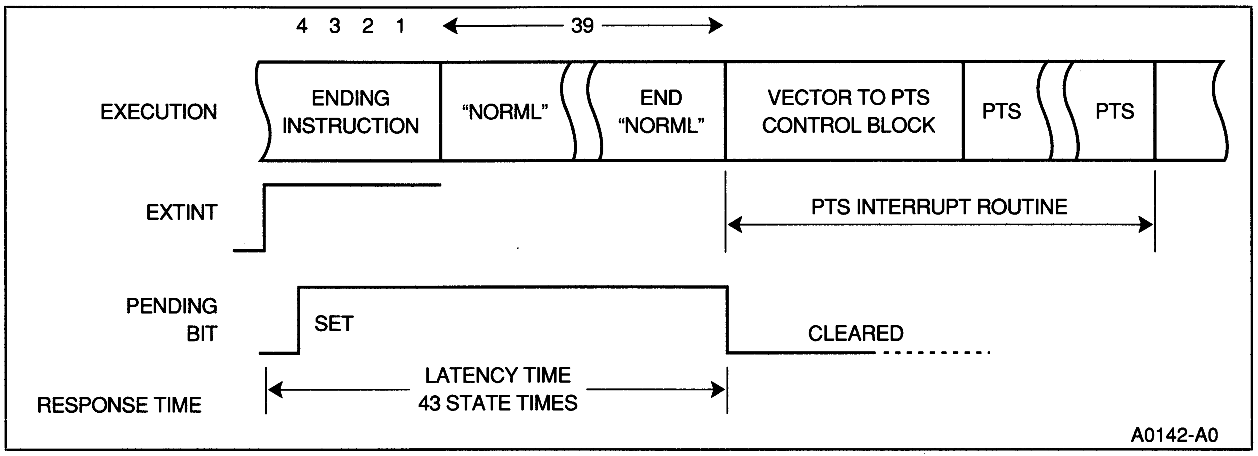


Figure 5-4. PTS Interrupt Response Time

5.4. SPECIAL INTERRUPTS

The 8XC196KC/KD supports three special interrupts: Unimplemented Opcode, Software Trap, and NMI. These interrupts are not affected by the interrupt enable bit (I) in the PSW (PSW.1), and they cannot be masked. All of these interrupts are serviced by the Interrupt Controller; they cannot be assigned to the PTS. Of these three, only NMI goes through the Transition Detector and Priority Encoder. The other two special interrupts go directly to the Interrupt Controller for servicing. Be aware that these interrupts are often assigned to special functions in Intel development tools.

5.4.1. Unimplemented Opcode

If the CPU attempts to execute an unimplemented opcode, an indirect vector through location 2012H occurs. This prevents random software execution during hardware and software failures. The interrupt vector should contain the starting address of an error routine that will not further corrupt an already erroneous situation. The Unimplemented Opcode interrupt prevents other interrupts from being acknowledged until after the next instruction is executed.

5.4.2. Software Trap

The TRAP instruction (opcode 0F7H) causes an interrupt call that is vectored through location 2010H. The TRAP instruction provides a single-instruction interrupt that is useful when debugging software or generating software interrupts. The TRAP instruction prevents other interrupts from being acknowledged until after the next instruction is executed.

5.4.3. NMI

The external NMI pin generates a Nonmaskable Interrupt for implementation of critical interrupt routines. NMI has the highest priority of all the prioritized interrupts. It is passed directly from the Transition Detector to the Priority Encoder, and it vectors indirectly through location 203EH. The NMI interrupt is sampled during Phase 2 (CLKOUT high) and is latched internally. If the pin is held high, multiple interrupts will not occur. If your system does not use the NMI interrupt, ground the NMI pin to prevent spurious interrupts.

For design symmetry with the INT_PEND1 register, an NMI mask bit exists in the INT_MASK1 register. However, the mask bit has no function; NMI is enabled for both NMI_MASK set and NMI_MASK cleared. To ensure compatibility with future products, always write zero to the NMI mask bit.

The NMI on the 8096BH vectors directly to location 0000H. For compatibility with 8096BH software that uses the NMI, load 0000H into location 203EH.

5.5. PROGRAMMING THE INTERRUPTS

Table 5-4 lists the programmable registers that affect the performance and function of the Interrupt Controller and PTS. Refer to Appendix C for detailed descriptions of these registers.

5.5.1. Selecting Either PTS or Standard Interrupt Service

The PTS Select register (PTSSEL) selects either a PTS cycle or a standard software interrupt service routine for each of the fifteen maskable interrupt requests. Setting a bit selects a PTS cycle; clearing a bit selects a standard interrupt service routine. See Appendix C for a detailed description of the PTSSEL register.

Table 5-4. Interrupt and PTS Control and Status Registers

Register Mnemonic	Register Name	Description
INT_MASK INT_MASK1	Interrupt Mask Registers	These registers enable/disable each maskable interrupt (that is, each interrupt except Unimplemented Opcode, Software Trap, and NMI.)
INT_PEND INT_PEND1	Interrupt Pending Registers	The bits in this register are set by hardware to indicate that an interrupt is pending.
IOC1	Input/Output Control Register 1	This register selects the source of the INT00, INT02, and INT07 interrupts.
IOS1	Input/Output Status Register 1	This register contains flags that indicate which events triggered interrupts.
PSW	Program Status Word	This register contains one bit that globally enables or disables servicing of all maskable interrupts and another that enables or disables the PTS.
PTSSEL	PTS Select Register	This register selects either a PTS cycle or a standard interrupt service routine for each of the fifteen maskable interrupt requests.
PTSSRV	PTS Service Register	The bits in this register are set by hardware to request an end-of-PTS interrupt.

5.5.2. Enabling PTS Interrupts

After you assign an interrupt to the PTS, you must enable both the PTS and the individual interrupt. The PTS enable (PSE) bit in the Program Status Word (PSW.2) globally enables or disables the PTS. The EPTS instruction sets the bit, which enables the PTS. The DPTS instruction clears the bit, which disables the PTS. The bits in INT_MASK and INT_MASK1 individually enable or disable the PTS interrupts (see Table 5-5).

5.5.3. Enabling Standard Interrupts

When you assign an interrupt to a standard software service routine, you must enable both the servicing of the interrupt and the individual interrupt. The global interrupt enable (I) bit in the Program Status Word (PSW.1) globally enables or disables the servicing of all maskable interrupts. The EI instruction sets the bit, which enables interrupt servicing. The DI instruction clears the bit, which disables interrupt servicing. The bits in INT_MASK and INT_MASK1 individually enable or disable the interrupts (see Table 5-5). Interrupts that occur while interrupt servicing is globally disabled (PSW.1 cleared) are held in the interrupt pending registers.

Table 5-5. Standard Interrupt Sources, Vectors, and Register Bits

Interrupt Source	Interrupt Vector(s)	Number	Interrupt Enabled by Setting (1)	Source Selected by (2)
A/D Conversion Complete	A/D Conversion Complete	INT01	INT_MASK.1	—
A/D Conversion Start	Software Timer	INT05	INT_MASK.5	—
HSI FIFO Fourth Entry	HSI FIFO 4	INT10	INT_MASK1.2	—
HSI FIFO Full	HSI FIFO Full	INT14	INT_MASK1.6	—
	HSI Data Available	INT02	INT_MASK.2	IOC1.7 = 1
HSI Holding Reg. Loaded	HSI Data Available	INT02	INT_MASK.2	IOC1.7 = 0
HSI.0	HSI.0 Pin	INT04	INT_MASK.4	—
HSO.0–HSO.5	High Speed Outputs	INT03	INT_MASK.3	—
NMI	NMI	INT15	—	—
P0.7	EXTINT	INT07	INT_MASK.7	IOC1.1 = 1
P2.2	EXTINT	INT07	INT_MASK.7	IOC1.1 = 0
	EXTINT1	INT13	INT_MASK1.5	—
RI Flag	Receive	INT09	INT_MASK1.1	—
	Serial Port	INT06	INT_MASK.6	—
Software Timers 0–3	Software Timer	INT05	INT_MASK.5	—
TI Flag	Transmit	INT08	INT_MASK1.0	—
	Serial Port	INT06	INT_MASK.6	—
Timer 1 Overflow	Timer Overflow	INT00	INT_MASK.0	IOC1.2 = 1
Timer 2 Capture	Timer 2 Capture	INT11	INT_MASK1.3	—
Timer 2 Overflow	Timer Overflow	INT00	INT_MASK.0	IOC1.3 = 1
	Timer 2 Overflow	INT12	INT_MASK1.4	—
Timer 2 Reset	Software Timer	INT05	INT_MASK.5	—

NOTES:

1. This column lists, for each interrupt source, the mask bit that must be set to enable the interrupt. Five interrupt sources can each generate two different interrupts — an 8096BH-compatible interrupt and a new, separate interrupt (HSI FIFO Full, EXTINT1, Receive, Transmit, Timer 2 Overflow). In all cases, only one interrupt should be enabled for each source. (That is, the mask bit should be set for only one of the two possible interrupts).
2. Three of the 8096BH-compatible interrupts (HSI Data Available, EXTINT, and Timer Overflow) can be generated by either of two sources. This column shows the IOC1 register bit and value that selects each source.

5.5.4. Selecting Interrupt Sources

Five interrupt sources can each generate two different interrupts — an 8096BH-compatible interrupt and a new, separate interrupt (HSI FIFO Full, EXTINT1, Receive, Transmit, Timer 2 Overflow). In all cases, only one interrupt should be enabled for each source. (That is, the mask bit should be set for only one of the two possible interrupts). Figure 5-5 shows the interrupt sources for each interrupt vector. Table 5-5 lists the IOC1 register bit and value that selects each source.

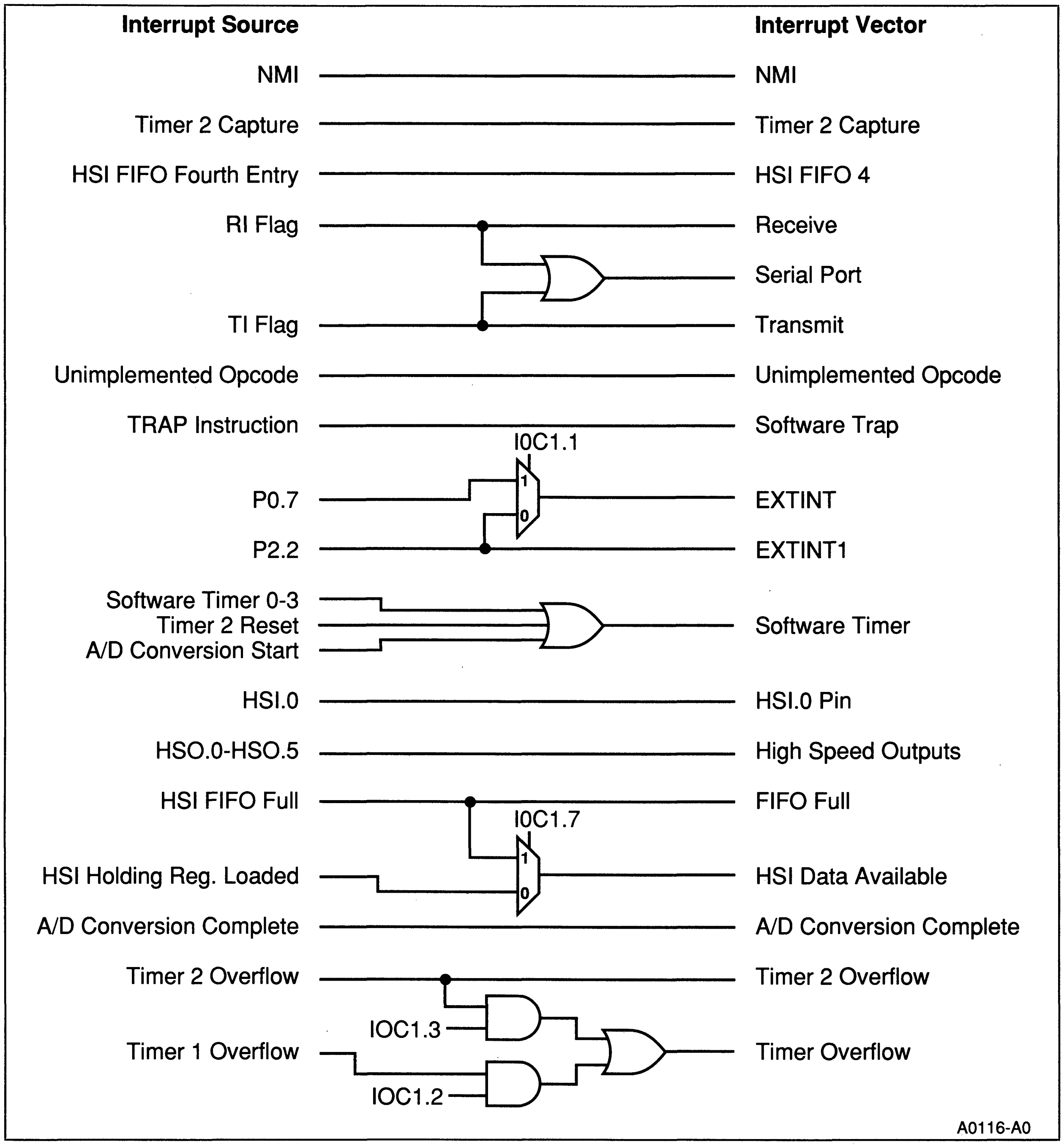


Figure 5-5. Interrupt Sources

5.5.5. Interrupt Mask Registers

The interrupt mask registers, INT_MASK and INT_MASK1, enable or disable (mask) individual interrupts. With the exception of the Nonmaskable Interrupt (NMI) bit (INT_MASK1.7), setting a bit enables the corresponding interrupt source; clearing a bit disables the source. When the device is reset, the interrupt mask registers are cleared (disabling interrupts).

For design symmetry with the INT_PEND1 register, an NMI mask bit exists in the INT_MASK1 register. However, the mask bit has no function; NMI is enabled for both NMI_MASK set and NMI_MASK cleared. To ensure compatibility with future products, always write zero to the NMI mask bit.

When the device is reset, the interrupt mask registers are cleared (disabling interrupts).

5.5.6. Interrupt Pending Registers

When the Transition Detector detects an interrupt, it sets the corresponding bit in the INT_PEND or INT_PEND1 register. This bit is set even if the individual interrupt is disabled (masked). The pending bit is cleared when the program vectors to the interrupt service routine (standard interrupts) or PTSCB (PTS interrupts). INT_PEND and INT_PEND1 can be read, to determine which interrupts are pending. They can also be modified (written), either to clear pending interrupts or to generate interrupts under software control.

Care should be taken in writing code that modifies these registers. For example, an instruction sequence that clears a pending bit could result in an interrupt being acknowledged after the sequence begins but before the bit is actually cleared. In this case a five-state-time *partial* interrupt cycle occurs. That is, the interrupt process begins, but never jumps to the interrupt service routine. This time delay can be avoided by making the code inseparable, in the sense that an interrupt will not be acknowledged while the code is executing. The easiest way to do this is to use the logical instructions in the two- or three-operand format, for example:

```
ANDB INT_PEND, #01111111B      ; Clears the EXTINT interrupt
ORB INT_PEND, #10000000B       ; Sets the EXTINT interrupt
```

The 8XC196KC/KD does not acknowledge interrupts during execution of these "read-modify-write" instructions.

The PTSSRV register holds requests for "end-of-PTS" interrupts. End-of-PTS interrupts indicate that the PTS needs servicing. The end-of-PTS interrupt service routine should reinitialize the PTSCB and set the appropriate PTSSEL bit to re-enable PTS interrupt service.

5.6. PTS CONTROL BLOCKS

Each PTS interrupt requires a block of data called the PTS Control Block (PTSCB). The PTSCB determines the PTS mode, the number of PTS cycles, and the address of the source and destination of data transfers. You must set up the PTSCB for each interrupt source before enabling the PTS interrupts. Each PTSCB requires eight data bytes in register RAM. The address of the first (lowest) byte is stored in the PTS Vector table in Special-Purpose memory (see Chapter 4, "Memory Partitions"). Write the first byte into an address evenly divisible by 8 (quad-word boundary). Figure 5-6 shows the PTSCB for each PTS mode. Unused PTSCB bytes can be used as extra RAM.

Single Transfer	Block Transfer	A/D Scan Mode	HSO Mode	HSI Mode
Unused	Unused	Unused	Unused	Unused
Unused	PTSBLOCK	Unused	PTSBLOCK	PTSBLOCK
PTSDST (HI)	PTSDST (HI)	PTSREG (HI)	Unused	Unused
PTSDST (LO)	PTSDST (LO)	PTSREG (LO)	Unused	Unused
PTSSRC (HI)	PTSSRC (HI)	PTS_S/D (HI)	PTSSRC (HI)	PTSDST (HI)
PTSSRC (LO)	PTSSRC (LO)	PTS_S/D (LO)	PTSSRC (LO)	PTSDST (LO)
PTSCON	PTSCON	PTSCON	PTSCON	PTSCON
PTSCOUNT	PTSCOUNT	PTSCOUNT	PTSCOUNT	PTSCOUNT

Figure 5-6. PTS Control Blocks

5.6.1. PTSCOUNT Register

The first location of each PTSCB is always the PTSCOUNT register. PTSCOUNT defines the number of PTS cycles to be executed consecutively without software intervention. Since PTSCOUNT is an 8-bit value, the maximum number of cycles is 256. PTSCOUNT is decremented at the end of each PTS cycle. When PTSCOUNT reaches zero, hardware clears the corresponding PTSSEL bit and sets the PTSSRV bit, which requests the end-of-PTS interrupt.

5.6.1.1. END-OF-PTS INTERRUPTS

An end-of-PTS interrupt is a standard interrupt. The Interrupt Controller processes it with an interrupt service routine that is stored in the memory location pointed to by the standard interrupt vector. For example, the PTS services the Transmit interrupt if PTSSEL.8 is set. The interrupt vectors through 2050H, but the corresponding end-of-PTS interrupt vectors through 2030H, the standard Transmit interrupt vector. When the end-of-PTS interrupt vectors to the interrupt service routine, hardware clears the PTSSRV bit. The interrupt service routine must set the PTSSEL bit to re-enable PTS service for the interrupt.

5.6.2. PTSCON Register

The second location of each PTSCB is always the PTSCON register. Three bits of the PTSCON register determine the PTS mode: Single Transfer, Block Transfer, A/D Scan, HSO, or HSI (see Table 5-6). The PTS mode defines the functions of the other bits; see Table 5-8 for Single and Block Transfer modes and Table 5-8. PTSCON has one configuration for the Single and Block Transfer modes (see Figure 5-7) and one for the A/D Scan, HSO, and HSI modes (see Figure 5-8). See Appendix C for a detailed description of the contents of PTSCON during each PTS mode. See Table A-11 in Appendix A for cycle execution time for each PTS mode.

Table 5-6. PTS Mode Select

Bit 7	Bit 6	Bit 5	Selected Mode
0	0	0	Single Transfer
0	0	1	HSI Mode
0	1	0	N/A
0	1	1	HSO Mode
1	0	0	Block Transfer
1	0	1	N/A
1	1	0	A/D Scan
1	1	1	N/A

Table 5-7. PTSCON Bits 0–4 (Single and Block Transfer Modes)

Bit Number	Bit Mnemonic	Bit Name	Description
0	DI	PTSDST Auto-Increment	Setting this bit causes the PTS destination register to increment at the end of each PTS cycle.
1	SI	PTSSRC Auto-Increment	Setting this bit causes the PTS source register to increment at the end of each PTS cycle.
2	DU	Update PTSDST	Setting this bit causes the PTSDST register to retain its final value at the end of a PTS cycle. Clearing it causes the register revert to the value that existed at the beginning of the PTS cycle.
3	SU	Update PTSSRC	Setting this bit causes the PTSSRC register to retain its final value at the end of a PTS cycle. Clearing it causes the register to revert to the value that existed at the beginning of the PTS cycle.
4	BW	Byte/Word Transfer	Setting this bit specifies a byte transfer. Clearing it specifies a word transfer.

Table 5-8. PTSCON Bit 3 (A/D Scan, HSI, and HSO Modes)

Bit Number	Bit Mnemonic	Bit Name	Description
3	UPDT	Update Register	Setting this bit causes the associated register(s) to be loaded with the value that exists at the end of a PTS cycle. Clearing it causes the register(s) to be loaded with the value that existed at the beginning of the PTS cycle. **Mode Register** A/D PTS_S/D HSI PTSDST HSO PTSSRC

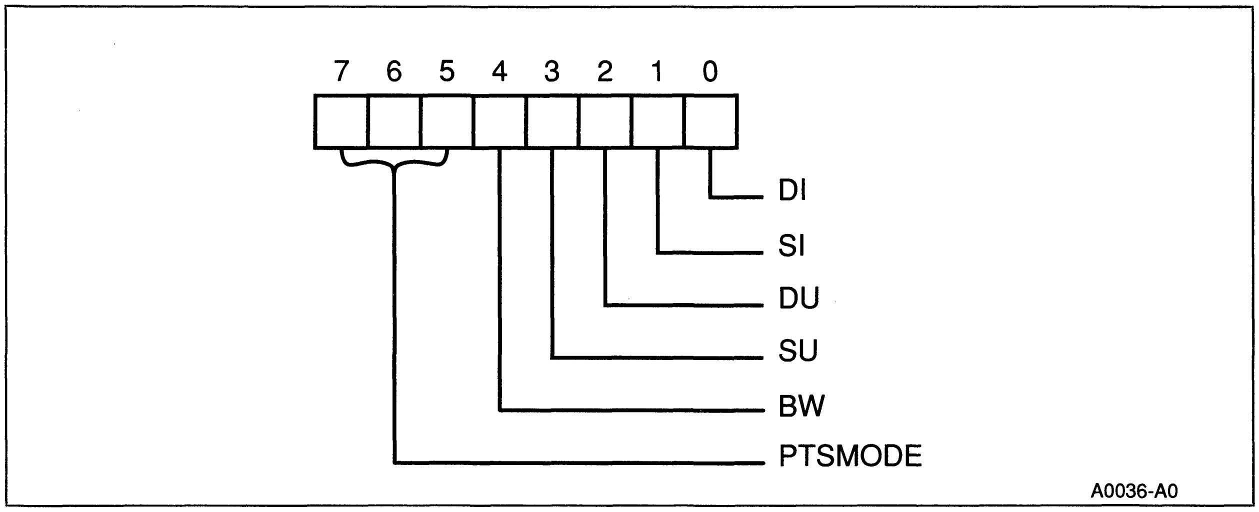


Figure 5-7. PTSCON (Single and Block Transfer Modes)

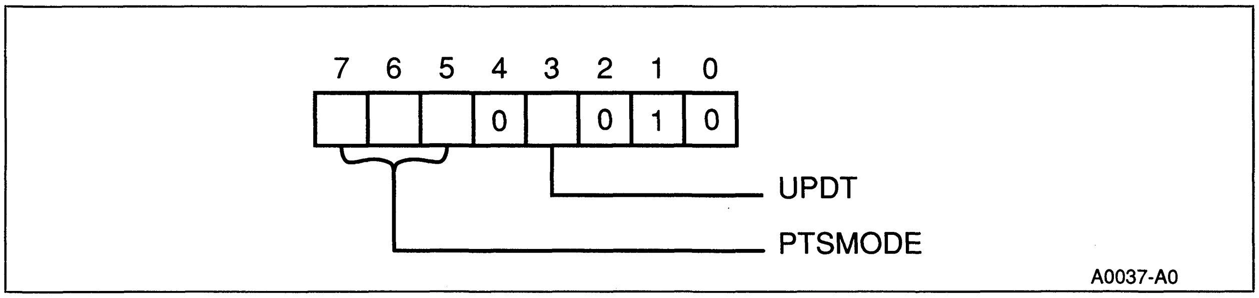


Figure 5-8. PTSCON (A/D Scan, HSI, and HSO Modes)

5.7. SINGLE TRANSFER MODE

In the Single Transfer mode, each PTS cycle transfers a single byte or word (selected by the BW bit in PTSCON) from one memory location to another. This mode is typically used with serial I/O port interrupts. The PTSCOUNT register specifies the number of transfers (each transfer is one PTS cycle). The PTS moves the byte or word from the location pointed to by the source register (PTSSRC) to the location pointed to by the destination register (PTSDST).

PTSSRC and PTSDST may point to any memory location; however, they must point to an even address if word transfers are selected. Setting the auto-increment and update bits causes the PTS to increment the source (if SI and SU are set) and/or destination (if DI and DU are set) address at the end of each PTS cycle. The address increments by one if byte transfers are selected or by two if word transfers are selected. In Single Transfer mode, each pair of auto-increment and update bits must both be either set or cleared. Programming the increment and update bits to (0,1) or (1,0) selects an invalid mode. PTSSRC and PTSDST can be incremented (and updated) independently of each other.

5.7.1. Single Transfer Mode Example

The following PTSCB defines nine PTS cycles (see Table 5-9). Each cycle moves a single word from location 20H to an external memory location. The PTS transfers the first word to location 6000H. Then it increments and updates the destination address and decrements the PTSCOUNT register; it does not increment the source address. When the second cycle begins, the PTS moves a second word from location 20H to location 6002H. When PTSCOUNT equals zero, the PTS will have filled locations 6000H–600FH, and an end-of-PTS interrupt is generated.

Table 5-9. Single Transfer Mode PTSCB

Unused
Unused
PTSDST (HI) = 60H
PTSDST (LO) = 00H
PTSSRC (HI) = 00H
PTSSRC (LO) = 20H
PTSCON = 15H (DI, DU, & BW = 1)
PTSCOUNT = 09H

5.8. BLOCK TRANSFER MODE

In Block Transfer mode, the PTS moves a block of data from one memory location to another. The PTSBLOCK register specifies the number of bytes or words in each block (*n* = 1–32). The PTS moves the block of bytes or words from the location pointed to by the source register (PTSSRC) to the location pointed to by the destination register (PTSDST).

PTSSRC and PTSDST may point to any memory location; however, they must point to an even address if word transfers are selected. Setting the auto-increment bits in the PTSCON register, causes the PTS to increment the source (SI set) and/or destination (DI set) address at the end of each PTS transfer. If the update bit is also set, the incremented address is saved in the PTSSRC (SU set) or PTSDST (DU set) register after each PTS cycle. Setting both the increment and update bits means that the source and/or destination address will be incremented after each cycle. The registers increment by one if byte transfers are selected or by two if word transfers are selected. The increment and update features may be selected independently (unlike in Single Transfer Mode).

In this mode, it is important to differentiate between a *PTS transfer* and a *PTS cycle*. A *PTS transfer* is the movement of a single byte or word from the source to the destination. A *PTS cycle* consists of the transfer of an entire block of bytes or words. Because a PTS cycle is uninterruptable, the Block Transfer mode can create long interrupt latency. The worst-case latency could be as high as 500 states. This worst-case latency assumes a block transfer of 32 words from one external memory location to another using an 8-bit bus with no wait states. See Table A-11 in Appendix A for PTS cycle execution times.

5.8.1. Block Transfer Mode Example

The PTSCB in Table 5-10 defines three PTS cycles that will each transfer the bytes in memory locations 20H–24H to one of the following blocks: 6000H–6004H, 6005H–6009H, or 600AH–600EH. Each PTS cycle requires a burst of five transfers. The source and destination are incremented after each transfer, but only the destination is updated after each cycle. The first byte of each cycle is always read from location 20H.

Table 5-10. Block Transfer Mode PTSCB

Unused
PTSBLOCK = 05H
PTSDST (HI) = 60H
PTSDST (LO) = 00H
PTSSRC (HI) = 00H
PTSSRC (LO) = 20H
PTSCON = 97H (DI, SI, DU, BW = 1)
PTSCOUNT = 03H

5.9. A/D SCAN MODE

In the A/D Scan mode, the PTS causes the A/D converter to perform multiple conversions on one or more channels and then stores the results. To use the A/D Scan mode, you must first set up a command/data table in memory (see Table 5-11). The command/data table contains A/D commands that are interleaved with blank memory locations. The PTS stores the conversion results in these blank locations.

To initiate A/D Scan mode, enable the A/D Conversion Complete interrupt and assign it to the PTS, then have software start the first conversion. When the A/D finishes the first conversion and generates an A/D Conversion Complete interrupt, the PTS cycle is initiated.

During each PTS cycle, the PTS stores the results from the previous conversion and then executes the next conversion command. Since the conversion results are not stored until the next PTS cycle, the last command location should contain all zeros to prevent a final conversion from starting. Typically, the A/D commands are loaded into the table from an external ROM. Only the amount of available memory limits the table size; it can reside in internal or external RAM.

Table 5-11. A/D Scan Mode Command/Data Table

XXX + 0AH	A/D Result 2	
XXX + 8H	Unused	A/D Command 3
XXX + 6H	A/D Result 1	
XXX + 4H	Unused	A/D Command 2
XXX + 2H	A/D Result 0*	
XXX	Unused	A/D Command 1

*Result of the A/D conversion that initiates the PTS cycle.

In A/D Scan mode, the PTSCOUNT specifies the total number of A/D conversion cycles. The PTS_S/D register points to the table of conversion commands and results. Setting the UPDT bit in the PTSCON register (PTSCON.3) causes the PTS_S/D register to retain its final value at the end of the PTS cycle. Clearing it causes the register to revert to the value that existed at the beginning of the PTS cycle. PTS_REG points to address 02H in HWindow 0. When read, this location contains the AD_RESULT register; when written, it contains the AD_COMMAND register. The A/D Scan mode also uses two temporary registers that are inaccessible to the user.

5.9.1. PTS Cycles in A/D Scan Mode

Software must start the first A/D conversion. The A/D Conversion Complete interrupt initiates the PTS cycle. The following actions occur after the PTS cycle begins:

1. The PTS reads the first command, stores it in a temporary location, and then increments the PTS_S/D register twice. PTS_S/D now points to the first blank location in the command/data table (address xxx + 2).

2. The PTS reads the AD_RESULT register, stores the results of the first conversion into location xxx + 2 in the command/data table, and increments the PTS_S/D register twice. PTS_S/D now points to xxx + 4.

3. The PTS loads the command from the temporary location into the AD_COMMAND register. This starts the next A/D conversion cycle.

4. If UPDT (PTSCON.3) is clear, the PTS_S/D register is reinitialized to its original value. The next cycle will use the same command and overwrite previous data. If UPDT is set, the PTS saves the new contents of PTS_S/D and it points to the next command.

5. PTSCOUNT is decremented and the CPU returns to regular program execution. When PTSCOUNT reaches zero, hardware clears the corresponding PTSSEL bit and sets the PTSSRV bit, which requests the end-of-PTS interrupt.

When the conversion started by the PTS cycle completes and the A/D generates the A/D Conversion Complete interrupt, a new PTS cycle begins. Steps 1–5 repeat.

Because the lower six bits of the AD_RESULT register contain status information, the end-of-PTS interrupt service routine could shift the results data to the right six times to leave only the conversion results in the memory locations.

5.9.2. A/D Scan Mode Example 1

The command/data table shown in Table 5-12 sets up a series of A/D conversions, beginning with channel 7 and ending with channel 0. Each table entry is a word (two bytes). Table 5-13 shows the corresponding PTSCB.

Software starts a conversion on Channel 7. Upon completion of the conversion, the A/D Conversion Complete interrupt initiates the first PTS cycle. Step 1 stores the Channel 6 command in a temporary location and increments PTS_S/D to 102H. Step 2 stores the result of the Channel 7 conversion in location 102H and increments PTS_S/D to 104H. Step 3 loads the Channel 6 command from the temporary location into the AD_COMMAND register to start the next conversion. Step 4 updates PTS_S/D (PTS_S/D points to 104H) and step 5 decrements PTSCOUNT to 7. The next cycle begins by storing the Channel 5 command in the temporary location. During the eighth cycle (PTSCOUNT = 1) the dummy command is loaded into the AD_COMMAND register and no conversion is performed. PTSCOUNT is decremented to zero and the end-of-PTS interrupt is requested.

Table 5-12. Command/Data Table (Example 1)

Address	Contents
11EH	AD_RESULT for ACH0
11CH	0000H (Dummy command)
11AH	AD_RESULT for ACH1
118H	AD_COMMAND for ACH0
116H	AD_RESULT for ACH2
114H	AD_COMMAND for ACH1
112H	AD_RESULT for ACH3
110H	AD_COMMAND for ACH2
10EH	AD_RESULT for ACH4
10CH	AD_COMMAND for ACH3
10AH	AD_RESULT for ACH5
108H	AD_COMMAND for ACH4
106H	AD_RESULT for ACH6
104H	AD_COMMAND for ACH5
102H	AD_RESULT for ACH7
100H	AD_COMMAND for ACH6

Table 5-13. A/D Scan Mode PTSCB (Example 1)

Unused
Unused
PTS_REG (HI) = 00H
PTS_REG (LO) = 02H
PTS_S/D (HI) = 01H
PTS_S/D (LO) = 00H
PTSCON = CAH (UPDT = 1)
PTSCOUNT = 08H

5.9.3. A/D Scan Mode Example 2

Table 5-14 sets up a series of ten PTS cycles that each read a single A/D channel and store the result in a single location (102H). UPDT is cleared so that original contents of PTS_S/D are restored after the cycle. The command/data table is shown in Table 5-15.

Table 5-14. A/D Scan Mode PTSCB (Example 2)

Unused
Unused
PTS_REG (HI) = 00H
PTS_REG (LO) = 02H
PTS_S/D (HI) = 01H
PTS_S/D (LO) = 00H
PTSCON = C2H (UPDT = 0)
PTSCOUNT = 0AH

Table 5-15. Command/DataTable (Example 2)

Address	Contents
102H	AD_RESULT for ACH*x*
100H	AD_COMMAND for ACH*x*

Software starts a conversion on Channel *x*. The first PTS cycle begins when the conversion is finished and the A/D Conversion Complete interrupt is generated. The PTS stores the conversion results in location 102H and then copies the conversion command from location 100H to the AD_COMMAND register. The CPU can process or move the conversion results data from the table before the next conversion completes and a new PTS cycle begins. When the next cycle begins, PTS_S/D again points to 100H. The conversion results are written to location 102H and the command at location 100H is re-executed.

5.10. HSI MODE

In HSI mode, the PTS dumps the contents of HSI FIFO to a table in either internal or external memory. The PTSDST register contains the address of the table.

Any HSI interrupt can be used to trigger the PTS cycle. The PTSBLOCK register specifies how many HSI FIFO blocks (*n* = 1–7) will be transferred to the memory table during each PTS cycle. Enter a value that corresponds with the interrupt that generates the PTS cycle. For example, the fourth FIFO entry can cause the HSI to generate the HSI FIFO 4 interrupt (INT10). If this interrupt was used to initiate the PTS cycle, then you would write 4 into the PTSBLOCK register so that the four FIFO entries would be dumped to the table.

FIFO data is accessed by reading the HSI_STATUS register first and then reading the HSI_TIME register. The HSI_STATUS register contains the event status bits and the current HSI pin states. The HSI_TIME register contains the time, with respect to the Timer 1 count value, at which the HSI event was triggered. See Chapter 8, "High-Speed Input/Output Unit," for a detailed description of the HSI module and Appendix C, "8XC196KC/KD Registers," for a detailed description of the HSI registers.

Each PTS transfer moves the HSI_STATUS and HSI_TIME registers into consecutive words in memory (see Table 5-16). Setting the UPDT bit (PTSCON.3) causes the PTSDST register to retain its final value at the end of the PTS cycle.

Table 5-16. HSI Mode PTS Table

XXX + 0AH	HSI_TIME_2	
XXX + 8H	0FFH	HSI_STATUS_2
XXX + 6H	HSI_TIME_1	
XXX + 4H	0FFH	HSI_STATUS_1
XXX + 2H	HSI_TIME_0	
XXX	0FFH	HSI_STATUS_0

5.10.1. HSI Mode Example

The PTSCB in Table 5-17 defines ten PTS cycles that will each transfer seven blocks of HSI_STATUS/HSI_TIME data from the HSI FIFO to a table starting at memory location 100H. The destination address is incremented after each transfer and updated with the new value after each cycle.

Table 5-17. HSI Mode PTSCB

Unused
PTSBLOCK = 07H
Unused
Unused
PTSDST (HI) = 01H
PTSDST (LO) = 00H
PTSCON = 2AH (UPDT = 1)
PTSCOUNT = 0AH

5.11. HSO MODE

In HSO mode, the PTS loads the HSO CAM file from a table located in either internal or external memory (see Table 5-18). The HSO mode is similar to the HSI mode, except that the PTS moves the data from the table to the HSO CAM file instead of vice versa. The PTSSRC register contains the address of the table.

Table 5-18. HSO Mode PTS Table

XXX + 0AH	HSO_TIME_2	
XXX + 8H	Unused	HSO_COMMAND_2
XXX + 6H	HSO_TIME_1	
XXX + 4H	Unused	HSO_COMMAND_1
XXX + 2H	HSO_TIME_0	
XXX	Unused	HSO_COMMAND_0

Each CAM register is 24 bits wide. Sixteen bits identify when, with respect to Timer 1 or Timer 2, the action should occur. The remaining 8 bits define the action. Load this information into the memory table in the format shown in Table 5-18. Each PTS cycle transfers data to the HSO_COMMAND and HSO_TIME registers. The data is transferred from the registers into the HSO holding register. The command is held in the holding register until there is an empty CAM register, at which time the command enters the CAM. See Chapter 8, "High-Speed Input/Output Unit" for more information about the CAM file.

Any HSO interrupt can be used to trigger the PTS cycle. The PTSBLOCK register specifies how many HSO entries (*n* = 1–8) will be transferred from the memory table during each PTS cycle. Setting the UPDT bit (PTSCON.3) causes the PTSSRC register to retain its final value at the end of the PTS cycle. See Chapter 8, “High-Speed Input/Output Unit,” for a detailed description of the HSO module and Appendix C, “8XC196KC/KD Registers,” for information about the HSO registers.

5.11.1. HSO Mode Example

The PTSCB in Table 5-19 defines ten PTS cycles that will each transfer eight blocks of HSO_COMMAND/HSO_TIME data to the HSO from the table starting at memory location 100H. The source address is incremented after each transfer and updated with the new value after each cycle.

Table 5-19. HSO Mode PTSCB

Unused
PTSBLOCK = 08H
Unused
Unused
PTSSRC (HI) = 01H
PTSSRC (LO) = 00H
PTSCON = 6AH (UPDT = 1)
PTSCOUNT = 0AH

I/O Ports 6

CHAPTER 6
I/O PORTS

I/O ports provide a mechanism to transfer information between the device and the surrounding system circuitry. They can read system status, monitor system operation, output device status, configure system options, generate control signals, provide serial communication, and so on. Their usefulness in an application is limited only by the number of I/O pins available and the imagination of the engineer.

6.1. FUNCTIONAL OVERVIEW

The 8XC196KC/KD has five 8-bit I/O ports. Each I/O port has pins that operate as either input-only (input), output-only (output), quasi-bidirectional (QBD), open-drain bidirectional, or a combination of the four. Almost all of the I/O port pins perform an alternate function. For example, one of the I/O port pins can become a PWM output, while another can be used as an analog input.

Every I/O port pin has a default operation (e.g., output). However, the operation of the pin may change when its alternate function is selected. One such change may be that a quasi-bidirectional pin becomes an output-only pin when the alternate function is selected. In any case, the pin has a basic set of operating characteristics associated with a given configuration (e.g., output). Table 6-1 lists the five I/O ports and their default and alternate functions.

The fifth column of Table 6-1 indicates which SFR determines whether the pin operates as a port or its alternate function. Some port pins operate both as a port and the alternate function at the same time (e.g., Port 0 and the analog inputs). In this case, the fifth column of Table 6-1 is left empty.

In addition to the standard I/O ports, the High-Speed Input module and the High-Speed Output module provide extra I/O functions if the timer-related features of these pins are not needed.

Table 6-1. Port Pin Functions

Port Pin(s)	Port Type	Alternate Function	Alternate Type	Control SFR
P0	Input	ACH0–ACH7	Input	
P1.0, P1.1, P1.2	Quasi-Bidirectional	None		
P1.3, P1.4	Quasi-Bidirectional	PWM1, PWM2	Output	IOC3
P1.5, P1.6	Quasi-Bidirectional	BREQ#, HLDA#	Output	WSR
P1.7	Quasi-Bidirectional	HOLD	Input	WSR
P2.0	Output	TXD	Output	IOC1
P2.1	Input	RXD	Input or Output	SPCON
P2.2	Input	EXTINT	Input	IOC1
P2.3	Input	T2CLK	Input	IOC0
P2.4	Input	T2RST	Input	IOC0
P2.5	Output	PWM0	Output	IOC1
P2.6	Quasi-Bidirectional	T2UP-DN	Input	IOC2
P2.7	Quasi-Bidirectional	T2CAPTURE	Quasi-Bidirectional	
P3, P4	Open-Drain Bidirectional	AD0–AD15	Bidirectional	

6.1.1. Input Port Pins

Any port pin defined as an input can only be read. Any write operation to the pin is ignored (or is not possible). The major circuits of an input pin consist of an input sample latch and a read buffer (see Figure 6-1).

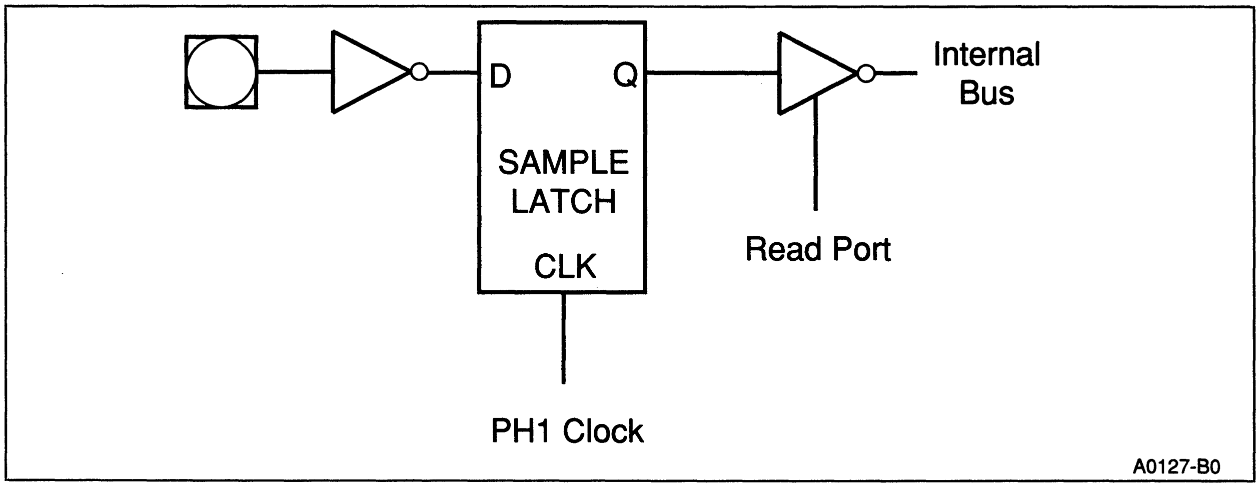


Figure 6-1. Input Port Pin

The port pin is sampled one state time before the read buffer is enabled. Sampling occurs during Phase 1 (while CLKOUT is low) and resolves the value (high or low) of the pin before it is presented to the internal bus. If the pin value changes during the sample time, the new value may or may not be recorded during the read.

Input port pins have no output drivers. The input leakage of these pins is very low, as is their capacitive loading.

6.1.2. Output Port Pins

An output pin consists of port latch and pin driver logic (see Figure 6-2).

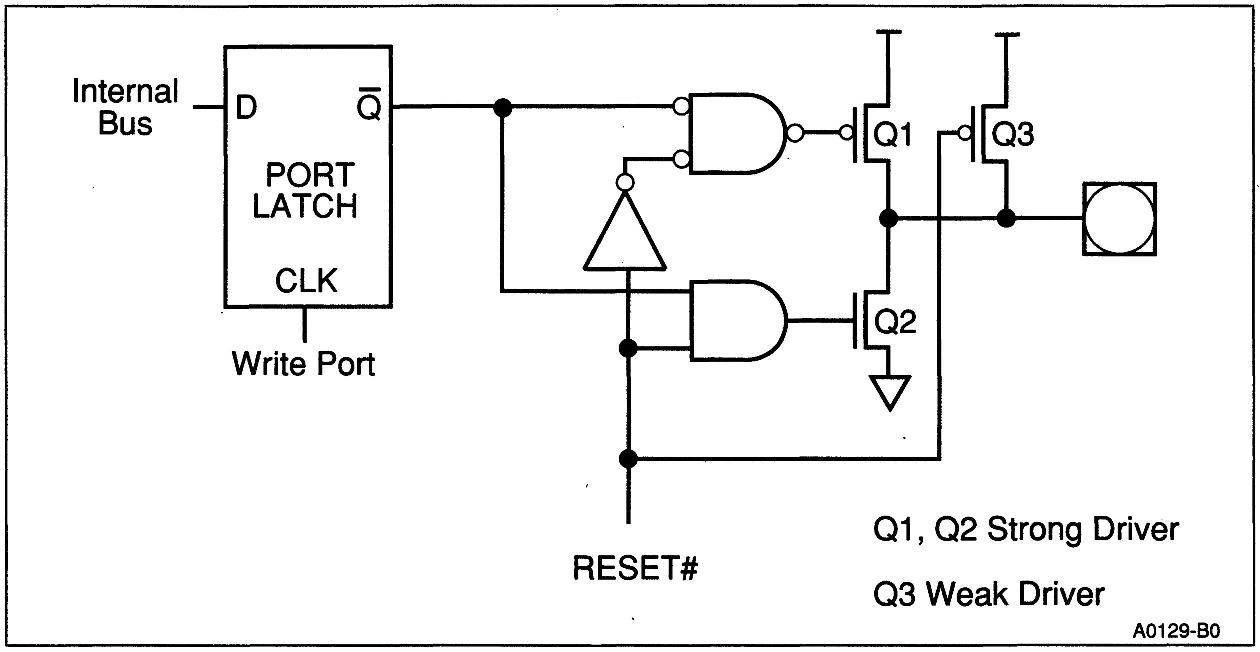


Figure 6-2. Output Port Pin

The new value of the output pin transitions during Phase 1 (while CLKOUT is low). The pin value remains stable until another write operation changes it (or a device reset occurs). Output port pins do not have sample and read buffer circuitry. When an output pin is read, its value is undefined and should be masked.

Since there is no mechanism to read the value of the port pin, it is possible only to set or clear an output pin. If it is necessary to complement the value of the pin, an image of the pin must be kept in memory. This memory image is manipulated and then used to set or clear the pin.

6.1.3. Quasi-Bidirectional Port Pins

Quasi-bidirectional pins can be used as input and/or output pins (without the need for direction control logic). These pins output a strong low value or a weak high value. The weak high value can be externally overridden, providing an input function.

Figure 6-3 illustrates the major circuits that make up a quasi-bidirectional port pin. Like an input pin, it has a sample latch and read buffer. Like an output pin, it has a data latch. It is the output driver structure that makes a quasi-bidirectional pin unique.

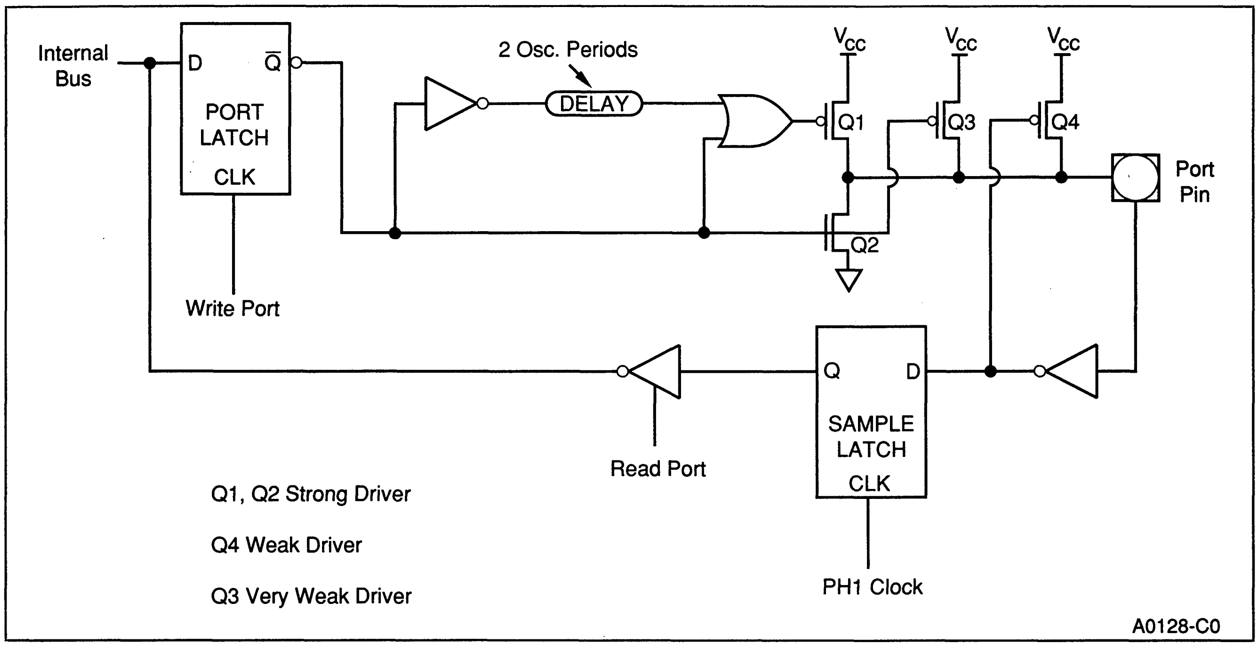


Figure 6-3. Quasi-Bidirectional Port Pins

The output of the pin can have only two static values: strongly driven low or weakly driven high. (If the output is transitioning from low to high, it is strongly driven high for a brief time; this is discussed later.) While the pin is driving low, it is not possible to overcome the driver (i.e., it cannot be driven high externally without exceeding device specifications).

Writing a one to the port pin disables the strong low driver (Q2) and enables a very weak high driver (Q3). To get the pin to transition high quickly, a strong high driver (Q1) is enabled for one state time and then disabled (leaving only Q3 active). Unless the port pin is heavily loaded, Q3 is capable of keeping the pin at a logic one state.

Because a quasi-bidirectional port pin has read circuitry, it is possible to read the output value of the pin directly (unlike the output port pins). However, since the port pin can be externally overridden with a zero, reading the port pin could falsely indicate that it was written as a zero.

The ability to overdrive the weak output driver is what gives the quasi-bidirectional port pin its input capability. To reduce the amount of current that flows when the pin is externally pulled low, the weak output driver (Q4) is turned off when a valid zero (low) is detected (approximately 2 volts).

Figure 6-4 illustrates the transfer characteristic for the port pin when Q3 and Q4 are enabled. As the pin voltage is lowered from V_{CC}, the current sourced by the pin increases until the threshold current, I_{TL}, is reached. At this point, Q4 is turned off and only a very weak pull-up (Q3) is present (I_{IL} in the data sheet).

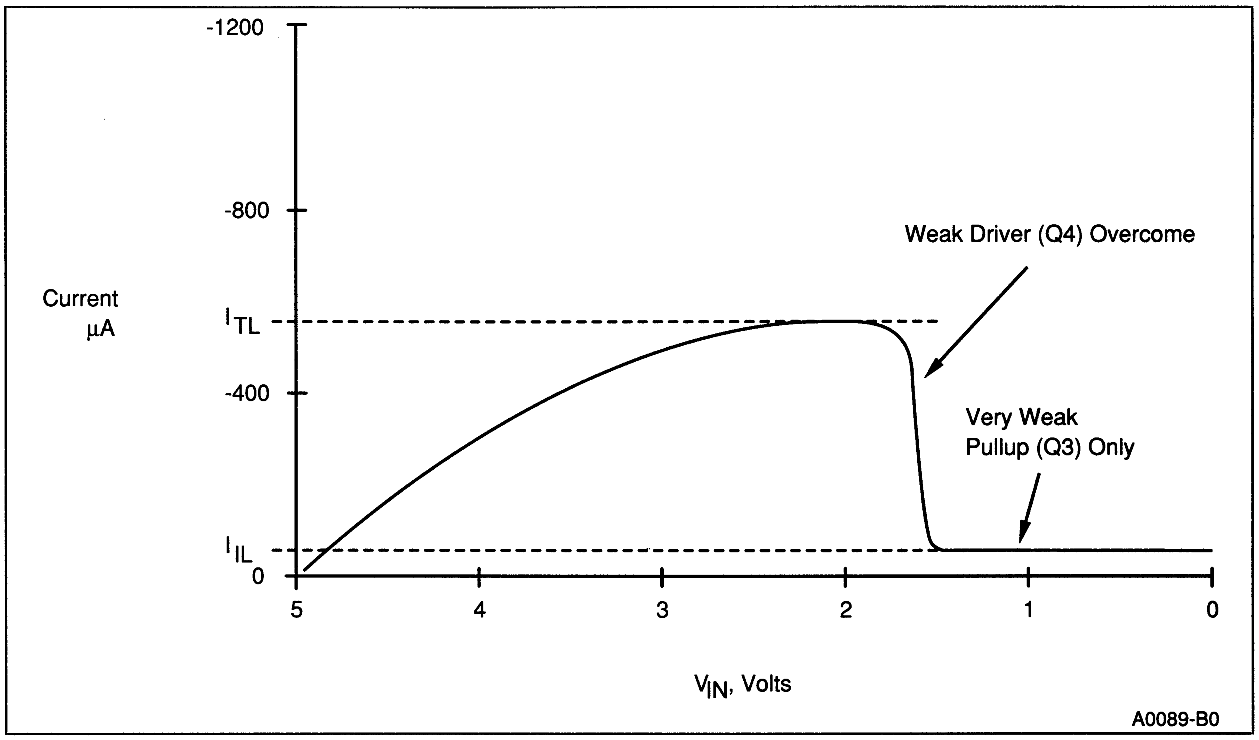


Figure 6-4. Quasi-Bidirectional Input Transfer Characteristic

The output of the quasi-bidirectional port pin also contains a pull-up driver (Q3) that is enabled whenever a one has been written to the output latch. This pull-up driver supplies a very small amount of current that causes a floating pin to return to a high (one) value. The current is specified as I_{IL} in the data sheet.

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. Specifically, care should be exercised when using XOR opcodes or any opcode that 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}.

The timing characteristics of the quasi-bidirectional port pins are similar to those of the input and output port pins. When the port pin is read, its value is sampled during Phase 1 (CLKOUT low) one state before the read buffers are enabled. A write to the output latch changes the pin value during Phase 1.

6.1.4. Open-Drain Bidirectional Port Pins

Open-drain bidirectional port pins operate like quasi-bidirectional port pins, except that there are no strong or weak high drivers. Figure 6-5 shows the basic circuits of the open-drain bidirectional port pin.

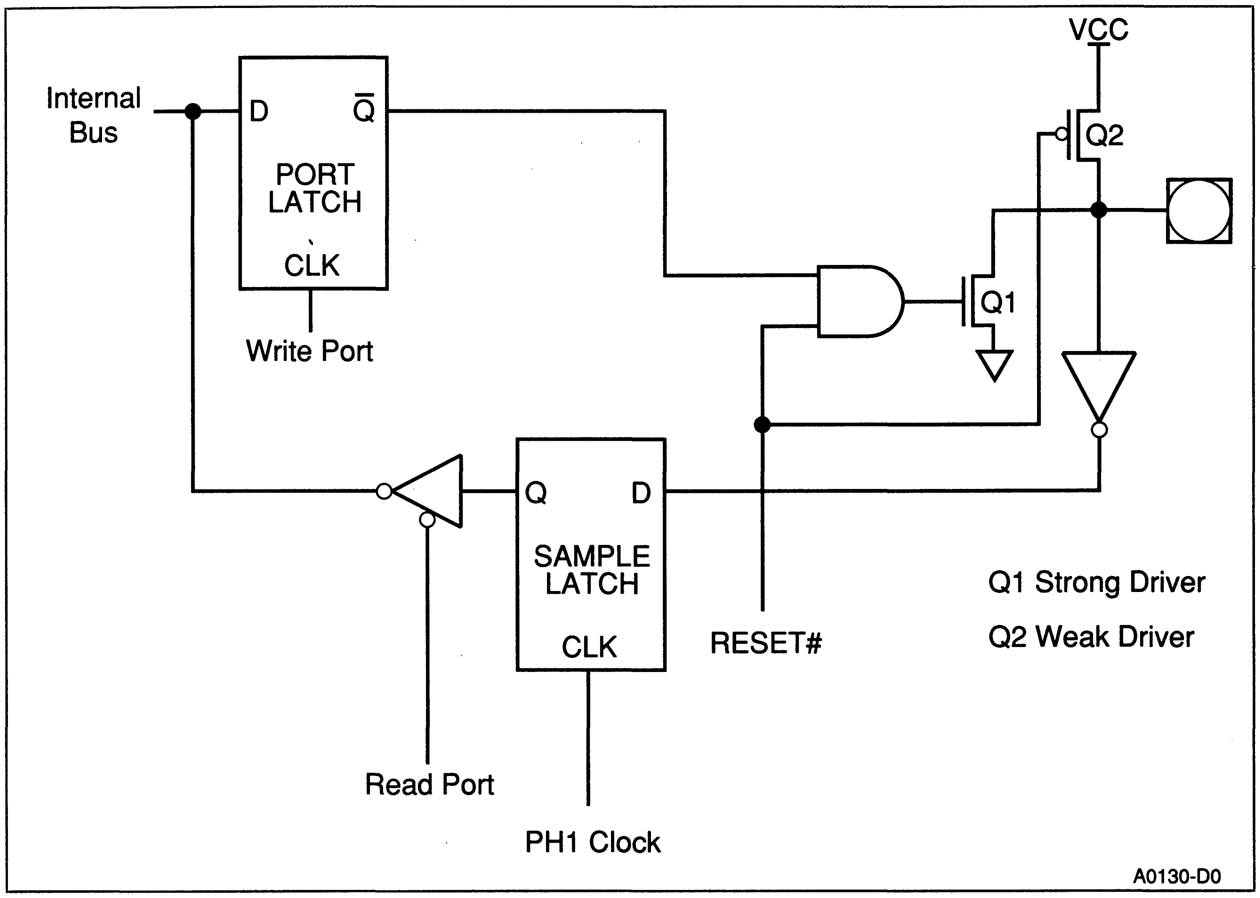


Figure 6-5. Open-Drain Bidirectional Port Pin

Writing a zero to the port latch enables the strong low driver (Q1). While Q1 is enabled, it is not possible to overdrive the zero externally without exceeding device specifications. A one written to the port latch disables Q1 and leaves the port pin floating. Externally, the port pin can be driven high with a pull-up resistor or logic gate.

6.2. PROGRAMMING THE I/O PORTS

Each of the five I/O ports has an associated Special Function Register (SFR) to read or write the port pins. Some of the I/O ports have an additional SFR that reconfigures the port pin to support an alternate function. Not all port SFRs are both readable and writeable (e.g., the Port 0 SFR is only readable). Also, not all SFRs exist in the same horizontal window (e.g., the SFR control registers are written in HWindow 0 but read back in HWindow 15).

IOPORT0, IOPORT1, and IOPORT2 are the SFRs used to read Port 0, read/write Port 1, and read/write Port 2, respectively. These registers have a similar format, shown in Figure 6-6. A complete description of each register can be found in Appendix C. Table 6-2 summarizes what happens when an I/O port is read or written and what conditions may exist to cause an unexpected result.

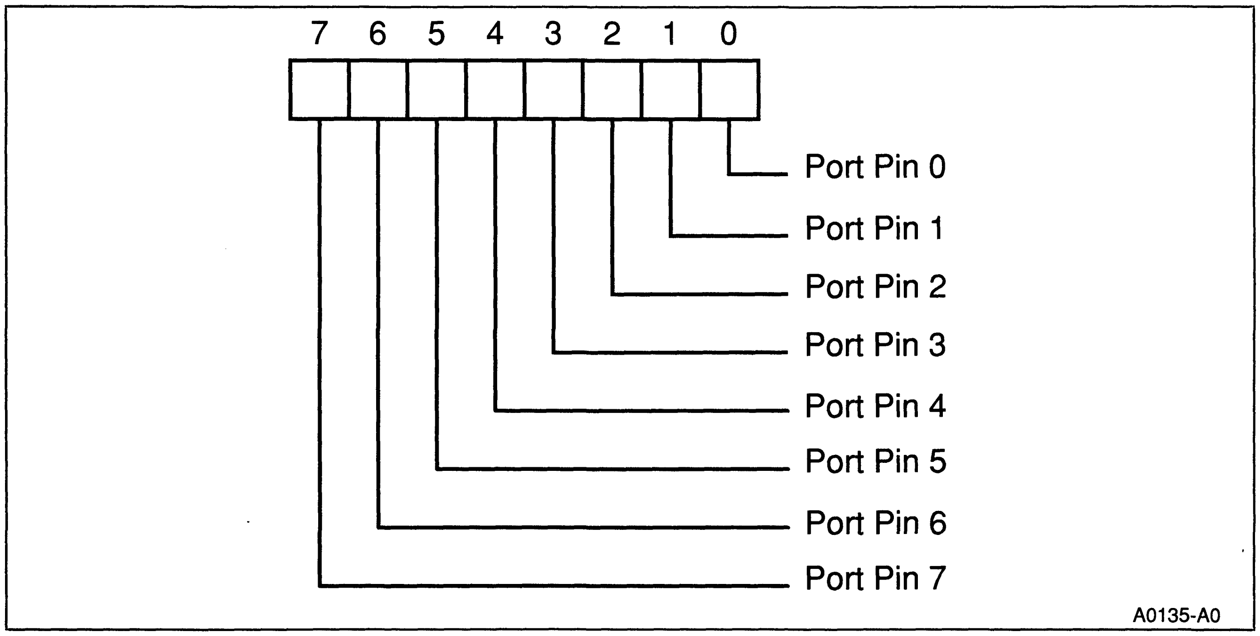


Figure 6-6. I/O Port SFR

Table 6-2. Summary of Port Read/Write Operation

Port	Accessed By	Read Operation	Write Operation
0	IOPORT0	Reads the current value of the port pins. The port should not be read while an A/D conversion is in progress.	Not possible. Writing to IOPORT0 does not affect Port 0, but does affect the baud rate controller.
1	IOPORT1	Reads the current value of the port pins. A port pin may have been written a value of one but return a value of zero because it is being overridden externally.	Writes change the state of the port pin latch. Changing the port pin latch has no effect on a port pin if its alternate function has been selected.
2	IOPORT2	Reads the current value of the port pins, except those pins that are output only. Output-only pins return an unspecified value and should be masked by software.	Writes change the state of the port pin latch, except those pins that are input only. Input-only port pins have no pin latches, so the value written is ignored.
3,4	1FFEH	Reads the current value of the port pins. A port pin may have been written a value of one but return a value of zero because it is being overridden externally. Ports 3 and 4 are always accessed together as a word read operation.	Changes the state of the port pin latch. Any external access of the bus controller overrides the value of the port pin latch to perform the bus operation. Ports 3 and 4 are always accessed together as a word write operation.

After a valid device reset, all of the I/O ports except Port 3 and Port 4 behave as port functions. When EA# is low during the rising edge of RESET#, Port 3 and Port 4 automatically support the system address/data bus and cannot be used as I/O ports unless they are reconstructed externally. All or portions of Port 0, Port 1, and Port 2 are optionally configured to enable alternate pin functions.

The SFR used to configure the ports depends on which I/O port pin is being reconfigured and is not discussed in this chapter. Instead, the chapter describing the operation of the peripheral that makes use of the I/O port pin contains the information needed to reconfigure the pin's function. Refer to Table 6-1 on page 6-2 to determine which SFR controls the alternate function of an I/O port pin (or pins).

6.2.1. Port Input

Reading an I/O port first samples the value of the port pin before it is stored internally. Because each port read samples a new value, it not possible to "latch" the port value simply by reading it once. If the port value must be operated on multiple times, a temporary memory location must be used to hold the initial value read. For example, the following operation reads the current value of Port 1 and stores the value in a memory location labeled P1TEMP:

```
LDB P1TEMP, ioport1          ;Read P1 and store in P1TEMP
```

Certain pins of Port 2 are configured as outputs only and do not contain any input circuitry. After the port is read, any SFR bit location containing an output-only pin should be masked before the input value is operated on.

To use a quasi-bidirectional port pin as an input, it is first necessary to ensure that the strong low driver is turned off. Writing a one to the port pin disables this strong driver. For example, the code below configures the lower half of Port 1 to be used as inputs:

```
LDB IOPORT1,#00001111B       ;Write 1s to Port 1 input pins
```

6.2.1.1. PORT 0 CONSIDERATIONS

Port 0 pins are special in that they may individually be used as digital inputs and analog inputs at the same time. A Port 0 pin being used as a digital input acts as a high-impedance input. However, Port 0 pins being used as analog inputs are required to provide current for a short time to charge the internal sample capacitor. This means that the input characteristics of a pin momentarily change if a conversion is being done on that pin. In either case, if Port 0 is to be used as analog or digital I/O, it is necessary to provide power to this port through the V_{REF} pin and the ANGND pin.

The A/D converter is sensitive to noise, and one way to induce noise into the converter is to read Port 0. It is strongly recommended that no port read operation occur while an A/D conversion is in progress. Port 0 has been designed such that the sample latch is not clocked unless a port read is being executed.

6.2.2. Port Output

Writing to an I/O port sets or clears its associated port latch. The output of the port latch is then used by the output driver. A write operation affects only output-only or bidirectional (quasi-bidirectional or open-drain) pins. A write to an input-only pin performs no operation (i.e., no latch is set or cleared).

There are many ways to write to an I/O port. The port can be written directly, using the following instructions:

```
LDB ioport1,#10000001B        ;Set bits 0 and 7, clear bits 1-6
                              ; or
LDB intrega,#81H              ;Value to be written
STB intrega,ioport1           ;Output value to port
```

Typically, a port is directly written to only for initialization or when all of the port pins need to change (or can change) at once. More often, only a single pin needs to be set, cleared, or toggled. To perform single-bit operations, a read-modify-write operation is performed on the port value, as follows:

```
LDB AL, ioport1               ;Read current port value
ORB AL, #10000001B            ;Set bits 0,7 without affecting bit 1-6
LDB ioport1, AL               ;Write the new port value
```

The above operation could all be done in one instruction:

```
ORB ioport1,#10000001B        ;reads ioport1, modifies it,
                              ;writes ioport1 back
```

The following operation could be used to complement a port pin value:

```
XORB ioport1,#1000000B        ;Complement P1.7
```

In all of the above operations, the present value of the port is read, modified, and then written back. However, there are some instances in which the above operations would yield less than desirable results. One example involves Port 2, which has two output-only pins. Since output-only pins have no read buffer logic, the port's current value cannot be read. Obviously, a read-modify-write operation does not make sense.

In another example, suppose Port 1 consisted of both inputs and outputs. Since all the pins of Port 1 are quasi-bidirectional, any of the pins used as inputs are required to be written as a one. This presents a problem when the port is read, modified, and then written back. If any of the input pins have a zero value being driven externally, performing a read-modify-write operation would write a zero back to the input port pin. Writing a zero to a quasi-bidirectional pin enables the strong low driver and causes a short-circuit condition when a one is driven externally.

To solve the example problems mentioned above, a mirror image of the port is maintained in memory. The operation (set, clear, toggle) to be performed on the port pins occurs on the memory image, which is then written to the port directly. To illustrate, the following operation is used to toggle P1.7 and assumes P1IMAGE can contain only a zero for those outputs driving a zero:.

```
XORB P1IMAGE,#10000000B       ;Complement P1.7 Image
LDB ioport1, P1IMAGE          ;Write new value to Port 1
```

There are other ways to get around the problem, such as making sure that the OR operation contains a one for any bidirectional pins used as inputs. However, it is important to remember that mixing inputs and outputs on the same port requires special attention when modifying an output value..

6.2.3. Port 3 and Port 4 Access

Accessing Port 3 and 4 as I/O is easily done from the internal Port 3 and Port 4 register (IOPORT34) at location 1FFEH. Note that Ports 3 and 4 can be accessed only as a word. It is not possible to read or write Port 3 alone or Port 4 alone. To use Ports 3 and 4 as I/O ports, the device must be set for internal execution (see "Port 3 and Port 4 Considerations" on page 6-11).

Since the LD and ST instructions require the use of internal registers, it may be necessary to first move the port information into the lower register file before using the data. If the data is already internal, the LD is unnecessary. For instance, to write a word value to Ports 3 and 4:

```
LD intreg,portdata            ;Register <- data
                              ;Not needed if already internal
ST intreg,1FFEH               ;Register -> Port 3 and 4
```

To input from Ports 3 and 4 requires that ones first be written to the port registers to set up the input port configuration circuit. This will put the ports in a high-impedance mode. Note that the ports are reset to this input condition. If zeroes have been written to the port, then ones must be rewritten to pins that are to be 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.

6.2.3.1. PORT 3 AND PORT 4 CONSIDERATIONS

The pins of Ports 3 and 4 have two I/O functions. They function as either an open-drain bidirectional port, a bidirectional system address/data bus (with strong low **and** high pin driver outputs), or both. How Ports 3 and 4 function depends on the state of the EA# pin during the last reset and on the need to access external program/data memory.

The EA# pin not only determines the operation of the Port 3 and 4 pins but also affects what happens when the ports are accessed. If EA# is low during a device reset, Ports 3 and 4 are used only to support the system address/data bus. A read or write to the Ports 3 and 4 address, 1FFEH, is interpreted as an external bus access, not a port access. If required, Ports 3 and 4 can be "reconstructed" by the circuit shown in Figure 6-7. This circuit uses standard latches and drivers to imitate the open-drain I/O function. A bus read/write access to address 1FFEH must be decoded to enable the latches properly.

When EA# is sampled high during a device reset, Ports 3 and 4 have two functions: I/O and system address/data bus. Normally the ports function as open-drain I/O pins. A zero written to the port latch at address 1FFEH enables a strong low driver, while a one written to the port latch disables the strong low driver and floats the pin. Writing a one to the port pins enables them to be used as inputs. The read or write to the port is interpreted as an internal access, so no external bus cycle is executed. Ports 3 and 4 remain open-drain I/O ports at all times unless an external bus cycle is executed.

Any access to an address not interpreted as an internal address results in the execution of an external bus cycle. To perform the bus cycle, Ports 3 and 4 temporarily switch function to support the system address/data bus. This means that the port pins have strong high and low drivers during address and data write portions of the bus cycle, and that they float during data read portions of the bus cycle. After the bus cycle completes, the port pins switch back to their I/O port function. The state of the port pin after the switch back to the I/O function depends on the value last written to the port latch.

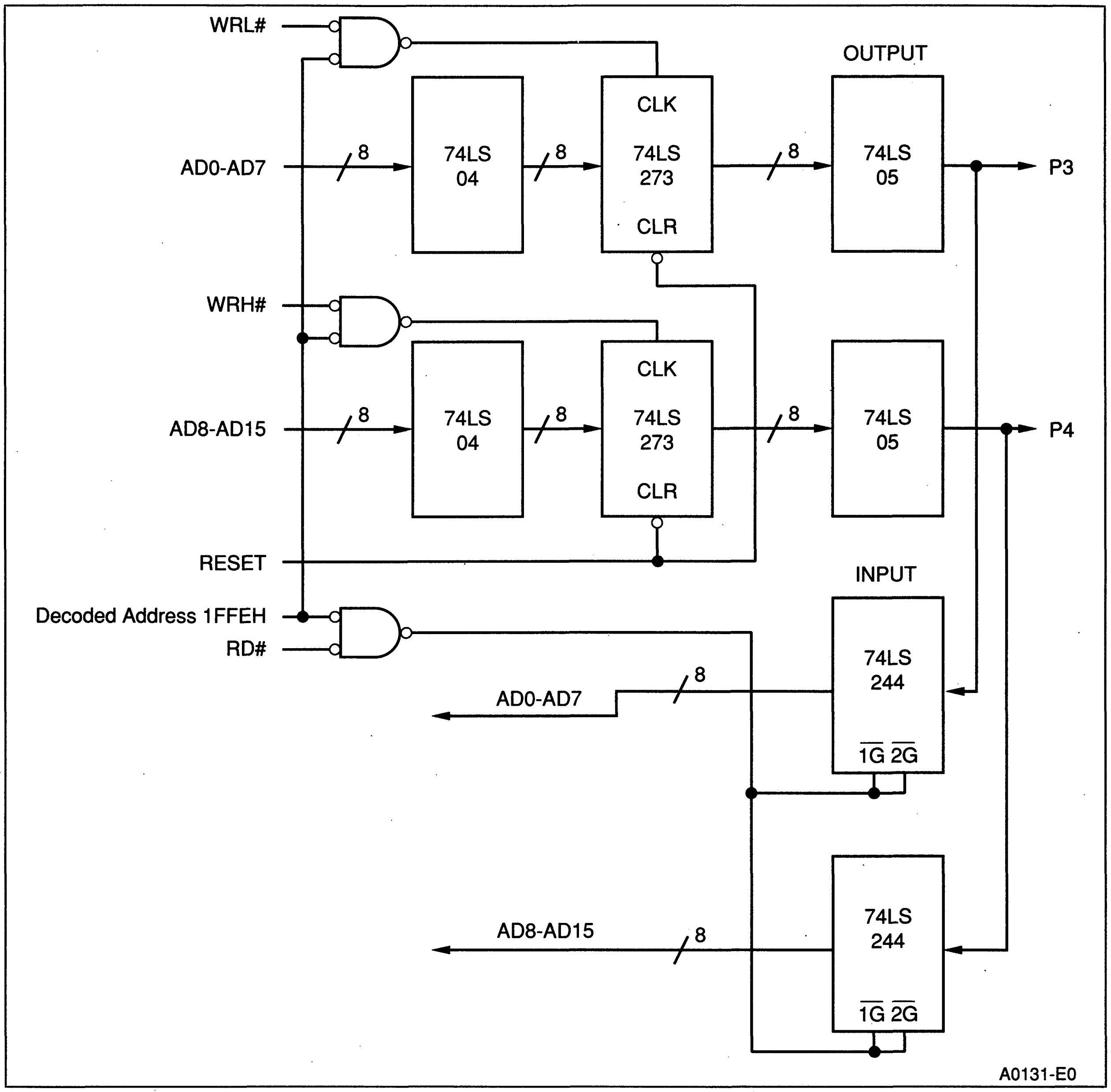


Figure 6-7. Ports 3 and 4 Reconstruction

In applications that use the system address/data bus, the value programmed into the port latch has some significance. The port latch value appears on the bus during idle times (i.e., during those times that Ports 3 and 4 are not the system address/data bus). Writing a zero to Ports 3 and 4 results in the system address/data bus being driven to a strong low when idle, while writing a one to Ports 3 and 4 causes the system address/data bus to float.

Which way the port latches should be written depends on the application. Writing zero means that the bus are driven low when idle, while writing one means the bus may float when idle. A floating bus could cause excessive current draw, but is easily fixed by placing pull-up resistors on the system address/data bus. Having the bus drive low removes the need to have resistors, but the design needs to ensure that no other device can drive the bus at the same time (possibly causing a short-circuit condition).

When Ports 3 and 4 are used exclusively as the system address/data bus, terminating the bus with pull-up resistors causes 0FFFFH to be read off the bus when an instruction fetch is made to an unimplemented memory location. Fetching a 0FFH operand causes the device to reset. Assuming that an instruction fetch to unimplemented memory is an indication of a system or software failure, having the device reset itself is good and prevents further device operation.

6.3. HARDWARE CONNECTION HINTS FOR QBD PORTS

When using the quasi-bidirectional ports as inputs tied to switches, series resistors may be needed if the ports are written to internally after the part is initialized. For example, writing a zero to a port pin that is externally tied to V_{CC} through a switch causes a short-circuit (high current) situation. The amount of current sourced to ground from each pin can exceed 20 mA. Sourcing this much current could make it possible to exceed the specification limits for the pin or port.

This potential problem can be solved in software or hardware. In software, never write a zero to a pin being used as an input. In hardware, a 1-KΩ resistor in series with each pin will limit current to a reasonable value without impeding the ability to override the high-impedance pull-up. The problem is less severe when the inputs are tied to electronic devices (e.g., TTL or CMOS gates) instead of switches, since the electronic devices tend not to sink excessive amounts of current.

Writing to a quasi-bidirectional port with electronic devices attached to the pins requires special attention. Consider trying to toggle P1.1 as an output:

```
        XORB ioport1,#00000010B        ;Complement P1.1
```

A problem can occur when the instruction executes. Even though P1.1 is driven high by the 8XC196KC/KD, it could possibly be held low externally. This typically happens when the port pin drives the base of an *npn* transistor, which in turn drives whatever needs to be toggled in the outside world. The base of the transistor will clamp the port pin to the transistor's V_{BE} above ground, typically 0.7V. The 8XC196KC/KD 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 problem can best be solved by the external driver design. Using a series resistor between the port pin and the base of the transistor often works, by raising the voltage present on the port pin. A software solution 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 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 pull-up 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.

Serial I/O Port

7

CHAPTER 7
SERIAL I/O PORT

The serial I/O port on the 8XC196KC/KD is an asynchronous/synchronous port that includes a Universal Asynchronous Receiver and Transmitter (UART). The UART has one synchronous mode (Mode 0) and three asynchronous modes (Modes 1, 2, and 3) for both transmission and reception. This chapter provides an overview of each mode and describes how to program the serial port.

To aid in understanding the discussions of each mode, the following paragraphs briefly describe the registers associated with the serial I/O port.

The SP_CON register selects the communications mode and enables or disables the receiver, even parity checking, and nine-bit data transmission. (See "Programming the Serial Port" on page 7-6.) The SP_STAT register contains the Receive Interrupt (RI) and Transmit Interrupt (TI) flags and three other status bits. (See "Serial Port Status" on page 7-10.) Note that reading the SP_STAT register clears all flags except TXE. For this reason, we recommend that you copy the contents of the SP_STAT register into a shadow register and then execute bit-test instructions such as JBC and JBS on the shadow register. Otherwise, if more than one bit-test instruction is executed, false values may be returned.

The serial port receives data into the SBUF (RX) register; it transmits data from the port through the SBUF (TX) register. The transmit and receive registers are accessed via separate *horizontal windows* (see Chapter 4, "Memory Partitions"), permitting simultaneous reads and writes to both. The transmitter and receiver are buffered to support continuous transmissions and to allow reception of a second byte before the first byte has been read.

The serial port interrupts are enabled and disabled through the Interrupt Mask register (INT_MASK or INT_MASK1); pending interrupts are indicated by the Interrupt Pending register (INT_PEND or INT_PEND1). (See "Enabling the Serial Port Interrupts" on page 7-8.)

An independent baud-rate generator controls the baud rate of the serial port. Either XTAL1 or T2CLK can provide the clock signal. The BAUD_RATE register selects the baud rate and the clock source. (See "Programming the Baud Rate and Clock Source" on page 7-8.)

Refer to Appendix C for details about the registers and to Appendix B for additional information about the signals discussed in this chapter.

7.1. MODE 0

The most common use of Mode 0, the synchronous mode, is to expand the I/O capability of the 8XC196KC/KD with shift registers (see Figure 7-1). In this mode, the TXD pin outputs a set of eight clock pulses, while the RXD pin either transmits or receives data. Data is transferred eight bits at a time with the least-significant bit first. Figure 7-2 shows a diagram of the relative timing of these signals. Note that only Mode 0 uses RXD as an open-drain output.

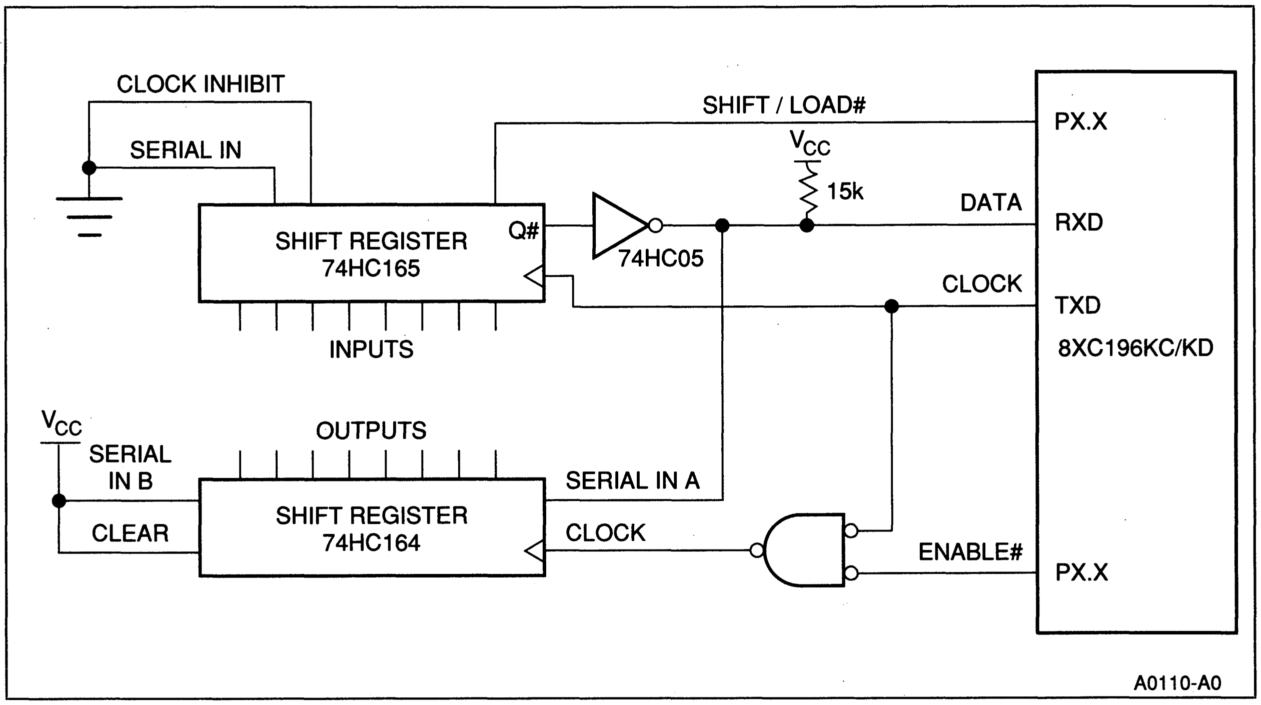


Figure 7-1. Typical Shift Register Circuit for Mode 0

In Mode 0, RXD must be disabled for transmissions to begin and enabled for receptions to begin. (See "Configuring TXD and RXD" on page 7-7.) When RXD is disabled, writing to SBUF (TX) starts a transmission. When RXD is enabled, either a rising edge on the RXD input or clearing the Receive Interrupt (RI) flag starts a reception.

Disabling RXD stops a reception in progress and inhibits further receptions. To avoid a partial or undesired complete reception, disable RXD before clearing the RI flag. 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.

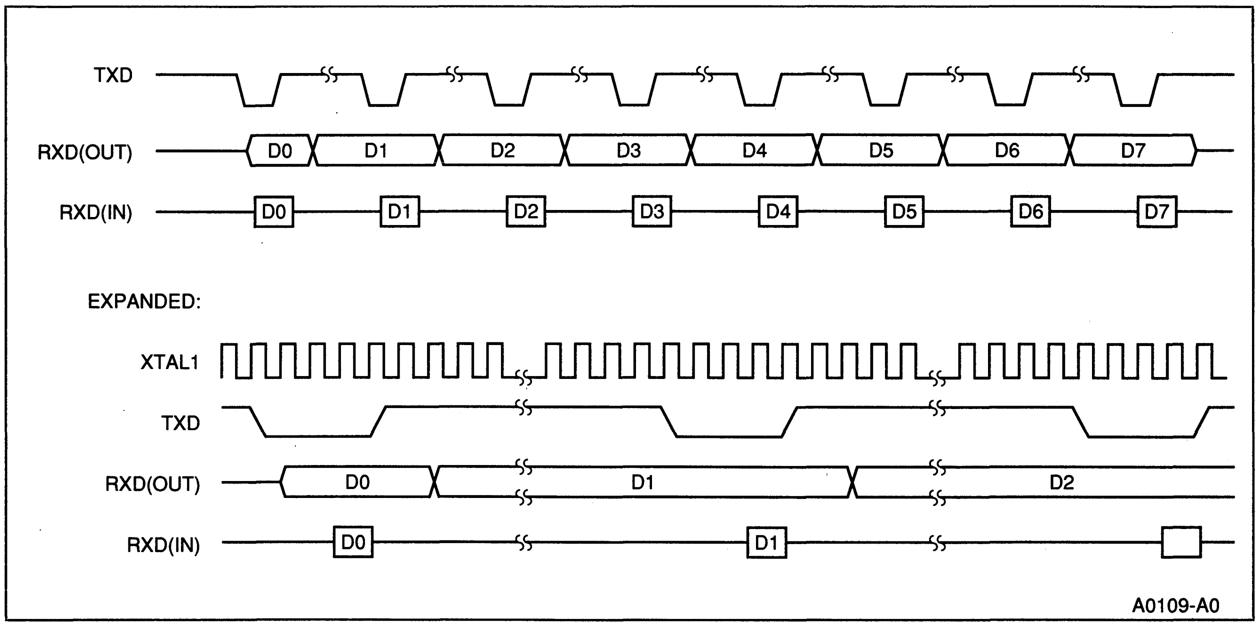


Figure 7-2. Mode 0 Timing

During a reception, the RI flag is set one bit time after the last data bit has been received. The receive interrupt signal is generated immediately before the RI flag is set. During a transmission, the TI flag is set immediately after the end of the last (eighth) data bit has been transmitted. The transmit interrupt signal is generated when the TI flag is set.

7.2. ASYNCHRONOUS MODES

Modes 1, 2, and 3 are full-duplex serial transmit/receive modes, meaning that they can transmit and receive data simultaneously. Mode 1 is the standard 8-bit, asynchronous mode used for normal serial communications. Modes 2 and 3 are 9-bit asynchronous modes typically used for interprocessor communications (see "Multiprocessor Communications" on page 7-5). In Mode 2, the serial port generates an interrupt only if the ninth data bit is set. In Mode 3, the serial port always generates an interrupt upon completion of a data transmission or reception.

When the serial port is configured for Mode 1, 2, or 3, writing to SBUF (TX) causes the serial port to start transmitting data. New data placed in SBUF (TX) is not transmitted until the stop bit of the previous data has been sent. A falling edge on the RXD input causes the serial port to begin receiving data if RXD is enabled. Disabling RXD stops a reception in progress and inhibits further receptions. (See "Configuring TXD and RXD" on page 7-7.)

7.2.1. Mode 1

Mode 1 is the standard asynchronous communications mode. The data frame used in this mode (see Figure 7-3) consists of ten bits: a start bit (0), eight data bits (LSB first), and a stop bit (1). If parity is enabled, an even parity bit is sent instead of the eighth data bit, and parity is checked on reception.

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. Therefore, the transmit clock may not be in sync with the receive clock, although both will be at the same frequency.

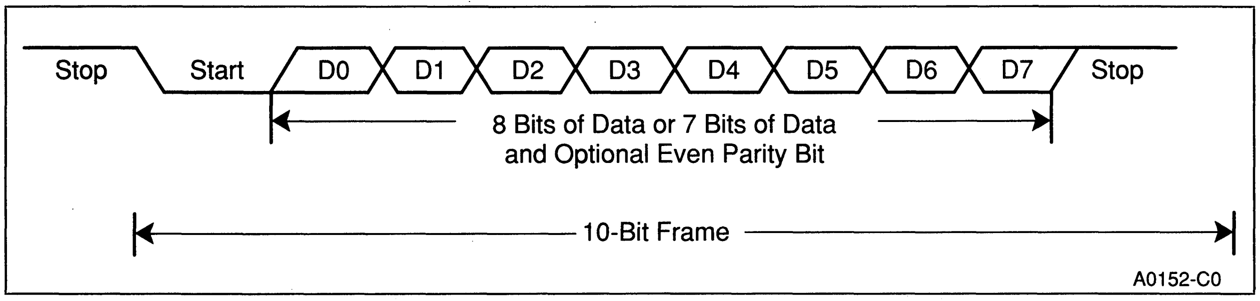

Stop
Start
D0
D1
D2
D3
D4
D5
D6
D7
Stop
8 Bits of Data or 7 Bits of Data and Optional Even Parity Bit
10-Bit Frame
A0152-C0

Figure 7-3. Serial Port Frames for Mode 1

The Transmit Interrupt (TI) and Receive Interrupt (RI) flags are set to indicate when operations are complete. During a reception, the RI flag is set just before the end of the stop bit. The receive interrupt signal is generated when the RI flag is set. During a transmission, the TI flag is set at the beginning of the stop bit. The transmit interrupt signal is generated when the TI flag is set. The next byte cannot be sent until the stop bit is sent.

Use caution when connecting more than two devices with the serial port in half-duplex (i.e., with one wire for transmit and receive). The receiving processor must wait for one bit time after the RI flag is set before starting to transmit. Otherwise, the transmission could corrupt the stop bit, causing a problem for other devices listening on the link.

7.2.2. Mode 2

Mode 2 is the asynchronous, ninth-bit recognition mode. This mode is commonly used with Mode 3 for multiprocessor communications. Figure 7-4 shows the data frame used in this mode. It consists of a start bit (0), nine data bits (LSB first), and a stop bit (1). During transmissions, setting the TB8 bit in the SP_CON register before writing to SBUF (TX) sets the ninth transmission bit. The hardware clears the TB8 bit after every transmission, so it must be set (if desired) prior to each write to SBUF (TX). During reception, the RI flag is set and an interrupt is generated **only** if the TB8 bit is set. This provides an easy way to have selective reception on a data link. (See "Multiprocessor Communications" on page 7-5.) Parity cannot be enabled in this mode.

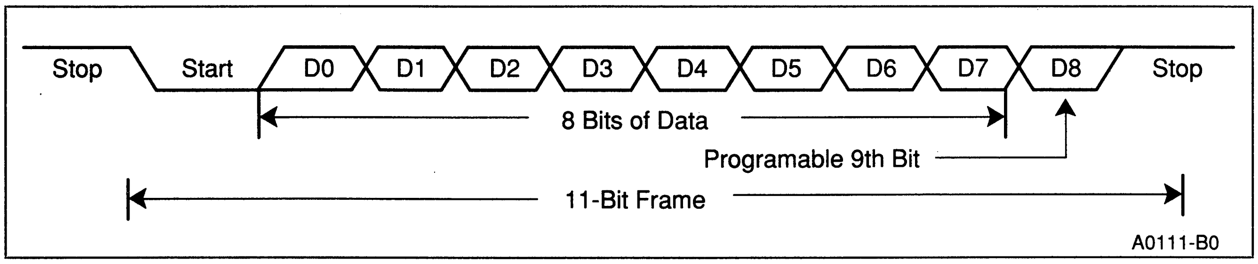


Figure 7-4. Serial Port Frames in Mode 2 and 3

7.2.3. Mode 3

Mode 3 is the asynchronous, ninth-bit mode. The data frame for this mode is identical to that of Mode 2. Mode 3 differs from Mode 2 during transmissions because parity can be enabled, in which case the ninth bit becomes the even parity bit. When parity is disabled, data bits 0–7 are written to the serial port transmit buffer, and the ninth data bit is written to bit 4 (TB8) bit in the SP_CON register. In Mode 3, a reception always causes an interrupt, regardless of the state of the ninth bit. If parity is disabled, the ninth data bit can be read in bit 7 (RB8) of the SP_STAT register. If parity is enabled, then RB8 becomes the Received Parity Error (RPE) flag.

7.2.4. Mode 2 and 3 Timings

Operation in Modes 2 and 3 is similar to Mode 1 operation. The only difference is that the data consists of 9 bits, so 11-bit packages are transmitted and received. During a reception, the RI flag is set just after the end of the stop bit. The receive interrupt signal is generated when the RI flag is set. During a transmission, the TI flag is set at the beginning of the stop bit. The transmit interrupt signal is generated when the TI flag is set. The ninth bit can be used for parity or multiprocessor communications.

7.2.5. Multiprocessor Communications

Modes 2 and 3 are provided for multiprocessor communications. In Mode 2, the serial port generates the Receive Interrupt (RI) only when the ninth data bit is set. In Mode 3, the serial port generates the Receive Interrupt (RI) regardless of the value of the ninth bit. The ninth bit is always set in address frames and always cleared in data frames. One way to use these modes for multiprocessor communication is to set the master processor to Mode 3 and the slave processors to Mode 2. When the master processor wants to transmit a block of data to one of several slaves, it sends out an address frame that identifies the target slave. An address frame will interrupt all slaves because the ninth bit is set. Each slave can examine the address byte and see if it is being addressed. The addressed slave switches to Mode 3 to receive the data frames, while the slaves that were not addressed remain in Mode 2 and are not interrupted.

7.3. PROGRAMMING THE SERIAL PORT

Table 7-1 lists the registers that affect the performance and function of the serial port.

Table 7-1. Serial Port Control and Status Registers

Register Mnemonic	Register Name	Description
BAUD_RATE	Baud Rate	This register selects the serial port baud rate and clock source. It must be written with two bytes, the least-significant byte first. The most-significant bit selects the clock source. The lower 15 bits represent the BAUD_VALUE, an unsigned integer that determines the baud rate.
IOC1	I/O Control Register 1	Setting bit 5 of this register enables the TXD function of P2.0.
SBUF (RX)	Serial Port Receive Buffer	This register contains data received from the serial port.
SBUF (TX)	Serial Port Transmit Buffer	This register contains data that is ready for transmission. In Modes 1, 2, and 3, writing to SBUF (TX) starts a transmission. In Mode 0, writing to SBUF (TX) starts a transmission only if the receiver is disabled (SP_CON.3=0)
SP_CON	Serial Port Control Register	This register selects the communications mode and enables or disables the receiver, even parity checking, and ninth-bit data transmissions. The TB8 bit is cleared after each transmission.
SP_STAT	Serial Port Status Register	This register contains the serial port status bits. It has a status bit for receive Overrun Errors (OE), Transmit buffer empty (TXE), Framing Errors (FE), Transmit Interrupts (TI), Receive Interrupts (RI), and Received Parity Error (RPE) or Received Bit 8 (RB8). Reading SP_STAT clears the OE, FE, TI, RI, and RPE/RB8 bits. Writing a byte to SBUF (TX) clears the TXE bit.

NOTE

Always write zeros to the reserved bits in these registers.

7.3.1. Selecting the Communications Mode and Enabling Parity

Bits 0 and 1 of the SP_CON register select the serial communications mode (see Figure 7-5 and Table 7-2). Selecting a new mode resets the serial I/O port and aborts any transmission or reception in progress on the channel. Setting bit 2 enables parity. Setting bit 3 enables the RXD function of pin P2.1. Bit 4 is the ninth data bit that will be transmitted in Mode 2 or 3 or, if parity is enabled, it is the even parity value. Note that bit 4 (TB8) is cleared after each transmission.

Table 7-2. Mode Selections

Bit 1	Bit 0	Selected Mode
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Mode 3

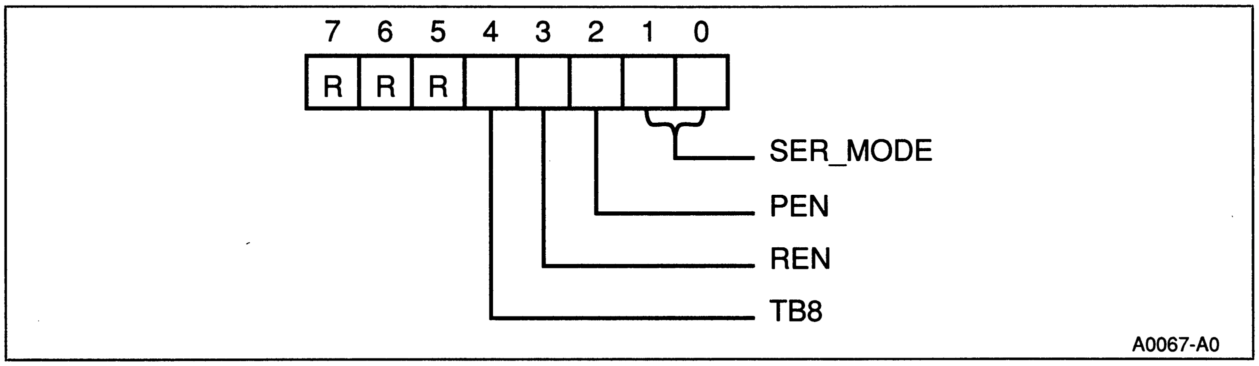


Figure 7-5. SP_CON Register

7.3.2. Configuring TXD and RXD

Table 7-3 lists the pins associated with the serial port. To enable TXD, set bit 5 of register IOC1; to disable TXD, clear the bit. To enable RXD, set bit 3 of register SP_CON; to disable RXD, clear the bit. (The "Programming the Baud Rate and Clock Source" section on page 7-9 discusses enabling T2CLK).

NOTE

The TXD pin controls entry into the ONCE mode when the 8XC196KC/KD is reset. In order to prevent inadvertently entering ONCE mode, external circuitry must not drive this pin low. See Chapter 12, "Special Operating Modes," for additional information on the ONCE mode.

Table 7-3. Serial Port Signals

Function Name	Additional Functions	Selected by	Type	Description
RXD	P2.1/PALE#	SP_CON.3=1	I/O	Receive Serial Data. In Modes 1, 2, and 3, RXD receives serial port input data. In Mode 0, it functions as an input or an open-drain output for data.
T2CLK	P2.3	BAUD_RATE MSB=0	I	Timer 2 clock input and one of two possible clock sources for the baud-rate generator input.
TXD	P2.0/PVER#	IOC1.5=1	O	Transmit Serial Data. In Modes 1, 2, and 3, TXD transmits serial port output data. In Mode 0, it is the serial clock output.

7.3.3. Enabling the Serial Port Interrupts

The serial port can be configured to generate either a single Serial Port interrupt or both a Transmit Interrupt (TI) and a Receive Interrupt (RI). If the Serial Port interrupt is masked and the Receive and Transmit interrupts are enabled, the RI flag and TI flag generate separate Receive and Transmit interrupts. If 8096BH compatibility is not an issue, this configuration is preferred.

If 8096H compatibility is desired, disable the Receive and Transmit interrupts and enable the Serial interrupt. Both RI flag and TI flag generate the Serial Port interrupt. When the TI flag generates the interrupt, bit 5 in the SP_STAT register is set. When the RI flag generates the interrupt, bit 6 in the SP_STAT register is set.

To enable the individual interrupts, set the TI_MASK and RI_MASK bits in the INT_MASK1 register. To enable the 8096BH-compatible interrupt, set the SER_MASK bit in the INT_MASK register. See Chapter 5, "Interrupts," for more information about interrupts.

Table 7-4. Serial Port Interrupt Registers

Register Mnemonic	Register Name	Hex Address	Description
INT_MASK	Interrupt Mask	08H	Setting bit 6 of this register enables the Serial Port interrupt (INT06, 200CH), which is the 8096BH-compatible configuration. Clearing bit 6 disables (masks) the interrupt.
INT_MASK1	Interrupt Mask 1	13H	Setting bit 0 of this register enables the Transmit interrupt (INT08, 2030H). Clearing bit 0 disables (masks) the interrupt. Setting bit 1 of this register enables the Receive interrupt (INT09, 2032H). Clearing bit 1 disables (masks) the interrupt.
INT_PEND	Interrupt Pending	09H	When set, bit 6 of this register indicates a pending Serial Port interrupt (INT06). It is cleared when the interrupt vectors to 200CH.
INT_PEND1	Interrupt Pending 1	12H	When set, bit 0 indicates a pending Transmit interrupt (INT08). It is cleared when the interrupt vectors to 2030H. When set, bit 1 indicates a pending Receive interrupt (INT09). It is cleared when the interrupt vectors to 2032H.

7.3.4. Programming the Baud Rate and Clock Source

The BAUD_RATE register selects the clock input into the baud-rate generator and defines the baud rate for all serial I/O modes. This word register must be loaded sequentially with two bytes. Always load the least-significant byte first.

The most-significant bit selects one of two possible clock sources for the baud-rate generator. To select the XTAL1 input signal (F_{OSC}) as the baud-rate clock source, set bit 15; to select an external signal on the T2CLK pin, clear bit 15. The maximum T2CLK input frequency is $F_{OSC}/4$.

The lower 15 bits represent BAUD_VALUE, an unsigned integer that determines the baud rate. The maximum value of BAUD_VALUE is 7FFFH (32,767 decimal). The minimum value in Modes 1, 2, and 3 is 0000H if XTAL1 is the baud-rate generator clock; otherwise, it is 0001H. The minimum value in Mode 0 is always 0001H.

Use the following equations to determine the BAUD_VALUE for a given baud rate.

Synchronous Mode 0: $$\text{BAUD\_VALUE} = \frac{F_{OSC}}{\text{Baud Rate} \times 2} - 1 \quad \text{or} \quad \frac{\text{T2CLK}}{\text{Baud Rate}}$$

Asynchronous Modes 1, 2, and 3: $$\text{BAUD\_VALUE} = \frac{F_{OSC}}{\text{Baud Rate} \times 16} - 1 \quad \text{or} \quad \frac{\text{T2CLK}}{\text{Baud Rate} \times 8}$$

Table 7-5 shows the BAUD_RATE values for common baud rates when using a 16 MHz XTAL1 clock input. Because of rounding, the BAUD_VALUE formula is not exact and the resulting baud rate is slightly different than desired. Table 7-5 shows the percentage of error when using the sample BAUD_RATE values. In most cases, a serial link will work with up to 5.0% difference in the receiving and transmitting baud rates.

Table 7-5. BAUD_RATE Values when Using XTAL1 at 16 MHz

	BAUD_RATE (Note 1)		% Error	
Baud Rate	**Mode 0**	**Mode 1, 2, 3**	**Mode 0**	**Mode 1, 2, 3**
9600	8340H	8067H	0.04	0.16
4800	8682H	80CFH	0.02	0.16
2400	8D04H	81A0H	0.01	0.08
1200	8A9AH	8340H	0	0.04
300	E82BH	8D04H	0	0.01

NOTES:

1. Bit 15 is always set when XTAL1 is selected as the clock source for the baud-rate generator.

7.4. SERIAL PORT STATUS

The status of the serial port is reflected in the SP_STAT register (see Figure 7-6). Note that reading the SP_STAT register clears all bits except TXE. For this reason, we recommend that you copy the contents of the SP_STAT register into a shadow register and then execute bit-test instructions such as JBC and JBS on the shadow register. Otherwise, the flags will be cleared when the bit-test instruction is executed.

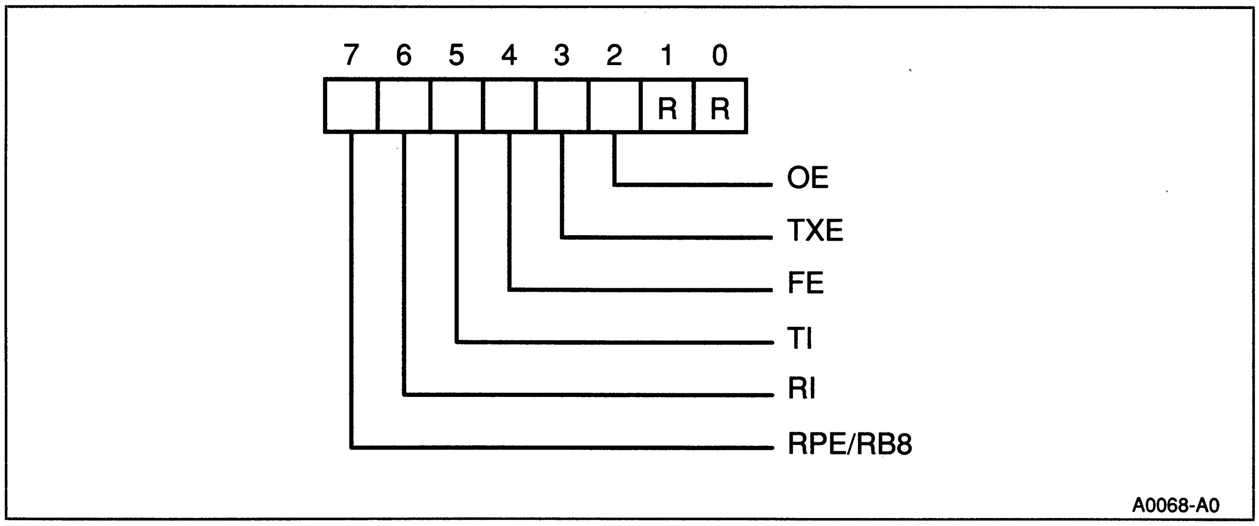


Figure 7-6. SP_STAT Register

The receiver on the 8XC196KC/KD checks for a valid stop bit. If a stop bit is not found within the appropriate time, the Framing Error (FE) bit in the SP_STAT register is set. When the stop bit is detected, the data in the receive shift register is loaded into SBUF (RX) and the Receive Interrupt (RI) flag is set. If this happens before the previous byte in SBUF (RX) is read, the Overrun Error (OE) bit is set. SBUF (RX) always contains the latest byte received; it is never a combination of the two bytes.

The Receive Interrupt (RI) flag indicates whether an incoming data byte has been received. The Transmit Interrupt (TI) flag indicates whether a data byte has finished transmitting. These flags also trigger interrupts and set the corresponding bits in the Interrupt Pending register. (See "Enabling the Serial Port Interrupts" on page 7-8.) Note that while a receive/transmit event sets the RI/TI bit in SP_STAT and the corresponding Interrupt Pending bit, a software write to RI/TI in SP_STAT does not affect the Interrupt Pending bits and does not cause an interrupt. Similarly, reading SP_STAT clears the RI and TI bits, but does not clear the corresponding bits in the Interrupt Pending register.

The Transmitter Empty (TXE) bit is set if SBUF (TX) and its buffer are empty and ready to take up to two bytes. TXE is cleared as soon as a byte is written to SBUF (TX). One byte may be written if TI alone is set. By definition, if TXE has just been set, a transmission has completed and TI will be set.

The Received Parity Error (RPE) flag or the Received Bit 8 (RB8) flag applies for parity enabled or disabled, respectively. If parity is enabled, RPE is set if a parity error is detected. If parity is not enabled, RB8 is the ninth data bit received in Modes 2 and 3.

High-Speed Input/Output Unit

8

CHAPTER 8
HIGH-SPEED INPUT/OUTPUT UNIT

The 8XC196KC/KD contains four peripherals that together form a flexible, timer/counter-based, high-speed input/output (HSIO) unit. The peripherals that make up the HSIO are Timer 1, Timer 2, the High-Speed Input (HSI) module, and the High-Speed Output (HSO) module (see Figure 8-1). The HSIO can measure pulse widths, generate waveforms, and create periodic interrupts with very little CPU overhead. This chapter describes each module within the HSIO unit.

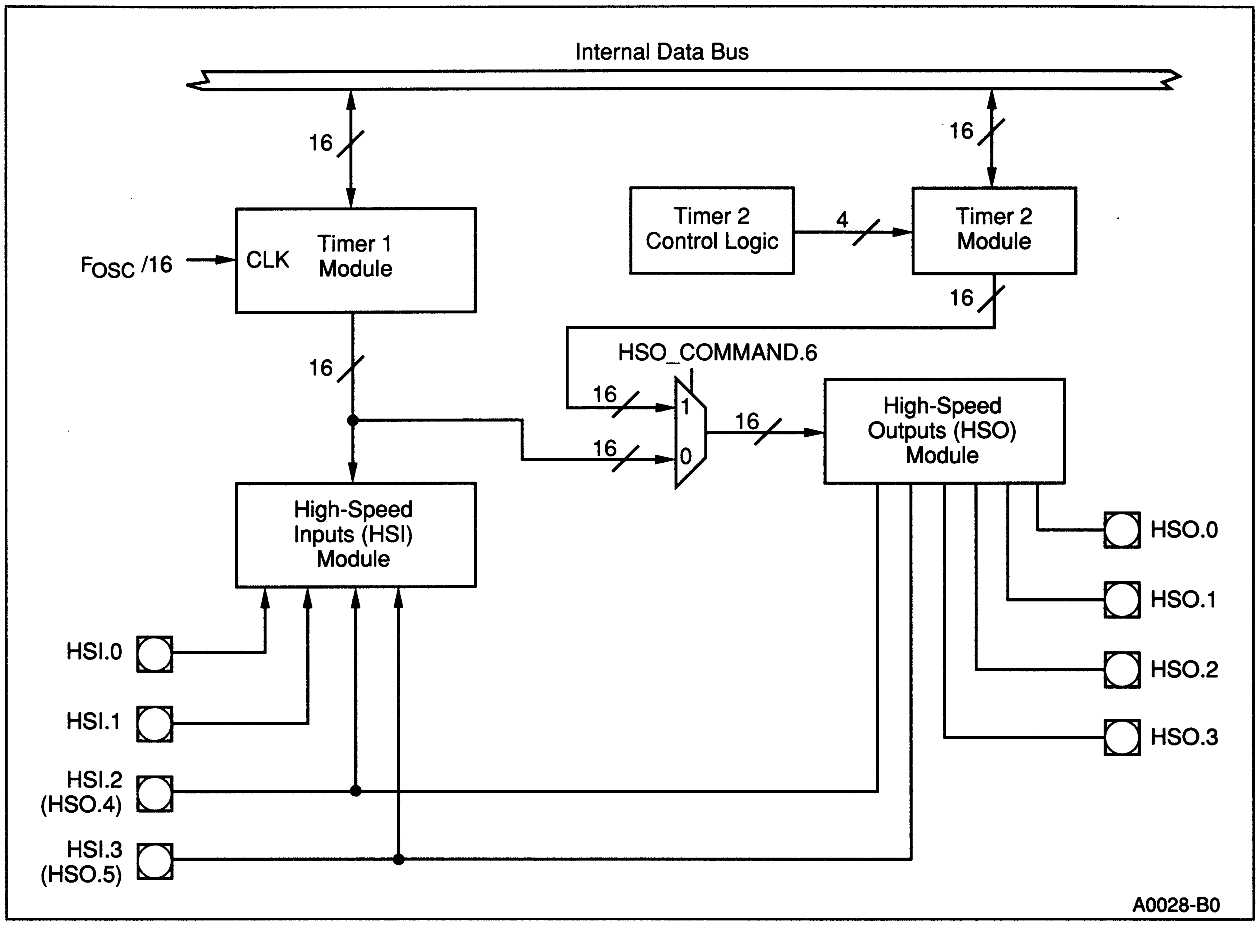


Figure 8-1. HSIO Block Diagram

8.1. TIMERS

The 8XC196KC/KD has two 16-bit timers, Timer 1 and Timer 2. The HSI module always uses Timer 1 as its time base. The HSO module uses either Timer 1 or Timer 2 as its time base.

8.1.1. Timer 1 Functional Overview

Timer 1 is a standard 16-bit, free-running timer that is incremented once every eight state times. Timer 1 is always the time base for the HSI module. It can also be selected as the time base for the HSO module. The two bytes of the TIMER1 register contain the Timer 1 count. You may initialize Timer 1 to a value other than zero by writing to the TIMER1 register in HWindow 15. If you change the TIMER1 value after initiating the HSI module, you may corrupt relative references between HSI events. Also, changing the TIMER1 value after initiating the HSO module with Timer 1 as the time base may cause skipped HSO commands.

8.1.2. Timer 2 Functional Overview

Timer 2 is a programmable 16-bit counter that can be used as the time base for the HSO module, as an up/down counter, or as an extra counter. It can also capture external events. Timer 2 can be clocked either internally or externally. When external clocking is selected, you may select either the T2CLK or HSI.1 pin as the clock source. The maximum count rate for Timer 2 is once every state time (fast increment mode) or every eight state times (normal mode).

The two bytes of the TIMER2 register contain the value of Timer 2. You may initialize Timer 2 to a value other than zero by writing to the TIMER2 register in HWindow 0. When using Timer 2 as the HSO time base, changing the TIMER2 value after initiating the HSO module may cause the HSO to skip commands. See "Timer Precautions" on page 8-9 for additional information.

8.1.3. Programming the Timer 2 Module

Table 8-1 lists the registers that affect the performance and function of the Timer 2 module. Appendix C, "8XC196KC/KD Registers," describes the registers in detail.

Table 8-1. Timer 2 Control and Status Registers

Register Mnemonic	Register Name	Description
TIMER2	Timer 2	Contains the value of Timer 2.
T2CAPTURE	Timer 2 Capture	A rising edge on P2.7 causes the value of Timer 2 to be captured into this register and generates a Timer 2 Capture interrupt (INT11).
INT_MASK INT_MASK1	Interrupt Mask Interrupt Mask 1	Enables or disables the Timer 2 interrupts.
IOC0	Input/Output Control Register 0	Selects the external clock and reset sources for Timer 2.
IOC1	Input/Output Control Register 1	Selects the interrupt source for the Timer Overflow interrupt (INT07).
IOC2	Input/Output Control Register 2	Enables or disables the fast increment mode and the up/down counter function, and selects the overflow boundary for the Timer 2 Overflow interrupt (INT12).
IOC3	Input/Output Control Register 3	Selects the internal or external clock source for Timer 2.

Table 8-2 lists the individual bits and port pins that control Timer 2 options. Column two lists the result of setting the bit or port pin. Column three lists the result of clearing the bit or port pin.

Table 8-2. Timer 2 Control Bits and Pins

Register Bits/ Port Pins	Bit = 1	Bit = 0
IOC0.1	Reset Timer 2 each write	No action
IOC0.3	Enables external reset	Disables external reset
IOC0.5	HSI.0 is the external reset source	T2RST (P2.4) is the external reset source
IOCO.7	HSI.1 is the external clock source	T2CLK (P2.3) is the external clock source
IOC1.3	Enable Timer 2 as source of Timer Overflow interrupt (INTOO)	Disables Timer 2 as source of Timer Overflow interrupt (INTOO)
IOC2.0	Enables fast increment mode	Disables fast increment mode
IOC2.1	Enables downcount feature	Count up only
IOC2.5	Interrupt on 7FFFH/8000H boundary	Interrupt on 0FFFFH/0000H boundary
T2UP-DN (P2.6)	Count down if IOC2.1 = 1	Count up if IOC2.1 = 1
T2CAPTURE (P2.7)	Captures TIMER2 into T2CAPTURE when a positive transition occurs and the logic level remains stable for at least one state time	—

8.1.3.1. SELECTING THE CLOCK SOURCE

Timer 2 counts both negative and positive transitions. It can be clocked either internally or externally, depending upon the state of the T2_ENA bit in the IOC3 register (IOC3.0). Setting IOC3.0 selects an internal clock source; clearing it selects an external clock source (see Figure 8-2).

When an external clock source is selected, the T2CLK_SRC bit in the IOC0 register (IOC0.7) controls which of two external sources is used. Setting IOC0.7 selects the HSI.1 pin as the external clock source; clearing it selects the T2CLK pin (P2.3) as the external clock source.

When Timer 2 is clocked externally, the FAST_T2_ENA bit in the IOC2 register (IOC2.0) controls the maximum input frequency on the external pin. When FAST_T2_ENA is set, the maximum input rate is one clock per state time (fast increment mode). When FAST_T2_ENA is cleared, the maximum input rate is one clock per eight state times (normal mode).

When Timer 2 is clocked internally, the FAST_T2_ENA bit selects the frequency of the clock input. When FAST_T2_ENA is set, the clock source equals $F_{OSC}/4$ and Timer 2 increments once every state time (fast increment mode). When FAST_T2_ENA is cleared, the clock source equals $F_{OSC}/32$ and Timer 2 increments once every eight state times.

If Timer 2 is the time base for the HSO module, use normal mode. The HSO requires eight state times to sample all CAM entries. HSO events could be missed if Timer 2 uses the fast increment mode. See "Timer Precautions" on page 8-9.

8.1.3.2. SETTING THE COUNT DIRECTION

The T2UD_ENA bit in the IOC2 register (IOC2.1) controls whether Timer 2 counts up only or counts either up or down depending upon the value of the T2UP-DN pin (see Figure 8-2). If IOC2.1 is cleared, Timer 2 always counts up. If IOC2.1 is set and T2UP-DN is low, Timer 2 counts up. If IOC2.1 is set and T2UP-DN is high, Timer 2 counts down.

The T2UP-DN signal must be stable either before or at the same time that the T2CLK signal changes, to ensure that the timer will change direction during the next clock period.

When Timer 2 is the time base for the HSO module, do not change the count direction or events may occur in the wrong order. See "Timer Precautions" on page 8-9 for details.

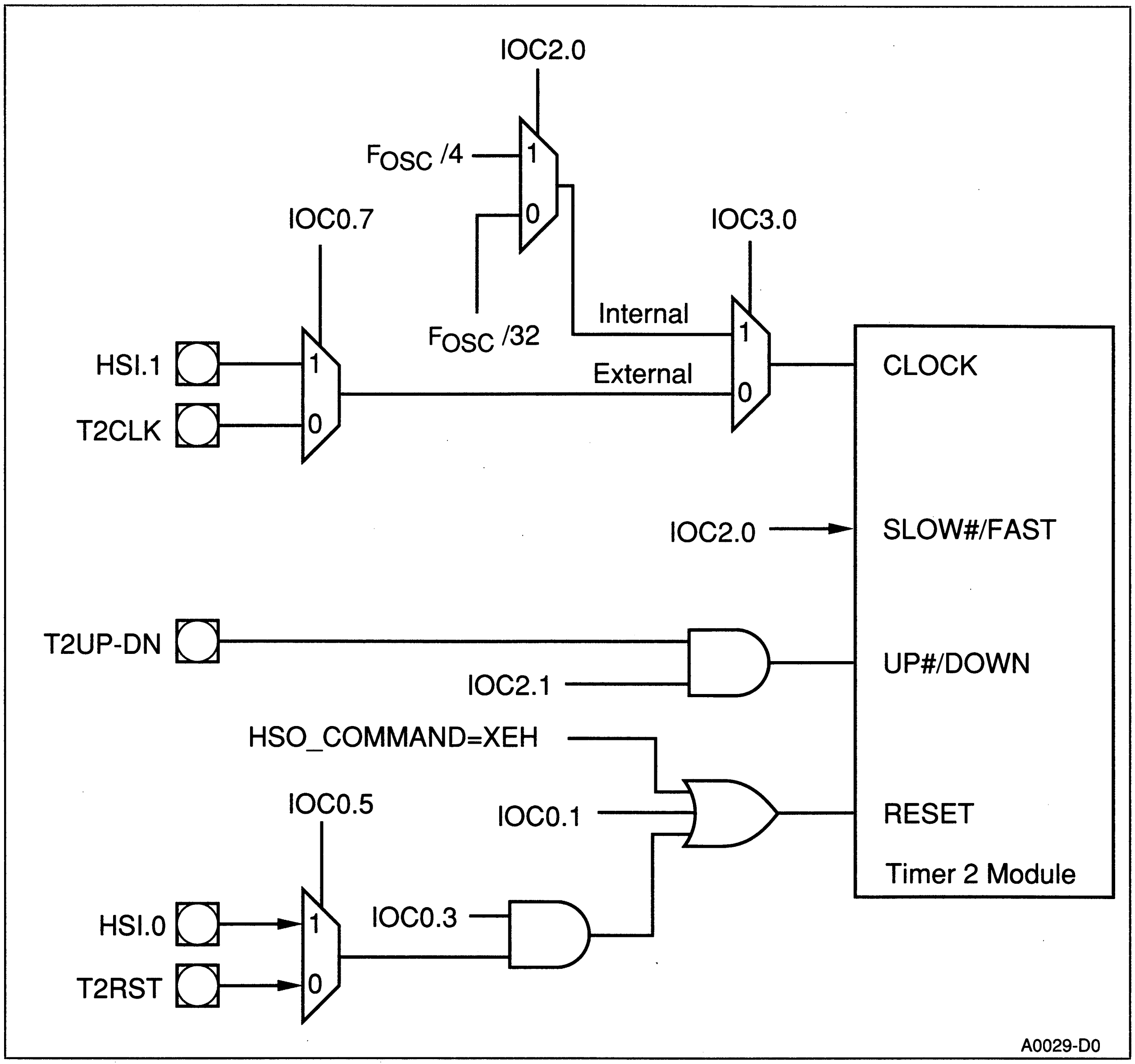


Figure 8-2. Timer 2 Control Logic

8.1.3.3. SELECTING TIMER 2 RESET

Timer 2 can be reset by hardware, software, or the HSO module (see Figure 8-2). Setting the T2RST_ENA bit (IOC0.3) enables an external reset source. Setting the T2RST_SRC bit (IOC0.5) selects the HSI.0 pin and clearing it selects the T2RST (P2.4) pin as the external Timer 2 reset signal. The rising edge of the selected signal resets Timer 2.

Software can reset this timer by setting the SW_T2RST bit (IOC0.1). The HSO can reset this timer by executing the Reset Timer 2 command (CMD_TAG = 0EH). This reset option is particularly useful when using the HSO module to generate pulse width modulated (PWM) outputs. (See "Using the HSO Module to Generate PWM Outputs" on page 8-23.)

8.1.4. Using the External Timer 2 Inputs

The T2UP-DN, T2CLK, T2RST, and T2CAPTURE pins are sampled while CLKOUT is low. These inputs must be stable for at least one full state time and must be valid at least 45 ns before the rising edge of CLKOUT to guarantee recognition.

8.1.4.1. SYNCHRONIZATION

T2RST is always synchronized to an internal, modulo-8 counter. The amount of time it takes to reset the timer depends upon when T2RST is asserted; the timer reset can take from one to nine state times after T2RST is asserted. This is true for both normal and fast increment modes.

During normal increment mode, both T2CLK and T2CAPTURE are also synchronized to the internal, modulo-8 counter. During fast increment mode, these signals pass directly to Timer 2. In all cases, when both signals are asserted simultaneously, the internal design of Timer 2 ensures that the capture will always occur before the clock increments.

8.1.4.2. ASSERTING T2RST, T2CLK, AND T2CAPTURE SIMULTANEOUSLY

Timer 2 behaves predictably when T2RST, T2CLK, and T2CAPTURE occur simultaneously. The behavior of Timer 2 is determined by the time that the signals are asserted in relation to the internal, modulo-8 counter. The internal modulo-8 counter has eight finite *states* numbered one through eight. The counter increments from one state to the next. After it reaches state eight, it rolls over to state one and continues incrementing in an endless loop. The T2CAPTURE and T2CLK signals are sent to Timer 2 during state one; the T2RST signal is sent to Timer 2 during state 2. Since it is **not** possible to synchronize events to this internal counter, you should compensate with either software or hardware to prevent problems.

8.1.4.2.1. Normal Increment Mode

When T2RST, T2CLK, and T2CAPTURE are simultaneously asserted during state 1, the reset occurs first.

Event Sequence 1	Contents of TIMER2	Contents of T2CAPTURE
1. Reset Timer 2	0000H	????H
2. Capture External Event	0000H	0000H
3. Increment Timer 2	0001H	0000H

When the signals are simultaneously asserted during states 2–8, the capture and increment occur before the reset.

Event Sequence 2	Contents of TIMER2	Contents of T2CAPTURE
1. Capture External Event	5A56H	5A56H
2. Increment Timer 2	5A57H	5A56H
3. Reset Timer 2	0000H	5A56H

8.1.4.2.2. Fast Increment Mode

When T2RST, T2CLK, and T2CAPTURE are simultaneously asserted during state 1, 3, 4, 5, 6, 7, or 8, the capture and increment occur before the reset.

Event Sequence 1	Contents of TIMER2	Contents of T2CAPTURE
1. Capture External Event	5A56H	5A56H
2. Increment Timer 2	5A57H	5A56H
3. Reset Timer 2	0000H	5A56H

When they are simultaneously asserted during state 2, the capture occurs, then the reset, and finally the increment.

Event Sequence 2	Contents of TIMER2	Contents of T2CAPTURE
1. Capture External Event	5A56H	5A56H
2. Reset Timer 2	0000H	5A56H
3. Increment Timer 2	0001H	5A56H

8.1.5. Timer Interrupts

Three interrupt vectors are associated with Timer 1 and Timer 2.

- Timer Overflow Interrupt
- Timer 2 Overflow Interrupt
- Timer 2 Capture Interrupt

The Timer Overflow Interrupt was a shared vector for both timers on the 8096BH. The other two are new on the 8XC196KC/KD.

The IOS1 register contains flags that indicate which events triggered interrupts. Reading the IOS1 register clears bits 0–5. For this reason, we recommend that you copy the contents of the IOS1 register into a shadow register and then execute bit-test instructions such as JBC or JBS on the shadow register.

8.1.5.1. TIMER OVERFLOW INTERRUPT

Both Timer 1 and Timer 2 can trigger the Timer Overflow interrupt (INT00). Set INT_MASK.0 to enable this interrupt. Set either IOC1.2 (Timer 1) or IOC1.3 (Timer 2) to select the source of the interrupt. When an overflow occurs, a status flag is set in the IOS1 register. A Timer 1 overflow sets IOS1.5 and a Timer 2 overflow sets IOS1.4.

8.1.5.2. TIMER 2 OVERFLOW INTERRUPT

Timer 2 can generate the Timer 2 Overflow interrupt (INT12) instead of the standard Timer Overflow interrupt. This interrupt is enabled by setting INT_MASK1.4. A Timer 2 overflow sets IOS1.4.

Timer 2 can generate the Timer 2 overflow interrupt at either the 0FFFFH/0000H or 7FFFH/8000H boundary. An overflow can occur in either direction. IOC2.5 selects the overflow boundary. When IOC2.5 is set, Timer 2 interrupts at the 7FFFH/8000H boundary. Otherwise, it interrupts at the 0FFFFH/0000H boundary.

8.1.5.3. TIMER 2 CAPTURE INTERRUPT

A positive transition on the T2CAPTURE pin causes the value of Timer 2 to be loaded into the T2CAPTURE register. This event generates a Timer 2 Capture interrupt (INT11) if INT_MASK1.3 is set and T2CAPTURE is asserted for more than two states times.

8.1.6. Timer Precautions

When using the timers as the HSI or HSO time base, following these guidelines will help you to avoid potential problems. The remainder of this section explains the guidelines and outlines the potential problems.

- Use caution when writing to the timer registers, TIMER1 and TIMER2.
- Configure Timer 2 to operate in normal mode (not in fast increment mode).
- Configure Timer 2 to count in only one direction.
- Use caution when resetting Timer 2.
- When Timer 2 is configured to be reset by an external pin, program events to occur when Timer 2 is equal to one rather than zero.

Changing the TIMER1 value after initiating the HSI module may corrupt relative references between HSI events. Also, changing the reference timer value (TIMER1 or TIMER2) after initiating the HSO module may cause programmed HSO events to be missed or to occur in the wrong order.

Because the HSO requires eight state times for a complete CAM search, Timer 2 must operate in its normal operating mode (not in fast increment mode) when it is used as the HSO reference. Clear the FAST_T2_ENA bit (IOC2.0) to select normal operating mode.

Timer 2 should count in only one direction when it is used as the HSO reference. Using Timer 2 as an up/down counter could cause events to be missed or to occur in the opposite order. Also, if Timer 2 oscillates around an entry's time tag, locked entries could occur several times. Clear the T2UD_ENA bit (IOC2.1) to configure Timer 2 as an up counter.

Do not reset Timer2 before its value reaches the highest programmed Timer 2 time in the CAM. The CAM holds an event pending until a time match occurs. If the Timer 2 value is never reached, the event will remain pending until the device is reset or the CAM is cleared.

When Timer 2 is configured to be reset by an external reset pin (IOC0.3 set), do not program events to occur when Timer 2 is equal to zero. If HSI.0 or T2RST (P2.3) resets Timer 2, the event may not occur. The external pins clear Timer 2 asynchronously, and Timer 2 may be incremented to one before the HSO can compare and act upon the CAM entry. Programming events to occur when Timer 2 is equal to one ensures that the HSO has sufficient time to recognize the CAM entry.

8.2. HIGH-SPEED INPUT MODULE

The High-Speed Input (HSI) module monitors four external input pins (HSI.0–HSI.3). When a predefined event occurs on one or more of the pins, the HSI module records the current Timer 1 count value along with a status bit for each input. The HSI module stores data for up to seven events in a 7 × 20-bit FIFO queue and for one additional event in a holding register. Data moves from the FIFO to the holding register and can be read only after it is loaded into the holding register.

The HSI can be programmed to capture each positive transition, each negative transition, every transition (both positive and negative), or every eighth positive transition. This flexibility enables you to use the HSI module to measure a variety of inputs (i.e., pulse widths, periods, duty cycles, phase differences, etc.). Recording every eighth positive transition allows faster pulse rates to be measured and counted.

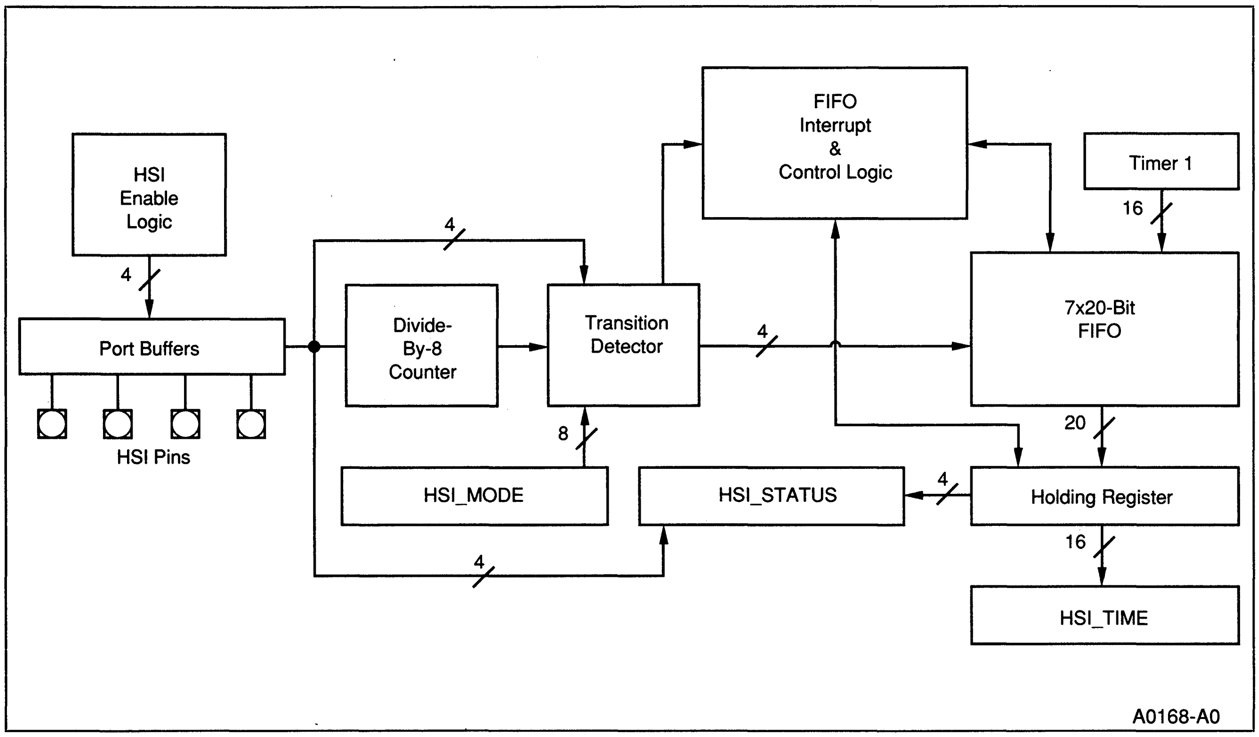


Figure 8-3. HSI Block Diagram

8.2.1. HSI Event Timing Considerations

Events are loaded into the HSI FIFO at the maximum rate of one event every nine state times. Always allow at least nine state times between consecutive events on each pin; otherwise, the HSI may miss an event. If the pin is configured for the eighth transition mode, the maximum rate for positive transitions is eight per 16 state times.

When monitoring repetitive events that occur once per nine state times, a mismatch between the nine-state HSI resolution and the eight-state timer causes one time-tag value to be skipped every nine timer counts. If an event occurs just before the timer increments, and the next event occurs nine state times later, the timer will increment twice between the two events (see Figure 8-4). The result is a FIFO entry with a time tag one count later than expected every nine counts.

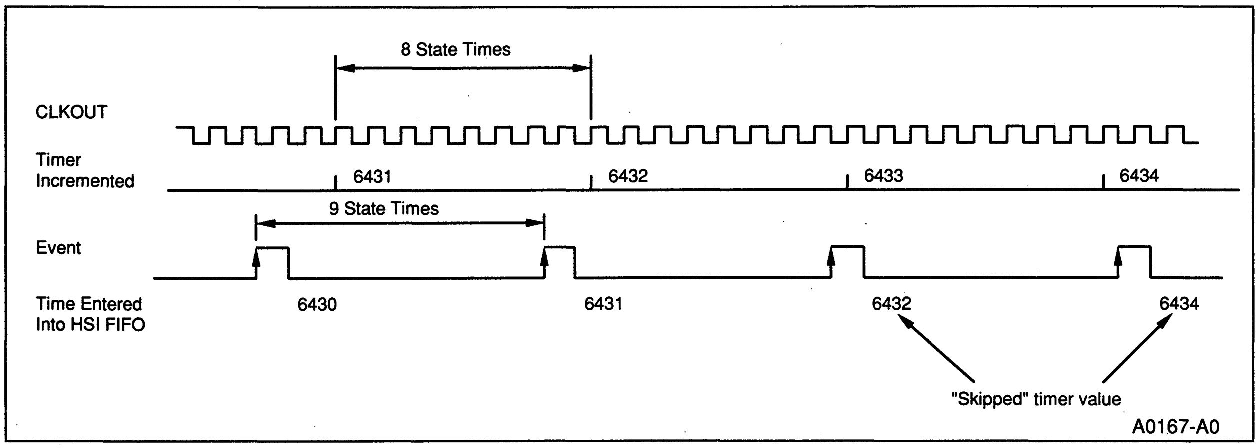


Figure 8-4. Skipped Timer Values

The first event into an empty FIFO will transfer to the holding register after eight state times. If a second event occurs after the first moves to the holding register, it is treated as the first event into an empty FIFO. Additional events are not recorded if both the FIFO and holding register are full.

If the first two events into an empty FIFO (not including the holding register) occur during the same internal phase, both are recorded with the same time tag. Otherwise, if the second event occurs within nine states after the first, its time tag is one count later than the tag time of the first. If this is the *skipped* timer value, the second event's time tag is two counts later than the time tag of the first.

The change detection logic samples the HSI pins while CLKOUT is low. To guarantee that the HSI event is recorded with the correct Timer 1 value, the HSI inputs must either be stable for at least one full state time or be valid at least 45 ns before the rising edge of CLKOUT. If neither condition is met, the HSI input may be sampled in the next CLKOUT. Therefore, if information is to be synchronized to the HSI module, it should be latched on the rising edge of CLKOUT.

The HSI module's internal divide-by-eight counter continuously counts transitions. When you program an HSI input to detect every eighth positive transition, the counter is not reset. It retains its accumulated count and continues counting. If the accumulated count was six, then just two additional positive transitions will trigger an HSI event capture. For this reason, it is a good idea to ignore the first HSI event when using the eight transitions mode.

8.2.2. Reading the Event Data

The event data can be read only after it is loaded into the holding register. Allow eight state times for data to move through the FIFO and into the holding register. When an event moves from the FIFO into the holding register, the HSI_RDY bit is set in the IOS1 register (IOS1.7). The HSI Data Available interrupt (INT02) is generated if the interrupt is enabled and the event is selected as the source of the interrupt. (See "Enabling HSI Interrupts" on page 8-14 for details).

To unload the holding register, first read the HSI_STATUS register. This causes the holding register to load the event status bits for HSI.0-HSI.3 into the even bits of the HSI_STATUS register. When set, the event status bits indicate that an event occurred on the corresponding HSI pin. The odd bits always contain the current state of the HSI.0–HSI.3 pins.

Then read the HSI_TIME register. The holding register loads the Timer 1 count value into the HSI_TIME register. Reading the HSI_TIME register causes the FIFO to load the next event into the holding register. If you read the HSI_TIME register before first reading the HSI_STATUS register, the event status bits are overwritten.

8.2.3. Programming the HSI Module

Table 8-7 lists the registers that affect the performance and function of the HSI module. Refer to Appendix C, "8XC196KC/KD Registers," for detailed information.

Table 8-3. HSI Control and Status Registers

Register Mnemonic	Register Name	Description
HSI_MODE	HSI Mode	Defines the type of transition that will cause an event on each HSI pin.
HSI_STATUS	HSI Status	Contains the event-detection bits and the current input level for each HSI pin.
HSI_TIME	HSI Time	Contains the 16-bit value of TIMER1 at the time of the event.
INT_MASK INT_MASK1	Interrupt Mask Interrupt Mask 1	Enables or disables the HSI interrupts.
IOC0	Input/Output Control Register 0	Enables or disables each HSI pin. Selects alternate functions for HSI.0 and HSI.1.
IOC1	Input/Output Control Register 1	Selects the source for the HSI Data Available interrupt (INT02).
IOS1	Input/Output Status Register 1	Indicates the status of the HSI FIFO.

8.2.3.1. DEFINING AN HSI EVENT

The HSI_MODE register controls, for each HSI pin, which of four types of events trigger a capture into the HSI FIFO: each positive transition, each negative transition, every transition (both positive and negative), or a series of eight positive transitions (see Figure 8-5).

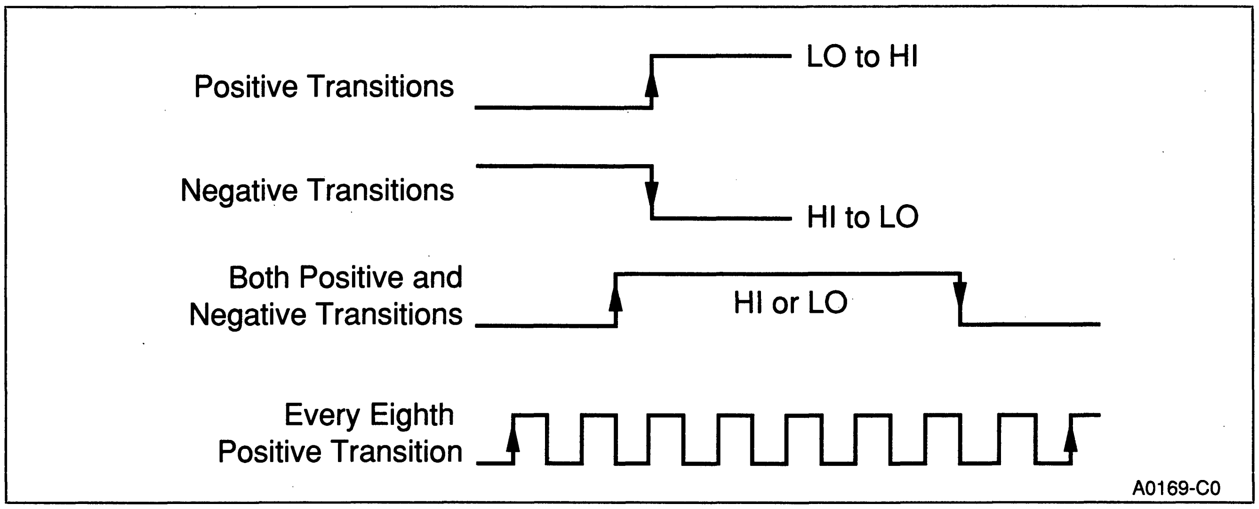


Figure 8-5. HSI Event Options

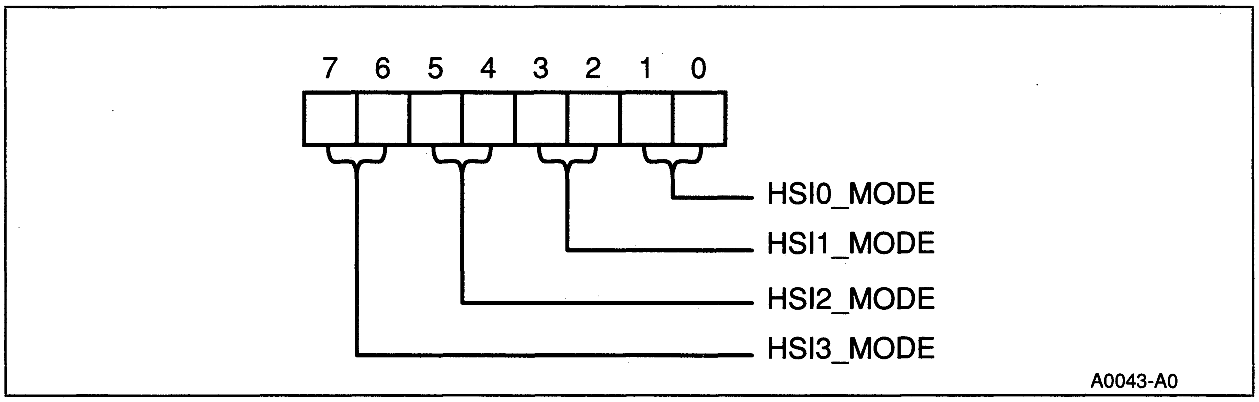


Figure 8-6. HSI_MODE Register

Each two-bit field within the HSI_MODE register is programmed to define the transition mode for the corresponding HSI input (see Figure 8-6 and Table 8-4).

Table 8-4. Transition Mode Encoding

HSI*x*_MODE	Description
00	Eight positive transitions trigger a capture into the HSI FIFO.
01	Each positive transition triggers a capture into the HSI FIFO.
10	Each negative transition triggers a capture into the HSI FIFO.
11	Every transition (both positive and negative) triggers a capture into the HSI FIFO.

8.2.3.2. ENABLING HSI INTERRUPTS

The HSI can generate an interrupt when any of the following events occur.

- The fourth entry is loaded into the FIFO.
- The sixth entry is loaded into the FIFO.
- An entry moves from the FIFO into the holding register.

In addition, the HSI.0 pin can be configured to function as an independent, external interrupt. It need not be enabled as an HSI input to function as an external interrupt.

The INT_MASK and INT_MASK1 registers contain bits that enable and disable each interrupt. Table 8-5 briefly describes the conditions that cause each HSI interrupt and then lists its interrupt number and priority.

Table 8-5. HSI Interrupts

Interrupt Source	Generated when	Interrupt Name	Priority (Note 1)
HSI FIFO Full (Note 2)	INT_MASK1.6 = 1 and the sixth entry is loaded into the FIFO. Holding register entries are not counted.	HSI FIFO Full (INT14)	14
	IOC1.7 = 1, INT_MASK.2 = 1, INT_MASK1.6 = 0, and the sixth entry is loaded into the FIFO. Holding register entries are not counted.	HSI Data Available (INT02)	2
HSI FIFO 4	INT_MASK1.2 = 1, and the fourth entry is loaded into the FIFO. Holding register entries are not counted.	HSI FIFO 4 (INT10)	10
HSI.0 External	INT_MASK1.4 = 1 and the HSI.0 signal transitions from low to high and remains high for at least one state time. This interrupt can be configured to function as an independent, external interrupt, the HSI.0 need not be enabled as an input to the HSI module.	HSI.0 Pin (INT04)	4
HSI Data Available	IOC1.7 = 0, INT_MASK.2 = 1, and an entry moves from the FIFO into the holding register. This interrupt indicates that at least one HSI event has occurred and is ready for processing.	HSI Data Available (INT02)	2

NOTES:

1. 15 is the highest priority; 0 is the lowest.
2. The sixth FIFO entry can trigger either the HSI Data Available interrupt (INT02) or the HSI FIFO Full interrupt (INT14) but should not be configured for both.

8.2.3.3. ENABLING AND DISABLING THE HSI PINS

You can individually enable or disable the HSI input function of each HSI pin by setting or clearing the corresponding bit in the IOC0 register (see Figure 8-7 and Appendix C, "8XC196KC/KD Registers," for a complete description of the IOC0 register).

Setting the appropriate bit in the IOC0 register connects the pin to the HSI transition detector and divide-by-eight counter. Clearing the bit disconnects the pin from the HSI logic, but not from the HSI_STATUS register (see Figure 8-7). Disabling the HSI input function does not disable or prevent the use of each pin's additional functions.

8.2.3.4. ASSIGNING ADDITIONAL FUNCTIONS TO HSI PINS

You can optionally program the HSI pins to provide alternate functions. Table 8-6 lists the additional functions for each HSI output pin. Column three describes how to select the additional function.

Table 8-6. Additional Functions for HSI Pins

HSI Function Name	Additional Functions	Selected by
HSI.0	HSI.0 Pin interrupt (INT04)	Setting INT_MASK1.4 enables the HSI.0 Pin interrupt (INT04). Assert HSI.0 for greater than two state times to guarantee recognition.
	Timer 2 reset source	Setting IOC0.5 selects HSI.0 as the Timer 2 reset source. When enabled, a rising edge on HSI.0 resets Timer 2.
HSI.1	Timer 2 clock source	Clearing IOC0.7 and IOC3.0 selects the HSI.1 pin as the Timer 2 clock source.
HSI.2	HSO.4	Setting IOC1.4 enables the output function of HSO.4. Note that the HSI and HSO functions can be active at the same time.
HSI.3	HSO.5	Setting IOC1.6 enables the output function of HSO.5.Note that the HSI and HSO functions can be active at the same time.

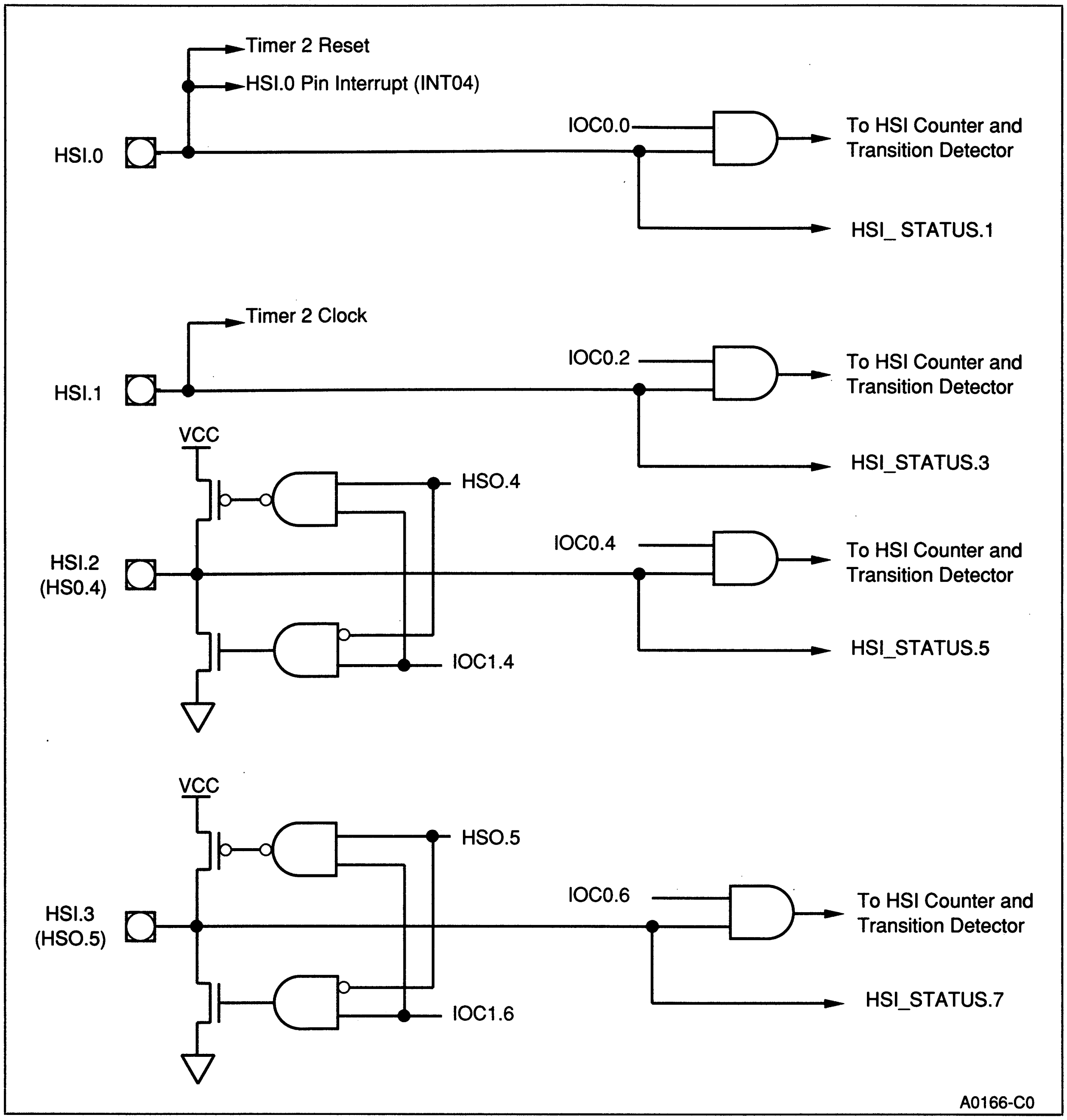

Timer 2 Reset
HSI.0 Pin Interrupt (INT04)
HSI.0
IOC0.0
To HSI Counter and Transition Detector
HSI_ STATUS.1
Timer 2 Clock
HSI.1
IOC0.2
To HSI Counter and Transition Detector
HSI_STATUS.3
VCC
HSO.4
IOC0.4
HSI.2 (HS0.4)
To HSI Counter and Transition Detector
HSI_STATUS.5
IOC1.4
VCC
HSO.5
IOC0.6
HSI.3 (HSO.5)
To HSI Counter and Transition Detector
HSI_STATUS.7
IOC1.6
A0166-C0

Figure 8-7. HSI Input Options

8.3. HIGH-SPEED OUTPUT MODULE

The High-Speed Output (HSO) module triggers events at specified times using either Timer 1 or Timer 2 as the time reference. These programmable events include starting an A/D conversion, resetting Timer 2, generating up to four software time delays, and setting or clearing one or more of the HSO outputs (HSO.0–HSO.5). The HSO module stores up to eight pending events and their specified times in a Content-Addressable Memory (CAM) file. Figure 8-8 is a block diagram of the HSO module.

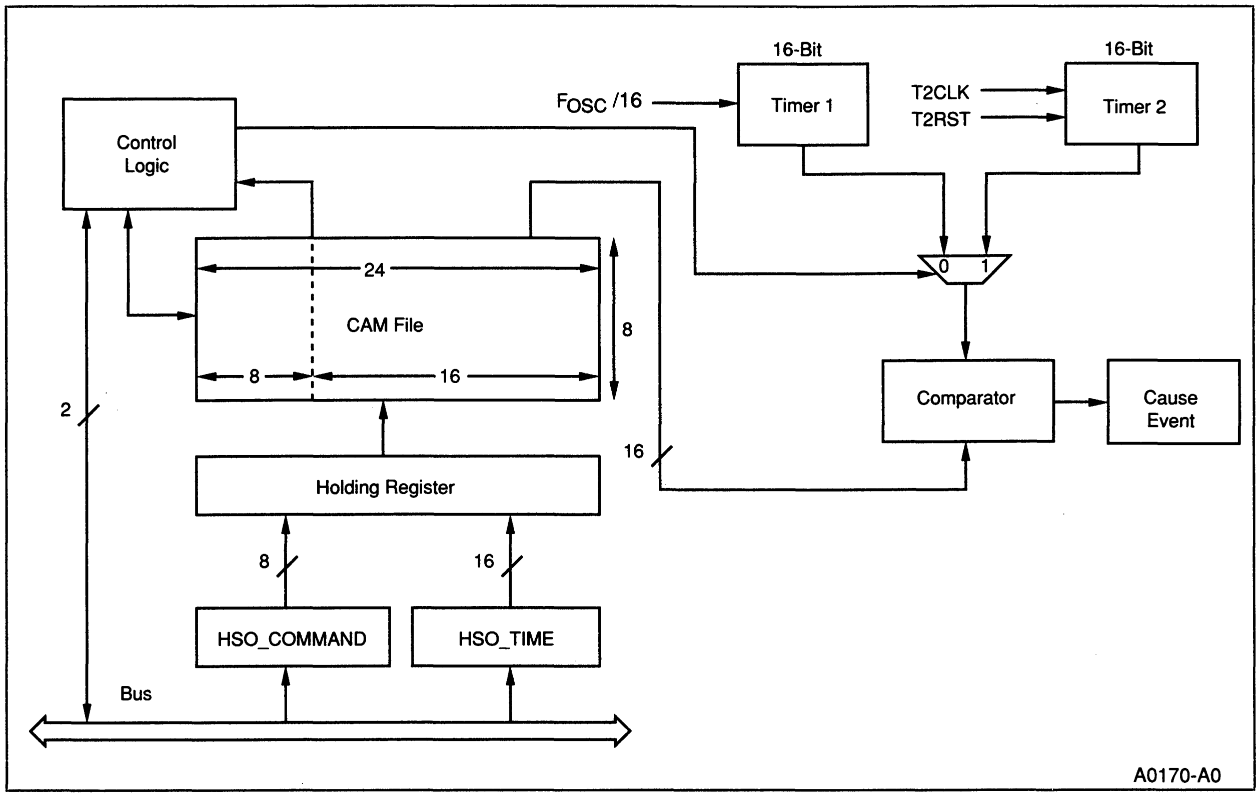


Figure 8-8. HSO Block Diagram

8.3.1. HSO Functional Overview

The CAM file is the main component of the HSO module. This file stores up to eight commands. Each CAM entry is 24 bits wide. Sixteen bits are loaded from the HSO_TIME register to specify the action time, and eight bits are loaded from the HSO_COMMAND register to specify the nature of the action, whether it will generate an interrupt, and whether Timer 1 or Timer 2 is the reference.

NOTE

When Timer 2 is used as the HSO module's reference timer, it must operate in normal mode (not fast increment mode) and should count in only one direction. See "Timer Precautions" on page 8-9 for additional information.

The HSO compares all eight CAM entries with the timer value before triggering an event. It takes one state time to compare each CAM entry, so eight state times are needed for a complete CAM search. The HSO triggers the event when the selected timer matches the programmed time value.

The HSO can trigger 15 different events, which are classified as *external* or *internal.* The *external events* are those that toggle one or more of the HSO outputs; the *internal events* are those that set up the software timers, reset Timer 2, and start an A/D conversion. All of these events set flags that can optionally generate interrupts. External events generate the High-Speed Output interrupt (INT03); internal events generate the Software Timer interrupt (INT05).

The four software timers provide a means for generating interrupts at predefined times. They are most commonly used to trigger interrupt routines that must occur at regular intervals. At the specified time, the HSO sets a flag in the IOS1 register; if the interrupt is enabled, it also generates a Software Timer interrupt. If more than one software timer expires within the same time frame, multiple status bits are set. The interrupt service routine can examine the IOS1 register to determine which software timer(s) caused the interrupt.

8.3.2. Programming the HSO Module

Table 8-7 lists the registers that affect the performance and function of the HSO module. Refer to Appendix C, "8XC196KC/KD Registers," for detailed information.

Table 8-7. HSO Control and Status Registers

Register Mnemonic	Register Name	Description
HSO_COMMAND	HSO Command	Determines what event or events the HSO will trigger at the specified time.
HSO_TIME	HSO Time	Specifies the time at which an HSO command is to be executed.
INT_MASK	Interrupt Mask	Enables or disables the HSO interrupts.
IOC1	Input/Output Control Register 1	Enables or disables HSO.4 and HSO.5 as outputs.
IOC2	Input/Output Control Register 2	Enables and disables command locking within the HSO CAM file; can also clear the HSO CAM.
IOS0	Input/Output Status Register 0	Indicates the current state of the HSO pins, holding register, and CAM. Writing to the bits can set or clear the corresponding HSO pin.
IOS1	Input/Output Status Register 1	Contains flags that indicate which internal events triggered interrupts.
IOS2	Input/Output Status Register 2	Contains flags that indicate which external HSO events have occurred.

8.3.2.1. PROGRAMMING HSO EVENTS

The contents of the HSO_TIME and HSO_COMMAND registers together define each HSO event. The HSO_TIME register specifies the time at which an HSO command is to be executed. Bits 0–3 of the HSO_COMMAND register contain the command that determines what event or events the HSO will trigger at the specified time (see Figure 8-9). Table 8-9 lists the HSO command tags. The HSO_COMMAND register also determines whether the event will generate an interrupt, controls whether a pin toggle command sets or clears the affected pin(s), selects the timer reference for the HSO command, and controls whether the command is locked into the CAM or cleared upon execution (see Table 8-9).

Table 8-8. CMD_TAG Encoding

CMD_TAG (in Hex)	Command Mnemonic	Definition
00	HSO0	Switch High-Speed Output 0
01	HSO1	Switch High-Speed Output 1
02	HSO2	Switch High-Speed Output 2
03	HSO3	Switch High-Speed Output 3
04	HSO4	Switch High-Speed Output 4
05	HSO5	Switch High-Speed Output 5
06	HSO01 *	Switch High-Speed Outputs 0 and 1
07	HSO23 *	Switch High-Speed Outputs 2 and 3
08	SWT0	Program Software Timer 0
09	SWT1	Program Software Timer 1
0A	SWT2	Program Software Timer 2
0B	SWT3	Program Software Timer 3
0C	HSOALL *	Switch High-Speed Outputs 0, 1, 2, 3, 4, 5
0D	—	Reserved; do not use
0E	T2RST	Reset Timer 2
0F	A_D	Start an A/D Conversion

* In these configurations, two or more pins are set or cleared simultaneously.

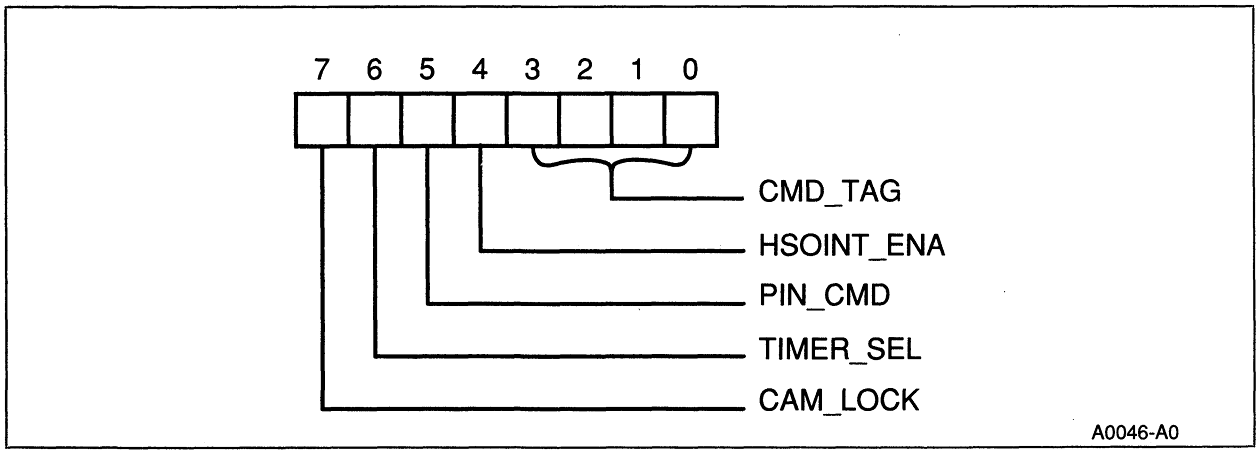


Figure 8-9. HSO_COMMAND Register

Table 8-9. HSO_COMMAND Register Bit Descriptions

Bit Number(s)	Bit Mnemonic	Bit Name	Description
0–3	CMD_TAG	HSO Commands	Determines what event or events will occur at the time specified in HSO_TIME. (See the "CMD_TAG Encoding" table.)
4	HSOINT_ENA	Enable/ Disable HSO Interrupt	Determines whether an HSO event generates an interrupt. 1= generate an interrupt 0= no interrupt When this bit is set, programmed HSO pin events generate the High-Speed Output interrupt (INT03, 2006H) and internal events generate the Software Timer interrupt (INT05, 200AH). When an interrupt is generated, the HSO event status bits in the IOS1 and IOS2 registers must be interrogated to determine which event caused the interrupt.
5	PIN_CMD	Set/Clear Selected HSO Pin	Determines whether the effect of a CMD_TAG command sets or clears the specified pin or pins. 1= Set the pin(s) 0= Clear the pin(s)
6	TIMER_SEL	Select Timer 1/ Timer 2	Selects the timer reference for the HSO command. 1= Select Timer 2 0= Select Timer 1
7	CAM_LOCK	Lock Entry Into CAM	When IOC2.6 is set (command locking enabled), this bit controls whether the HSO command is locked into the CAM or cleared after execution. 1= Lock command into CAM 0= Clear command from CAM after execution Writing a "1" to IOC2.7 clears all entries (locked or not) from the CAM, as does a chip reset.

NOTE

Before programming the HSO_COMMAND and HSO_TIME registers, read either IOS0.7 or IOS0.6 to verify that the HSO holding register is empty. If IOS0.7 is cleared, the holding register is empty. If IOS0.6 is cleared, the holding register is empty **and** at least one CAM register is empty. Writing to HSO_TIME when the holding register is not empty overwrites the current holding register value.

To enter a command into the CAM file, first write to the HSO_COMMAND register to define the event, and then write to the HSO_TIME register to specify the relative time at which the event is to occur. Writing to the HSO_TIME registers loads the HSO holding register. The command is held in the holding register until an empty CAM register is available, at which time the command enters the CAM file. It can take up to eight state times for a command to move from the holding register to the CAM file, so always allow at least eight state times between consecutive writes to the HSO_COMMAND and HSO_TIME registers. Otherwise, the command in the holding register can be overwritten before it is loaded into the CAM file.

NOTE

Commands in the holding register will not execute, even if their time tag is reached. Commands must be in the CAM file to execute.

The value in HSO_TIME can be either a specific timer value or a value that is offset from the current timer value. Use the standard load instruction to program a specific time. Otherwise, use a three-operand ADD instruction to specify a value that is offset from the current timer value.

```
LDB HSO_COMMAND, #what_to_do                ; Loads command
ADD HSO_TIME, TIMER1, #when_to_do_it        ; ADD_3op
```

To provide proper synchronization, the minimum time that should be loaded into a timer is *current_timer_value* + 2. Smaller values may cause the timer match to occur 65,636 counts later than expected. This restriction applies to both Timer 1 and Timer 2.

Use care when writing to the HSO_COMMAND and HSO_TIME registers. An interrupt can occur after the command tag is written to HSO_COMMAND and before the time value is loaded into HSO_TIME. If the interrupt routine writes to these registers, its command tag overwrites the main program's command tag in the CAM file, so the main program's command is never executed. Disabling interrupts when writing to the HSO module prevents this problem. (See "Modifying Interrupt Priorities" in Chapter 5.)

8.3.2.2. ENABLING THE HSO INTERRUPTS

Setting the HSOINT_ENA bit (HSO_COMMAND.4) causes an HSO event to generate an interrupt. The HSO module can generate two different interrupts: the High-Speed Output interrupt (INT03) and the Software Timer interrupt (INT05).

External events generate the High-Speed Output interrupt, if the interrupt is enabled (INT_MASK.3 set). The status bits related to the High-Speed Output interrupt are IOS2.0–IOS2.5. *Internal events* generate the Software Timer interrupt, if the interrupt is enabled (INT_MASK.5 set). The status bits related to the Software Timer interrupt are IOS1.0–IOS1.3 for the software timer commands and IOS2.6–IOS2.7 for Timer 2 reset and A/D conversion start, respectively.

When an interrupt is generated, read the HSO event status bits in the IOS1 and IOS2 registers to determine which event caused the interrupt. Reading the IOS1 register clears bits 0–5; reading the IOS2 register clears all bits. For this reason, we recommend that you copy the contents of the registers into shadow registers and then execute bit-test instructions such as JBC or JBS on the shadow registers.

8.3.2.3. LOCKING CAM ENTRIES

The LOCK_ENA bit in the IOC2 register (IOC2.6) enables and disables command locking. When this bit is set, the CAM_LOCK bit (HSO_COMMAND.7) controls whether individual commands are locked into the CAM or cleared after execution. When CAM_LOCK is set, the command remains in the CAM after execution. This feature makes it easy to generate periodic events or waveforms. Clearing CAM_LOCK causes the HSO to clear the command from the CAM after execution.

Locked entries provide the ability to program periodic or repetitive events, while minimizing the software overhead. One of the most useful features is that locked entries enable the HSO to generate multiple pulse width modulated (PWM) outputs, with minimum software overhead. (See "Using the HSO Module to Generate PWM Outputs.") A locked A/D conversion command causes the HSO to generate multiple A/D conversions. When Timer 2 is used as the HSO reference, a locked T2RST command can generate periodic events; events with HSO_TIME less than the Timer 2 reset value occur repeatedly as Timer 2 resets. Locked software timer commands can schedule recurrent software tasks; interrupt service routines can execute the tasks without having to program another HSO software timer command.

8.3.2.4. CLEARING THE CAM FILE

Three occurrences can remove an entry from the CAM:

- if the specified timer matches the programmed value and the event is not locked
- the device is reset
- the CAM_CLR bit (IOC2.7) is set

To clear locked events, set the CAM_CLR bit (IOC2.7) or reset the device. Either of these options will clear the entire CAM.

8.3.2.5. CANCELING AN EVENT

You can cancel an *external event* by writing the opposite event with the same time tag to the CAM. For example, if a command that sets HSO.1 when TIMER1 = 1234 is followed by a command that clears HSO.1 when TIMER1 = 1234, HSO.1 does not toggle. However, both commands remain in the CAM until either TIMER1 = 1234 (if the entries are not locked), the device is reset, or the CAM_CLR bit (IOC2.7) is set.

You can cancel an *internal event* only by setting the CAM_CLR bit (IOC2.7) or resetting the device.

8.3.2.6. ENABLING THE HSO.4 AND HSO.5 PINS

The HSO.4 and HSO.5 outputs are multiplexed with the HSI.2 and HSI.3 inputs, respectively. These pins can be enabled for both functions (see Figure 8-7 on page 8-16). Setting the HSO4_ENA bit (IOC1.4) enables HSO.4 as an output; setting the HSO5_ENA bit (IOC1.6) enables HSO.5.

8.3.2.7. USING HSO.0–HSO.6 AS OUTPUT PINS

The HSO outputs can provide up to six additional output pins if the HSO function is not needed. To use the HSO pins as standard outputs, enable the pins and then set or clear them by writing to the IOS0 register in HWindow 15.

8.3.3. Using the HSO Module to Generate PWM Outputs

Both the HSO module and the PWM modules can generate a rectangular pulse train that varies in duty cycle and period. With proper components, a highly accurate 8-bit D/A converter can be made using either the HSO or the PWM outputs. (See "Generating Analog Outputs" in Chapter 10.)

The HSO module can generate either a single PWM waveform or multiple PWM waveforms. A single waveform requires a programmed *period* and *clear_time* for the output pin and two HSO commands: one to set the pin and one to clear it. Either Timer 1 or Timer 2 can be used as the reference.

Generating multiple waveforms is done in a similar way, using Timer 2 as the reference. Multiple waveform generation requires a programmed *period* for Timer 2, a set time (*PWMx_ST*) and a clear time (*PWMx_CT*) for each output pin, and multiple HSO commands: one to reset Timer 2, several to set the pins, and several to clear the pins.

The following example shows a common way to generate multiple PWMs using HSO.0, HSO.1, and HSO.2 as the PWM outputs and Timer 2 as the reference timer. Figure 8-10 shows the three resulting PWMs.

```
SET_0:          LDB HSO_COMMAND, #1110000B          ;set HSO.0 when
                LD HSO_TIME, #PWM0_ST               ;timer2 = pwm0_st
                SKIP R0                             ;wait 4 state times
                SKIP R0                             ;wait 4 state times
SET_1:          LDB HSO_COMMAND, #1110001B          ;set HSO.1 when
                LD HSO_TIME, #PWM1_ST               ;timer2 = pwm1_st
                SKIP R0                             ;wait 4 state times
                SKIP R0                             ;wait 4 state times
SET_2:          LDB HSO_COMMAND, #1110010B          ;set HSO.2 when
                LD HSO_TIME, #PWM2_ST               ;timer2 = pwm2_st
                SKIP ZERO_REG                       ;wait 4 state times
                SKIP ZERO_REG                       ;wait 4 state times
CLEAR_0:        LDB HSO_COMMAND, #1100000B          ;clear HSO.0 when
                LD HSO_TIME, #PWM0_ST               ;timer2 = pwm0_ct
                SKIP ZERO_REG                       ;wait 4 state times
                SKIP ZERO_REG                       ;wait 4 state times
CLEAR_1:        LDB HSO_COMMAND, #1100001B          ;clear HSO.1 when
                LD HSO_TIME, #PWM1_ST               ;timer2 = pwm1_ct
                SKIP ZERO_REG                       ;wait 4 state times
                SKIP ZERO_REG                       ;wait 4 state times
CLEAR_2:        LDB HSO_COMMAND, #1100010B          ;clear HSO.2 when
                LD HSO_TIME, #PWM2_ST               ;timer2 = pwm2_ct
                SKIP ZERO_REG                       ;wait 4 state times
                SKIP ZERO_REG                       ;wait 4 state times
RESET:          LDB HSO_COMMAND, #11001110B         ;reset Timer 2 when
                LD HSO_TIME, #PERIOD                ;timer2 = period
```

Refer to Application Note AP-466, "Using the 80C196KB," for additional information and software examples that show how to use the HSO module to generate single and multiple PWMs.

8.3.4. HSO Output Timing

The timing of the HSO pins is synchronized to the specified timer. All external HSO pins that are due to change at a certain timer value will change just after the timer is incremented. Internally, the timer changes every eight state times during Phase 1. From an external perspective, the HSO pin should change just before the falling edge of CLKOUT and should be stable by its rising edge. Information from the HSO can be latched on the CLKOUT rising edge. Internal HSO events also occur when the reference timer increments.

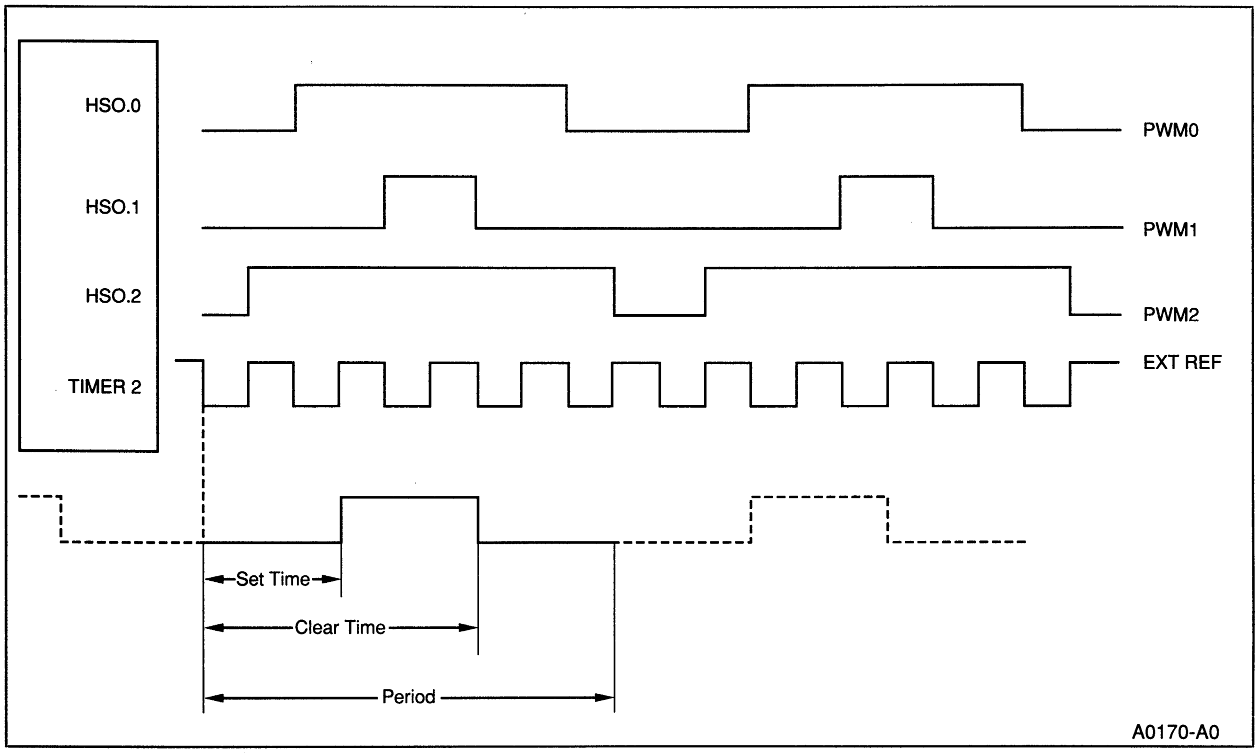


Figure 8-10. Example PWM Waveforms

Analog-to-Digital Converter

9

CHAPTER 9
ANALOG-TO-DIGITAL CONVERTER

This chapter provides an overview of the analog-to-digital (A/D) conversion process and describes how to program the A/D converter, read the conversion results, and interface with external circuitry. It also describes the transfer function and error sources.

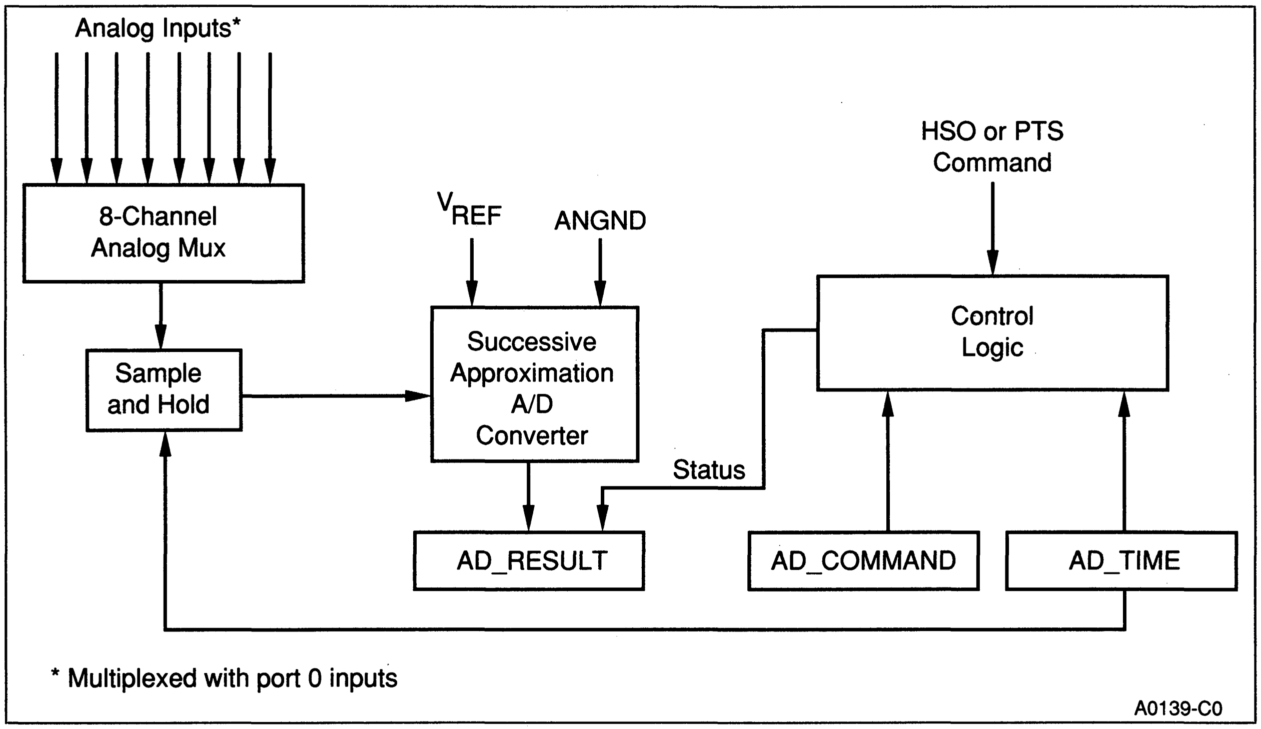


Figure 9-1. A/D Converter Block Diagram

9.1. A/D FUNCTIONAL OVERVIEW

The A/D converter module (shown in Figure 9-1) converts an analog input into an 8- or 10-bit digital representation. The main parts of the A/D converter module are:

- The eight analog inputs (ACH0–ACH7).
- An 8-channel multiplexer to select one of the eight input channels.
- The sample-and-hold circuit.
- The 10-bit successive approximation A/D converter.
- The AD_COMMAND register, which controls the converter operation via the control logic.
- The two-byte AD_RESULT register, which contains the conversion results and status information.
- The AD_TIME register, which contains the sample and conversion times.

Once the A/D converter receives the command to start a conversion, one state time elapses before sampling begins. During this sample delay, the successive approximation register is reset and the designated multiplexer channel is selected.

After the sample delay, the multiplexer output is connected to the sample capacitor and remains connected for the specified sample time. After this *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 A/D converter uses a successive approximation algorithm to perform the analog-to-digital conversion. 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 resistive ladder provides 20 mV steps (V_{REF} = 5.12 Volts), while capacitive coupling creates 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.

The 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 1/2 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 1/4 full scale (0011.1111.11B) is tried. If the analog input was greater than the test voltage, bit 9 of the SAR is set. Bit 8 is then cleared for the next test (0101.1111.11B). This binary search continues until 10 (or 8) tests have occurred, at which time the valid conversion result resides in the SAR where it can be read by software. The result is 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.

9.2. PROGRAMMING THE A/D CONVERTER

The following A/D converter parameters are programmable:

- Either eight-bit or ten-bit conversions
- Input channel selection
- Sample and Convert times
- Interrupt on completion or no interrupt
- 80196KB-compatible mode
- Immediate conversions or timed High-Speed Output (HSO) module or Peripheral Transaction Server (PTS) starts.

Table 9-1 lists the programmable registers that affect that performance and function of the A/D converter.

Table 9-1. A/D Control and Status Registers

Register Mnemonic	Register Name	Description
AD_COMMAND	A/D Command Register	This register selects the A/D channel number to be converted, determines whether the A/D conversion starts immediately or with an HSO command, and selects either the 8-bit or 10-bit conversion mode.
AD_RESULT	A/D Result Register	This register consists of two bytes. The high byte contains the eight most-significant bits from the conversion. The low byte indicates the A/D channel number that was used for the conversion, indicates whether a conversion is currently in progress, and contains the two least-significant bits from a ten-bit conversion.
AD_TIME	A/D Conversion Time	This register defines the sample and convert times, if enabled by the IOC2 register.
IOC2	Input/Output Control Register 2	This register selects whether the A/D conversion times are controlled by the AD_TIME register or by emulating the fast and normal conversion modes of the 8XC196KB.
INT_MASK	Interrupt Mask	This register enables/disables the A/D Conversion Complete interrupt (INT01, 2002H). Setting bit 1 enables this interrupt; clearing bit 1 disables (masks) the interrupt.
INT_PEND	Interrupt Pending	This register indicates that an A/D Conversion Complete interrupt is pending when bit 1 is set. Bit 1 is cleared when the interrupt vectors to 2002H.

9.2.1. Selecting the A/D Sample and Convert Times

Two parameters, sample time and convert time, control the time required for an A/D conversion. The sample time is the length of time that the analog input voltage is actually connected to the sample capacitor. If this time is too short, the sample capacitor will not charge completely. If the sample time is too long, the input voltage may change and cause conversion errors. The convert time is the length of time required to convert the analog input voltage stored on the sample capacitor to a digital value. The convert time must be long enough for the comparator and circuitry to settle and resolve the voltage. Excessively long conversion times allow the sample capacitor to discharge, degrading accuracy.

The sample and convert times either default to a predefined 80C196KB-compatible mode or are specified in the AD_TIME register. Clearing the AD_TIME_ENA bit (IOC2.3) enables an 80C196KB-compatible conversion. In the 80C196KB-compatible mode, the AD_FAST bit (IOC2.4) controls the sample and convert times. The 80C196KB slow mode uses 15 state times for the sample time and a total of 158 state times for the entire conversion. The 80C196KB fast mode uses eight state times for the sample time and a total of 91 state times for the conversion. When using either the fast or slow mode, convert these state times into microseconds to ensure that the data sheet T_{SAM} and T_{CONV} values are met at the operating clock frequency.

Setting the AD_TIME_ENA bit (IOC2.3) enables the AD_TIME register shown in Figure 9-2. The AD_TIME register programs the A/D conversion speed. The resolution and clock frequency determine the conversion speed. Use the T_{SAM} and T_{CONV} specifications on the data sheet to determine appropriate values for sample (SAM) and convert (CONV) times; otherwise, erroneous conversion results may occur.

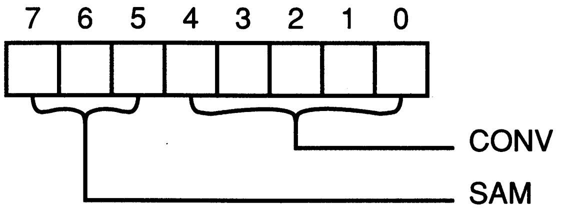


Figure 9-2. AD_TIME Register

Use the following formulas to determine the optimal SAM and CONV values:

$$SAM = \frac{(T_{SAM} \times F_{OSC} - 2)}{2} / 4 \qquad CONV = \left[\frac{(T_{CONV} \times F_{OSC} - 3)}{2} / B\right] - 1$$

where:

SAM	= 1 to 7
CONV	= 2 to 31
T_{SAM}	is the sample time, in µsec, from the data sheet
T_{CONV}	is the conversion time, in µsec, from the data sheet
F_{OSC}	is the XTAL1 frequency, in MHz
B	is the number of bits to be converted (8 or 10)

When the SAM and CONV values are known, write them to the AD_TIME register. Do not write to this register while a conversion is in progress; the results are unpredictable.

9.2.2. Programming the IOC2 Register

The IOC2 register (see Figure 9-3) selects whether the A/D conversion times are controlled by the AD_TIME register or by the AD_FAST bit (IOC2.4). Setting the AD_TIME_ENA bit enables the AD_TIME register and the converter uses the sample and convert times specified in AD_TIME (see Figure 9-2).

Clearing the AD_TIME_ENA bit selects the 80C196KB-compatible mode. In this mode, the AD_FAST bit enables or disables the A/D clock prescaler for complete compatibility with the 80C196KB's slow and fast conversion modes. Clearing AD_FAST enables the 80C196KB slow mode, which uses 15 state times for the sample time and a total of 158 state times for the entire conversion. Setting AD_FAST enables the 80C196KB fast mode, which uses 8 state times for the sample time and a total of 91 state times for the conversion.

NOTE

Changing modes while a conversion is in progress will cause unpredictable results.

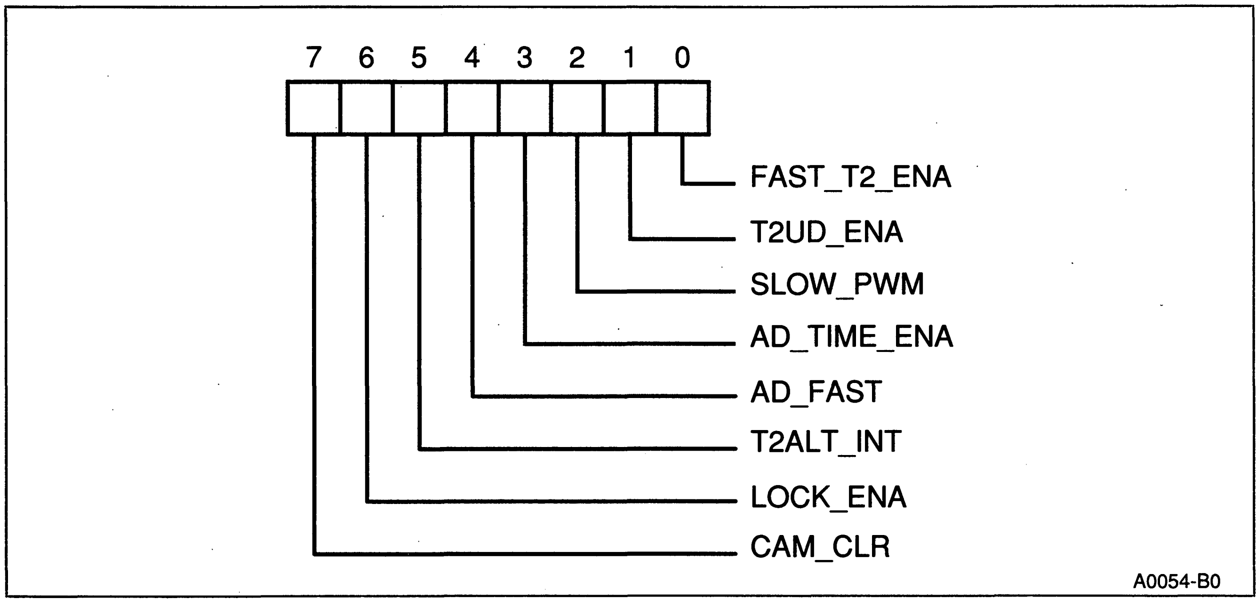


Figure 9-3. The IOC2 Register

9.2.3. Programming the AD_COMMAND Register

The AD_COMMAND register, shown in Figure 9-4, selects the channel to convert, selects an 8-bit or 10-bit conversion, and either starts the conversion or designates the High-Speed Input (HSO) module as the trigger.

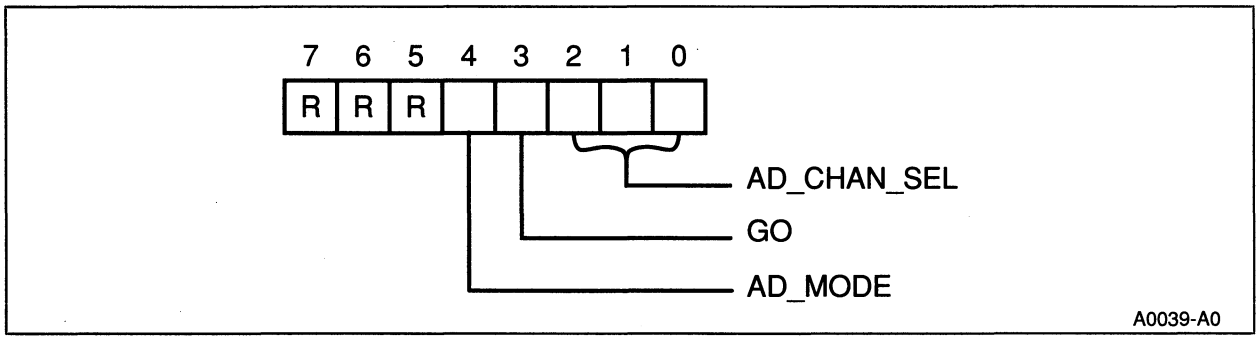


Figure 9-4. The AD_COMMAND Register

Write the channel number to the AD_CHAN_SEL bits (bits 0-2). To select 10-bit conversions, clear the AD_MODE bit (bit 4). To select 8-bit conversions, set the AD_MODE bit. Clear reserved bits 5–7.

Set the GO bit (bit 3) to start the conversion immediately. Conversions initiated by setting the GO bit will start within three state times after the instruction executes.

Clear the GO bit to have the HSO module initiate the conversion at a future time. The HSO module triggers the conversion by executing the command (CMD_TAG = 0FH). The conversion process begins when Timer 1 increments to the programmed value. This aids applications attempting to approach spectrally pure sampling because successive samples spaced by equal Timer 1 delays will occur with a variance of about ± 50 ns (assuming a stable clock on XTAL1).

The following code sample shows how to use the HSO module to trigger the conversion.

```
LDB AD_COMMAND, XXXX1XXXB            ;sets the GO bit
LDB HSO_COMMAND, #START_AD           ;loads the start AD conversion command
                                     ;into the HSO_COMMAND register.
ADD HSO_TIME, TIMER1, #TIME_DELAY    ;adds a 16-bit constant to the current
                                     ;contents of TIMER1, and puts the result
                                     ;in HSO_TIME
```

The HSO register allows multiple START_AD commands to be loaded into the CAM. However, be careful when writing the HSO commands. If Timer 1 increments and the new conversion is started while the previous one is still in progress, the conversion in progress is aborted and the new one is started.

The PTS can begin a new conversion in response to an A/D Conversion Complete interrupt (INT01). Chapter 5, "Interrupts," describes how to use the PTS in the A/D Mode.

9.3. READING THE CONVERSION RESULTS

The AD_RESULT register consists of two bytes (see Figure 9-5). The high byte (03H) contains the eight most-significant bits from the A/D conversion. The low byte (02H) indicates the A/D channel number that was used for the conversion, indicates whether a conversion is currently in progress, and contains the two least-significant bits from a ten-bit A/D conversion. The AD_RESULT register is cleared when a new conversion is started; therefore, to prevent losing data, both bytes must be read before a new conversion starts. See Appendix C for a detailed description of the AD_RESULT register.

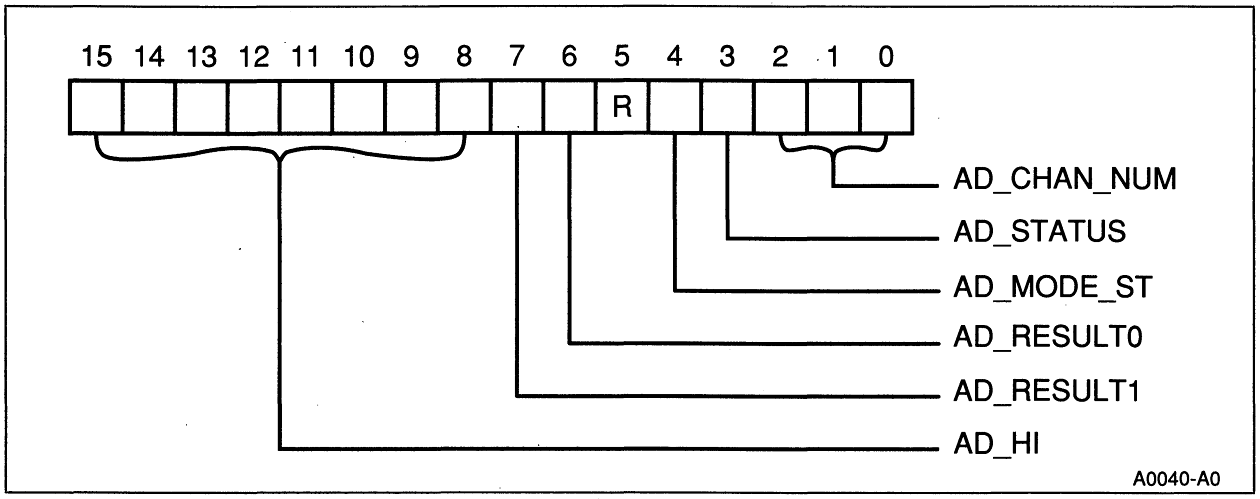


Figure 9-5. The AD_RESULT Register

The A/D converter generates the A/D Conversion Complete interrupt (INT01) when it completes a conversion. This interrupt vectors through location 2002H and is enabled by setting the AD_MASK bit (INT_MASK.1). The AD_PEND bit (INT_PEND.1) is set when a conversion completes, even if the interrupt is masked. This bit stays set until the interrupt vectors to 2002H or the register is cleared. See Chapter 5 for a detailed explanation of interrupts.

Another way to determine the conversion status is to poll the AD_RESULT register. The AD_STATUS bit (AD_RESULT.3) is set while a conversion is in progress. Since it can take up to eight state times to set this bit, do not begin polling until at least eight state times after starting the conversion.

9.4. INTERFACING WITH THE A/D CONVERTER

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

These external factors are idealized in Figure 9-6. The external input circuit must be able to charge a sample capacitor (C_S) through a series input resistance (R_1) to an accurate voltage given a DC input leakage (I_{LI1}). Typically, C_S is about 3.0 pF, R_1 is about 1 KΩ and I_{LI1} is specified as 3 µA maximum.

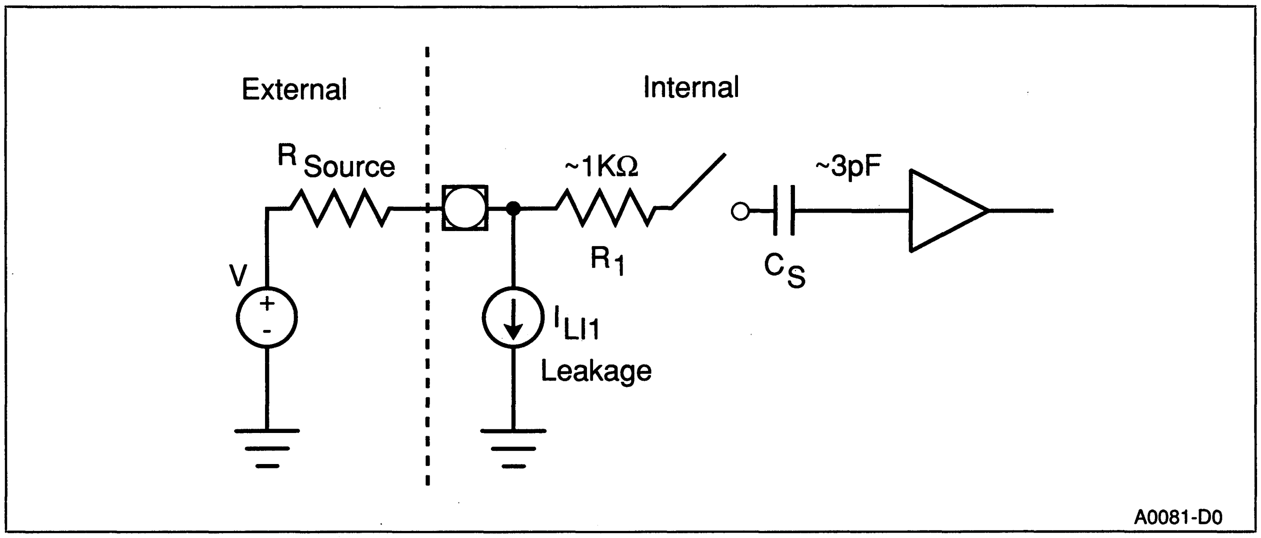


Figure 9-6. Idealized A/D Sampling Circuitry

External circuits with source impedances (R_{SOURCE}) of 1 KΩ or less will be able to maintain an input voltage within a tolerance of about ± 0.6 LSB (1.0 KΩ × 3.0 µA = 3.0 mV) given a DC input leakage (I_{LI1}). Source impedances above 2 KΩ can result in an external error of at least one LSB, due to the voltage drop caused by the 3 µA leakage.

Typically, I_{LI1} leakage is much lower than the maximum specification. Given typical leakage, source impedance may be increased substantially before a one-LSB error is apparent. However, a high source impedance may prevent the internal sample capacitor from fully charging during the sample window. This error can be calculated using the following formula.

$$\text{error (LSBs)} = \left(e^{\frac{-T_{SAM}}{RC}}\right) \times 1024$$

where:

T_{SAM} = sample time, in µseconds
$R = R_{SOURCE} + R_1$, in ohms
$C = C_S$, in µfarads

The effects of this error can be minimized by connecting an external capacitor (C_{EXT}) from the input pin to ANGND. The external signal will charge C_{EXT} to the source voltage. When the channel is sampled, a small portion of the charge stored in C_{EXT} will be transferred to the internal sample capacitor. The ratio of C_S to C_{EXT} causes the loss in accuracy. If C_{EXT} is .005 µF or greater, the maximum error will be –0.6 LSB.

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 (see Figure 9-7). In practice, one should include a small series resistor 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.

The suggested A/D input circuit shown in Figure 9-7 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 DCV. The pin has an absolute maximum low voltage rating of –0.5 DCV; this leaves about 0.3 DCV across the 270Ω resistor, or about 1 mA of current.

NOTE

Driving any analog input more than 0.5 V beyond ANGND or V_{REF} activates the input protection devices. This drives current into the internal reference circuitry and substantially degrades the accuracy of A/D conversions on all channels.

Thoroughly analyze the applicability of the circuit shown in Figure 9-7 before using it in an actual application.

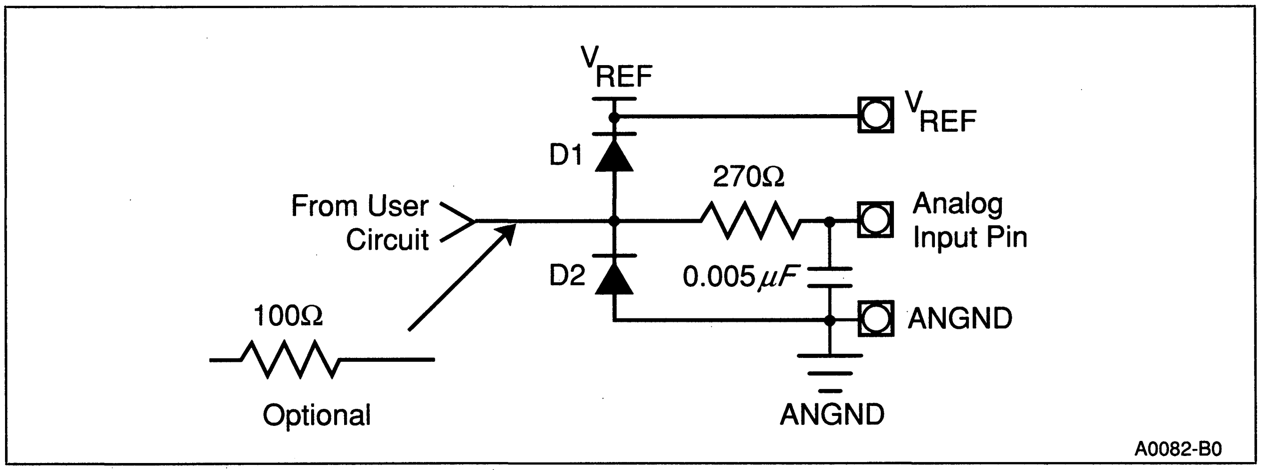


Figure 9-7. Suggested A/D Input Circuit

9.4.1. Analog Ground and Reference

Reference supply levels strongly influence the absolute accuracy of the conversion. For this reason, we recommend that you tie the ANGND pin to the V_{SS} pin as close to the device as possible, using a minimum trace length. In a noisy environment, we highly recommend the use of a separate analog ground plane that connects to V_{SS} at a single point as close to the device as possible. Bypass capacitors should also be used between V_{REF} and ANGND. ANGND should be within about a tenth of a volt of V_{SS}. V_{REF} should be well regulated and used only for the A/D converter. The V_{REF} supply can be between 4.0 and 5.5 V and needs to be able to source approximately 5 mA (see the data sheet for actual specifications). Large negative current spikes on the ANGND pin relative to V_{SS} may cause the analog circuitry to latch-up. This is an additional reason to follow careful grounding practice.

The analog voltage reference (V_{REF}) is the positive supply to which all A/D conversions are compared. It is also the supply to Port 0 if the A/D converter is not being used. If high accuracy is not required, V_{REF} can be tied to V_{CC}. If accuracy is important, V_{REF} must be very stable. One way to accomplish this is through the use of a precision power supply or a separate voltage regulator (usually an IC). These devices must be referenced to ANGND, **not** to V_{SS}, to ensure that V_{REF} tracks ANGND and not V_{SS}.

9.4.2. Using Mixed Analog and Digital Inputs

Port 0 may be used for both analog and digital input signals at the same time. However, when Port 0 is read (by reading location 0EH), some noise may be injected into the analog circuitry. For this reason, make certain that an analog conversion is not in progress when the port is read. Refer to Chapter 6, "I/O Ports" for information about using Port 0 as a digital input.

9.5. TRANSFER FUNCTION AND A/D ERROR SOURCES

The conversion result is an 8- or 10-bit representation of the ratio of the input voltage to the reference voltage. Use the following formula to calculate the 10-bit conversion result:

$$\text{RESULT (10-bit)} = 1023 \times \frac{V_{IN} - \text{ANGND}}{V_{REF} - \text{ANGND}}$$

This produces a stair-stepped *transfer function* when the output code is plotted versus input voltage. The resulting digital codes can be taken as simple ratiometric information, or they provide information about absolute voltages or relative voltage changes on the inputs.

The more demanding the application, 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 requires a detailed understanding of an A/D converter's operation and errors.

In many applications, it is less critical to record the absolute accuracy of an input than it is to detect that a change has occurred. This approach is acceptable as long as the converter is *monotonic* and has *no missing codes*. That is, increasing input voltages produce adjacent, unique output codes that are also increasing. Decreasing input voltages produce adjacent, unique output codes that are also decreasing. In other words, there exists a unique input voltage range for each 10-bit output code that produces that code only, with a repeatability of typically ± .25 LSBs (1.5 mV).

The inherent errors in an analog-to-digital conversion process are quantizing error, zero offset error, full-scale error, differential non-linearity, and non-linearity. All of these are *transfer function* errors related to the A/D converter. In addition, temperature coefficients, V_{CC} rejection, sample-hold feedthrough, multiplexer off-isolation, channel-to-channel matching, and random noise should be considered. Fortunately, one absolute error specification 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.

An unavoidable error 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. The transfer function for an ideal 3-bit A/D converter is shown as the Ideal Characteristic (see Figure 9-8).

Note that the Ideal Characteristic possesses unique qualities:

- its first code transition occurs when the input voltage is 0.5 LSB;
- its full-scale code transition occurs when the input voltage equals the full-scale reference minus 1.5 LSB; and
- its code widths are all exactly one LSB.

These qualities result in a digitization without zero offset, full-scale, or linearity errors. In other words, a perfect conversion.

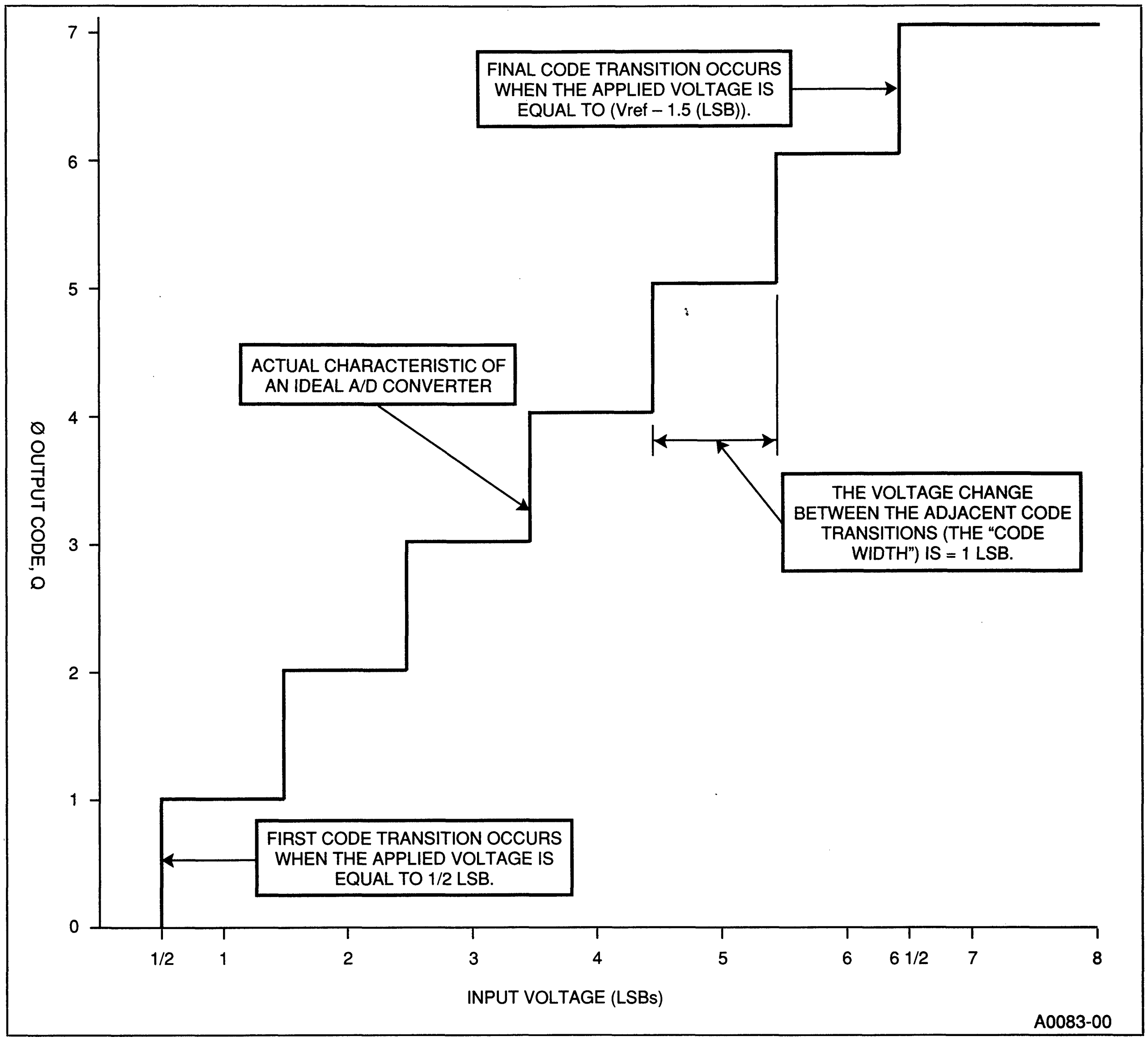


Figure 9-8. Ideal A/D Conversion Characteristics

The Actual Characteristic of a hypothetical 3-bit converter is not perfect. When the Ideal Characteristic is overlaid with the actual characteristic, the actual converter is seen to exhibit errors in the locations of the first and final code transitions and in code widths, as shown in Figure 9-9. The deviation of the first code transition from ideal is called *zero offset* error, and the deviation of the final code transition from ideal is *full-scale* error. The deviation of a code width from ideal causes two types of errors: Differential Non-Linearity and Non-Linearity. *Differential Non-Linearity* is a measure of local code-width error, whereas *Non-Linearity* is a measure of overall code-transition error.

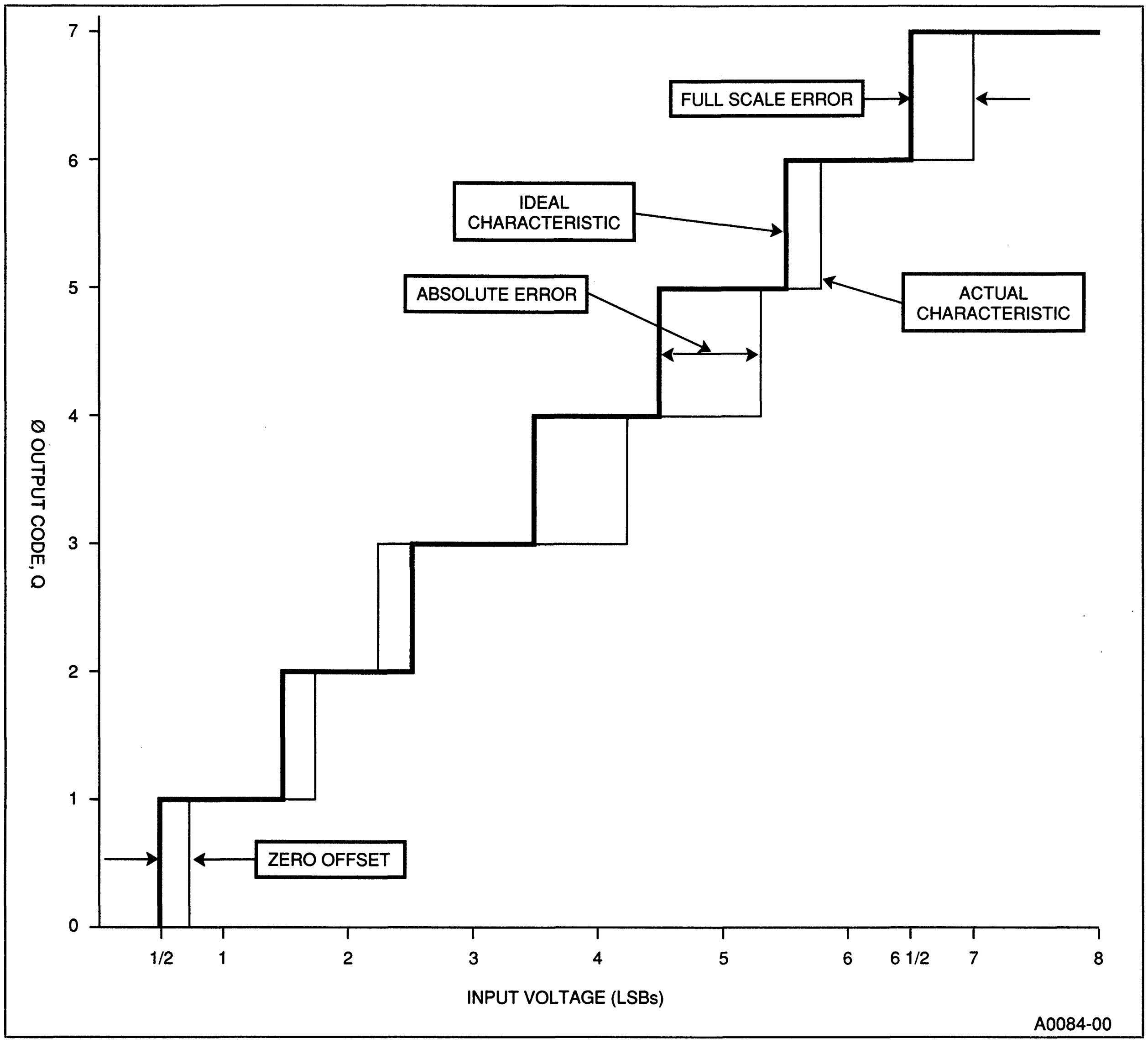


Figure 9-9. Actual and Ideal A/D Conversion Characteristics

Differential Non-Linearity is the degree to which actual *code widths* differ from the ideal one-LSB width. It 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. In the 10-bit converter, the code widths are ideally 5 mV (5.12 V_{REF} / 1024). If such a converter is specified to have a maximum Differential Non-Linearity of 2 LSBs (10 mV), then the maximum code width will be no greater than 10 mV larger than ideal, or 15 mV.

Because the 8XC196KC/KD converter has *no missing codes*, the minimum code width will always be greater than –1 (negative one). The Differential Non-Linearity error on a particular code width is compensated for by other code widths in the transfer function, such that 1024 unique steps occur. The actual code widths in this converter typically vary from 2.5 mV to 7.5 mV.

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 to exhibit non-monotonic behavior. Neither behavior is desirable in a closed-loop system. A converter has *no missing 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.

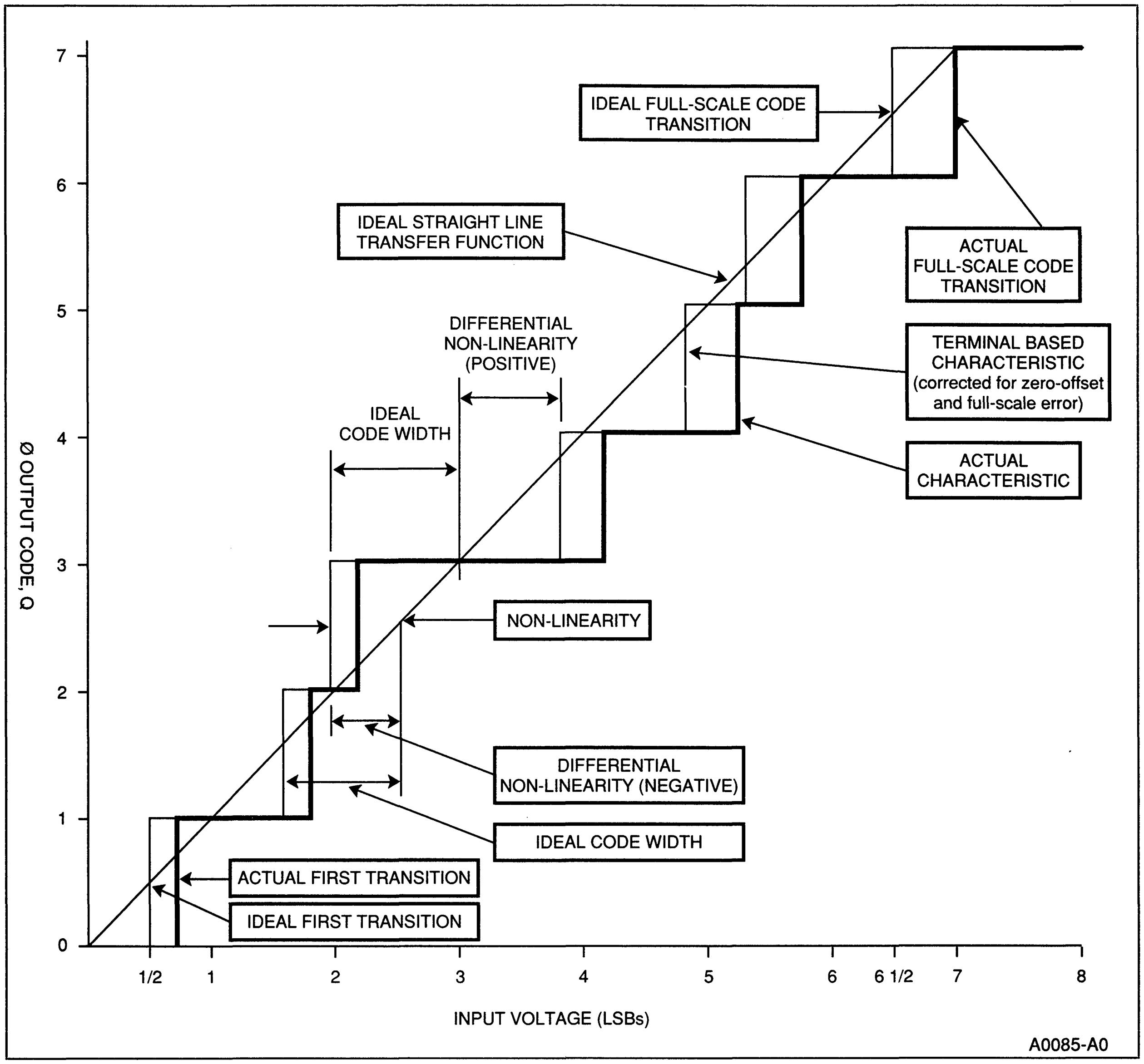
Differential Non-Linearity and Non-Linearity are quantified by measuring the Terminal-Based Linearity errors. A Terminal-Based Characteristic results when an Actual Characteristic is translated and scaled to eliminate zero offset and full-scale error, as shown in Figure 9-10. The Terminal-Based Characteristic 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 that include gain and offset trimming. In addition, V_{REF} on the 80C196KC/KD 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 temperature drift, failure to completely reject unwanted signals, multiplexer channel dissimilarities, and random noise. Fortunately, these effects are small.

Temperature drift is the rate at which typical specifications change with a change in temperature. These changes are reflected in the *temperature coefficients*.

Unwanted signals come from three main sources: noise on V_{CC}, input signal changes on the channel being converted (after the sample window has closed), and signals applied to channels not selected by the multiplexer.

Finally, multiplexer on-channel resistances differ slightly from one channel to the next, which causes *channel-to-channel matching* errors and *repeatability* errors; differences in DC leakage current from one channel to another and random noise in general contribute to repeatability errors.



IDEAL FULL-SCALE CODE TRANSITION
IDEAL STRAIGHT LINE TRANSFER FUNCTION
ACTUAL FULL-SCALE CODE TRANSITION
DIFFERENTIAL NON-LINEARITY (POSITIVE)
TERMINAL BASED CHARACTERISTIC (corrected for zero-offset and full-scale error)
IDEAL CODE WIDTH
ACTUAL CHARACTERISTIC
NON-LINEARITY
DIFFERENTIAL NON-LINEARITY (NEGATIVE)
IDEAL CODE WIDTH
ACTUAL FIRST TRANSITION
IDEAL FIRST TRANSITION
Ø OUTPUT CODE, Q
0
1
2
3
4
5
6
7
1/2
1
2
3
4
5
6
6 1/2
7
8
INPUT VOLTAGE (LSBs)
A0085-A0

Figure 9-10. Terminal-Based A/D Conversion Characteristics

Pulse Width Modulator 10

CHAPTER 10
PULSE WIDTH MODULATOR

The 8XC196KC/KD has three Pulse Width Modulator (PWM) modules (see Figure 10-1). Each outputs a variable duty cycle pulse at a fixed frequency. These outputs may be used to drive motors that require an unfiltered PWM waveform for optimal efficiency, or they may be filtered to produce a smooth analog signal.

This chapter provides a functional overview of the Pulse Width Modulator modules, describes how to program them, and provides sample circuitry for converting the PWM outputs to analog signals.

Refer to Appendix C for details about the registers discussed in this chapter and to Appendix B for information about the PWM*x* output pins.

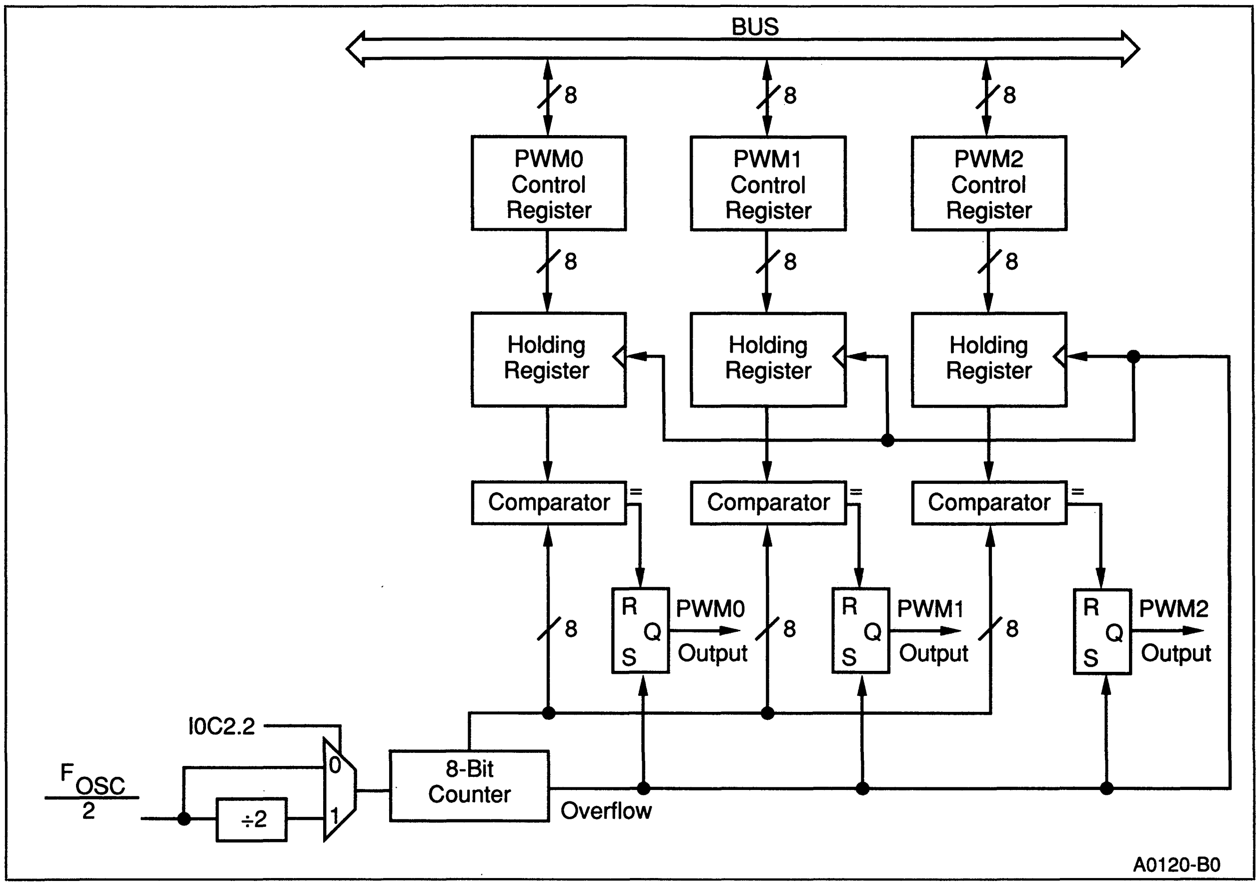


Figure 10-1. PWM Block Diagram

10.1. PWM FUNCTIONAL OVERVIEW

The PWM modules have the following main components:

- An eight-bit counter
- A divide-by-two clock prescaler
- Three comparators
- Three control registers (PWM*x*_CONTROL, where *x* is 0, 1, or 2)
- Three holding registers
- Three RS flip-flops

The SLOW_PWM bit (IOC2.2) controls the PWM output period by enabling or disabling the divide-by-two clock prescaler. Enabling the prescaler causes the 8-bit counter to increment once every two state times; disabling it causes the counter to increment once every state time.

Each control register contains an 8-bit value that is loaded into a holding register when the 8-bit counter overflows. The comparators compare the contents of the holding registers to the counter value. When the counter value is equal to zero, the PWM*x* output is driven high. It remains high until the counter value matches the value in the holding register, at which time the output is pulled low. (Loading PWM*x*_CONTROL with 00H forces the output to remain low.) When the counter overflows, the output is again switched high. Figure 10-2 shows typical PWM output waveforms.

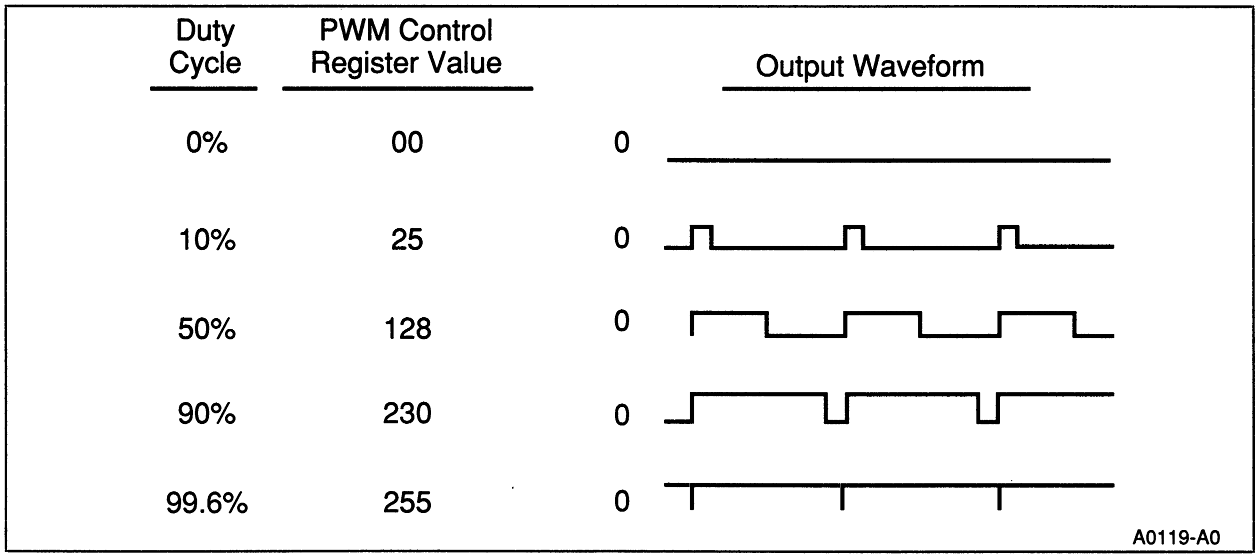


Figure 10-2. PWM Output Waveforms

10.2. PROGRAMMING THE PWM DUTY CYCLE

The PWM*x*_CONTROL register, in conjunction with the SLOW_PWM bit (IOC2.2), determines how long the PWM*x* output is held high during the pulse, effectively controlling the duty cycle. The value written to PWM*x*_CONTROL register can be from 0 to 255 state times (0% to 99.6% duty cycle). Setting IOC2.2 enables the PWM's divide-by-two clock prescaler; the total length of the pulse is 512 state times and the PWM*x*_CONTROL value is multiplied by 4. Clearing IOC2.2 disables the prescaler; the total pulse length is 256 state times and the PWM*x*_CONTROL value is multiplied by 2. Table 10-1 lists sample output frequencies.

Table 10-1. PWM Output Frequencies

	XTAL1 Frequency (F_{OSC})			
IOC2.2	**8 MHz**	**10 MHz**	**16 MHz**	**20 MHz**
0	15.6 KHz	19.6 KHz	31.25 KHz	39.1 KHz
1	7.8 KHz	9.8 KHz	15.63 KHz	19.5 KHz

Use the following formulas to calculate the PWM period and the time that the PWM output is high. Note that PWM*x*_CONTROL is an 8-bit value and F_{OSC} is the XTAL1 frequency, in MHz.

	Clock Prescaler Disabled (IOC2.2=0)	**Clock Prescaler Enabled (IOC2.2=1)**
PWM Period (in µsec.) =	$\frac{512}{F_{OSC}}$	$\frac{1024}{F_{OSC}}$
PWM*x* High (in µsec.) =	$\frac{\text{PWMx\_CONTROL} \times 2}{F_{OSC}}$	$\frac{\text{PWMx\_CONTROL} \times 4}{F_{OSC}}$

For example, if F_{OSC} equals 16 MHz, then the period of the PWM output waveform is 32 µs. If PWM0_CONTROL equals 8AH (138 decimal) and IOC2.2 is clear, PWM0 is held high for 17.25 µs (and low for 14.8 µs) of the total 32 µs, resulting in a duty cycle of approximately 54%. When IOC2.2 is set, the same values would produce a period of 64 µs and PWM0 would be held high for 34.5 µs (and low for 29.5 µs), for the same duty cycle, approximately 54%.

10.3. ENABLING THE PWM OUTPUTS

Each PWM output is multiplexed with a port pin. Table 10-2 shows the alternate port function along with the register setting that selects the PWM output instead of the port function. Selecting the PWM1 or PWM2 output function disables the quasi-bidirectional Port 1 function and enables strong pull-ups and pull-downs.

Table 10-2. PWM Output Alternate Functions

PWM Output	Alternate Port Function	PWM Output Enabled When:
PWM0	P2.5	IOC1.0 = 1
PWM1	P1.3	IOC3.2 = 1
PWM2	P1.4	IOC3.3 = 1

10.4. GENERATING ANALOG OUTPUTS

Both the PWM modules and the High-Speed Output (HSO) module can generate a rectangular pulse train that varies in duty cycle and period. Filtering this output will create a smooth analog signal. To make a signal swing over the desired analog range, first buffer the signal and then filter it. Figure 10-3 is a block diagram of the type of circuit needed to create the smooth analog signal. Use either a simple RC network or an active filter.

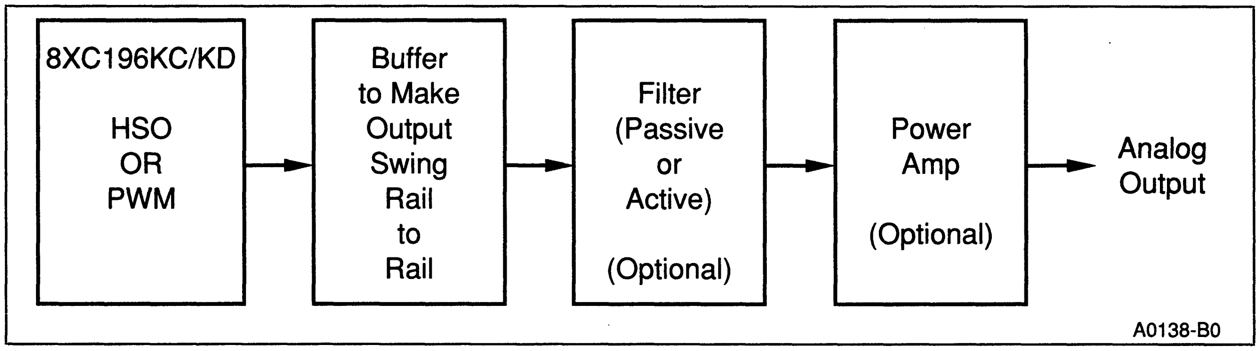

8XC196KC/KD
HSO OR PWM
Buffer to Make Output Swing Rail to Rail
Filter (Passive or Active) (Optional)
Power Amp (Optional)
Analog Output
A0138-B0

Figure 10-3. D/A Buffer Block Diagram

Figure 10-4 shows a sample circuit used for low output currents (less than 100 µA). Consider temperature and power-supply drift when selecting components for the external D/A circuitry. With proper components, a highly accurate 8-bit D/A converter can be made using either the PWM or HSO outputs. The HSO could theoretically extend the accuracy to sixteen bits, but temperature and noise problems would be difficult to control.

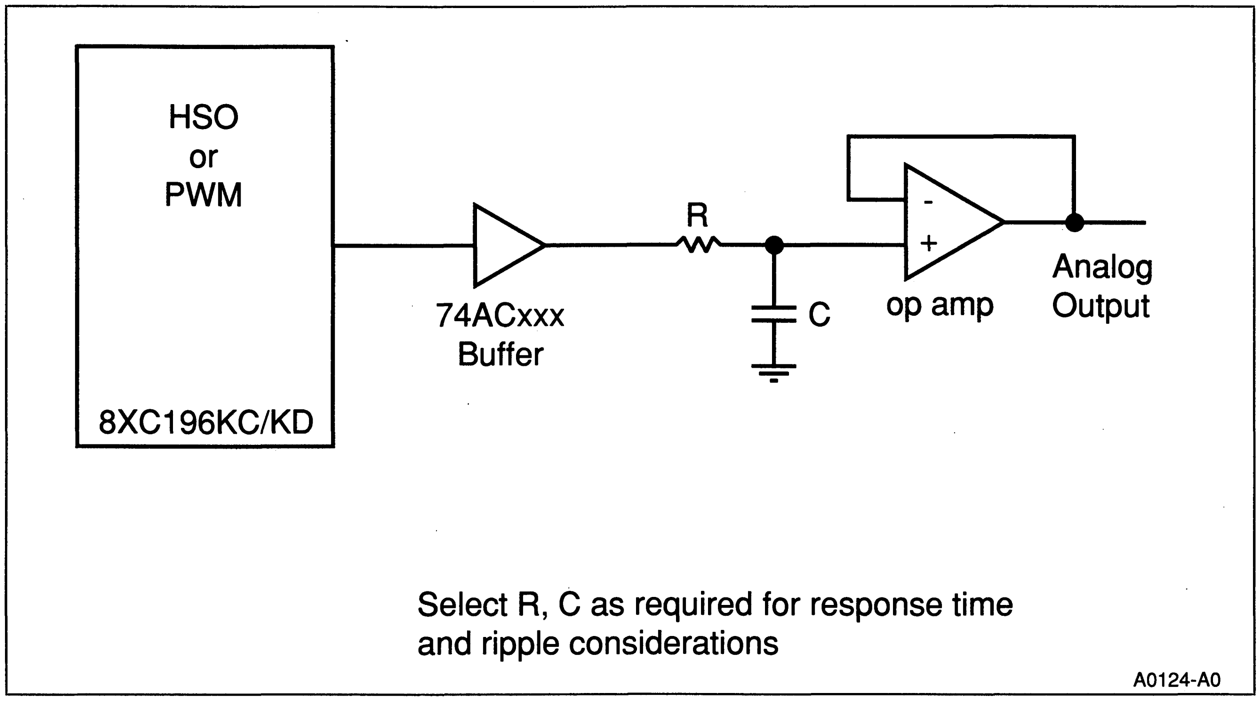

HSO
or
PWM
8XC196KC/KD
74ACxxx
Buffer
R
C
-
+
op amp
Analog
Output
Select R, C as required for response time and ripple considerations
A0124-A0

Figure 10-4. PWM to Analog Conversion Circuitry

Minimum Hardware Considerations

11

CHAPTER 11
MINIMUM HARDWARE CONSIDERATIONS

The 8XC196KC/KD has several basic requirements for operation within a system. It requires an external source of power, ground, the clock signal, and the RESET# signal. This chapter describes options for providing the basic requirements and describes other hardware considerations.

11.1. MINIMUM CONNECTIONS

Figure 11-1 shows a minimum hardware configuration. For predictable performance, it is important to tie unused inputs to V_{CC} or V_{SS} as shown in the sample circuit. Otherwise, unused inputs can float to a mid-voltage level and draw excessive current. External interrupt signals, such as NMI, may generate spurious interrupts if left unconnected.

NOTE

The RC network connected to V_{PP} is required only when the Powerdown feature is being used. Refer to Chapter 12, "Special Operating Modes," for information about Powerdown operation.

If HSI.2 and HSI.3 are configured as outputs, do not tie them to ground as shown in Figure 14-1.

11.1.1. Port Connections

The Port 0 pins also function as analog inputs into the A/D converter and they may be allowed to float. However, we do recommend that you connect unused Port 0 pins to V_{CC} or V_{SS}.

The quasi-bidirectional port pins (P1.0–P1.7, P2.6, and P2.7) have weak internal pull-ups. These pins may be allowed to float.

In the minimum configuration, the system address/data bus (AD0–AD15) functions as Ports 3 and 4 with a default value of 0FFFFH. Since the Port 3 and 4 outputs are open-drain outputs, they will float unless they are connected to external circuitry. To prevent potential problems, either tie the pins to ground or write zeros to Ports 3 and 4.

11.2. POWER AND GROUND PINS

Power to the 8XC196KC/KD flows through several pins. V_{CC} supplies the positive voltage to the digital portion of the device, while V_{REF} supplies the positive voltage to the A/D converter and Port 0. Connect V_{CC} and V_{REF} to +5V power supplies. If the A/D converter will be used, connect V_{REF} to a separate reference supply to minimize noise during A/D conversions.

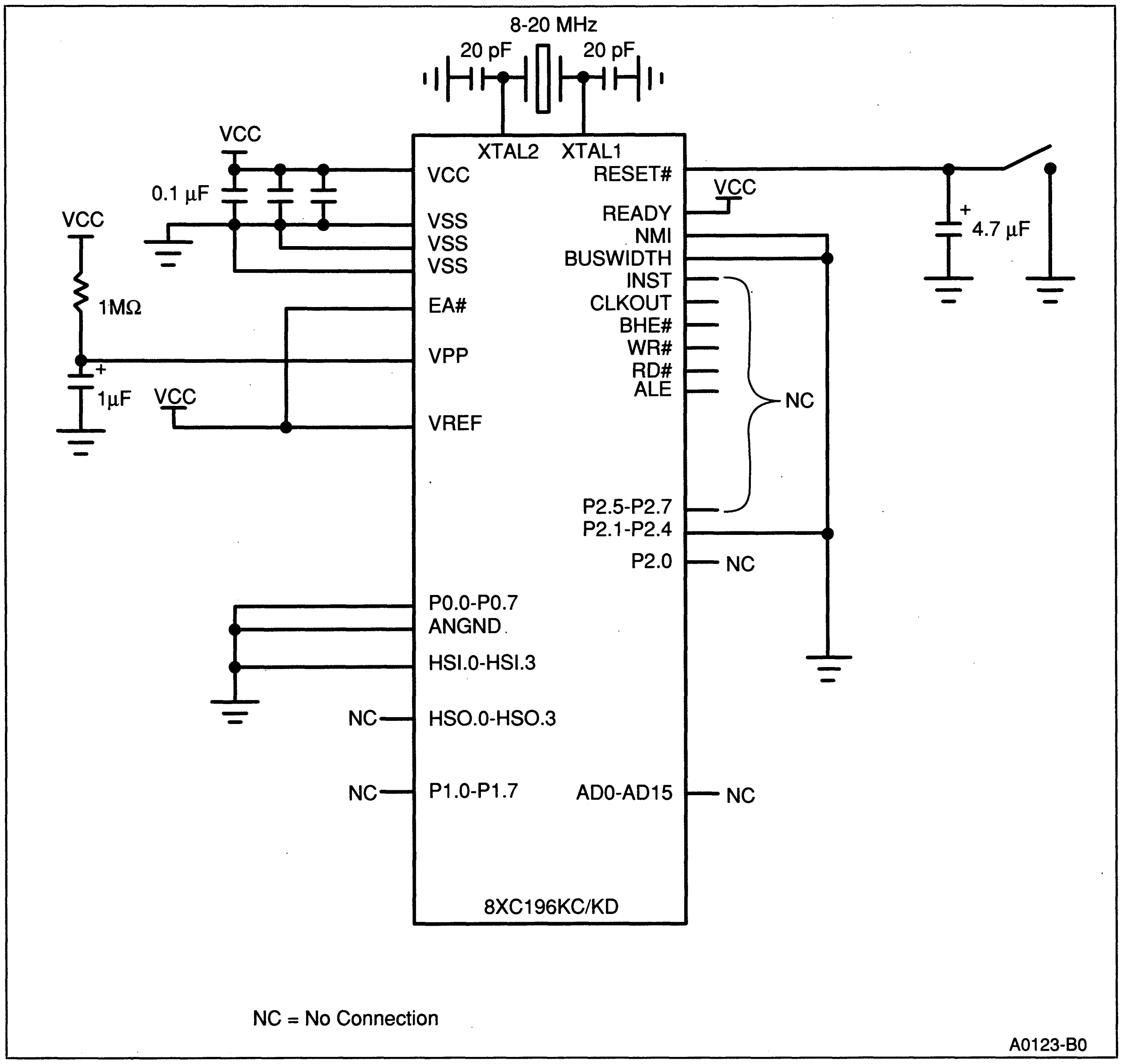


Figure 11-1. 8XC196KC/KD Minimum Hardware Connections

The V_{SS} pins and ANGND must all be nominally at the same potential (± 50 mV). For best A/D operation, connect ANGND and V_{SS} to a single point as close to the device as possible. Use the shortest possible path to connect the V_{SS} lines to ground. Add decoupling capacitors between each V_{SS} pin and V_{CC}.

Figure 11-2 shows connections for V_{CC}, V_{REF}, V_{SS}, and ANGND

NOTE

Even if the A/D converter will not be used, V_{REF} and ANGND must be connected to provide power to Port 0. Refer to Chapter 9, “Analog-to-Digital Converter,” for a detailed discussion of A/D power and ground recommendations.

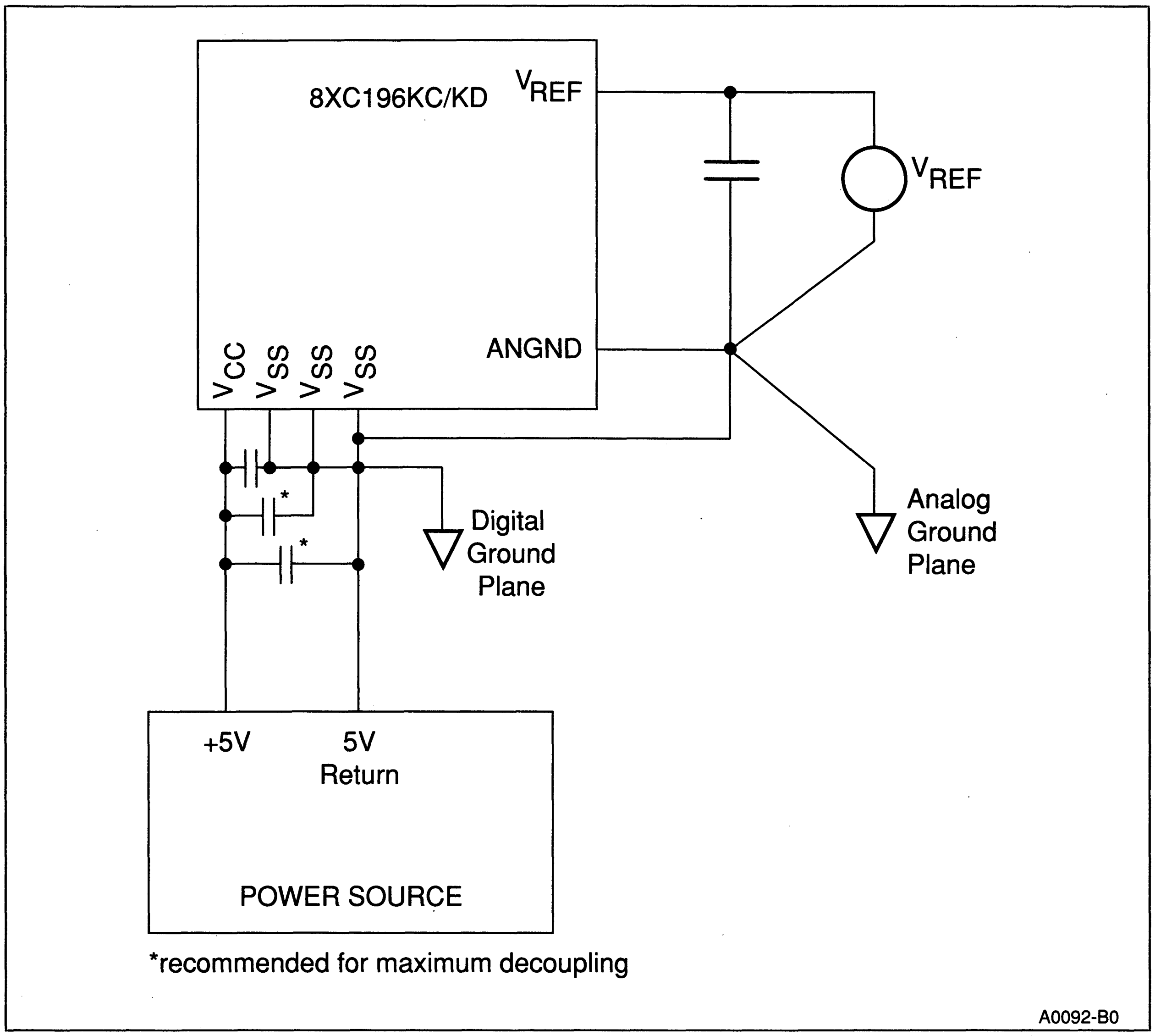


Figure 11-2. Power and Return Connections

11.2.1. Noise Protection Tips

The fast rise and fall times of high-speed CMOS logic often produces noise spikes on the power supply lines and outputs. To minimize noise, it is important to follow good design and board layout techniques. We recommend liberal use of decoupling capacitors and transient absorbers. Connect 0.01 µF bypass capacitors between V_{CC} and each V_{SS} pin and between V_{REF} and ANGND to reduce noise. Place the capacitors as close to the device as possible.

Multilayer printed circuit boards with separate V_{CC} and ground planes also help to minimize noise. For more information on noise protection, refer to Application Note AP-125, "Designing Microcontroller Systems for Noisy Environments."

11.3. CLOCK SOURCES

The 8XC196KC/KD can either generate the clock signal on-chip or use an external clock input signal. To use the on-chip oscillator, connect an external crystal to XTAL1 and XTAL2. Otherwise, connect an external clock input signal to XTAL1 and let XTAL2 float.

11.3.1. On-Chip Oscillator

The on-chip oscillator circuit consists of a crystal-controlled, positive reactance oscillator (see Figure 11-3). In this application, the crystal operates in a parallel resonance mode.

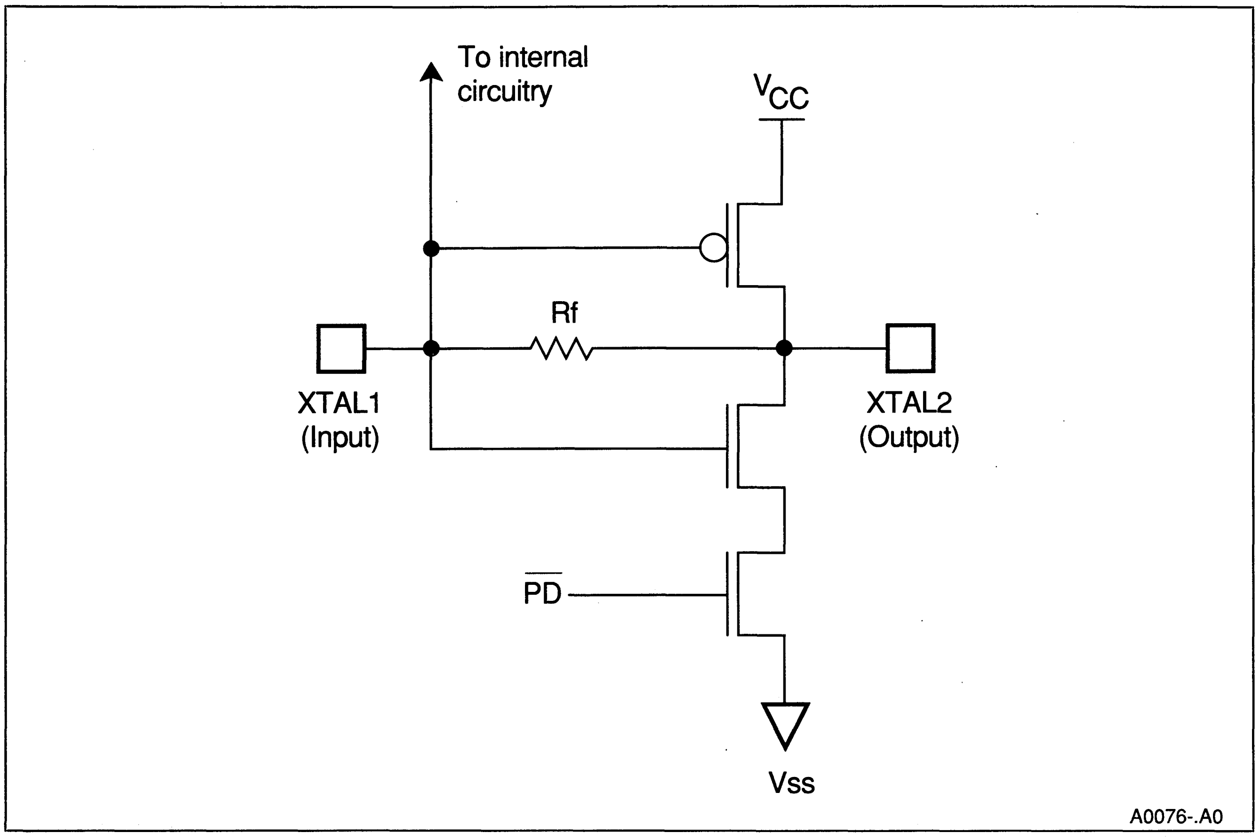


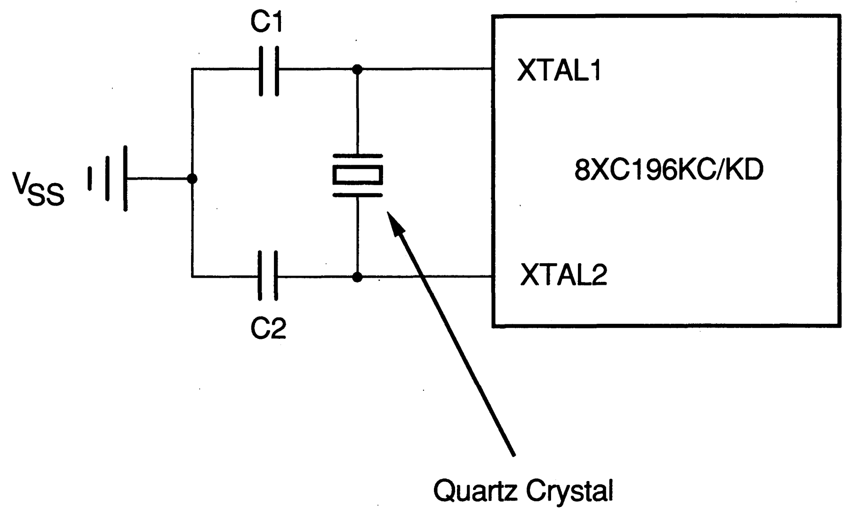
Figure 11-3. On-Chip Oscillator Circuit

The feedback resistor, Rf, consists of paralleled *n*-channel and *p*-channel FETs controlled by the internal powerdown signal (PD). In Powerdown mode, Rf acts as an open and the output drivers are disabled, which disables the oscillator. Both XTAL1 and XTAL2 have built-in electrostatic discharge (ESD) protection.

11.3.1.1. USING CRYSTAL OSCILLATORS

Figure 11-4 shows the connections between the external crystal and the 8XC196KC/KD. Consult the manufacturer's data sheet for performance specifications and required capacitor values. When designing an external oscillator circuit, consider the effects of parasitic board capacitance, extended operating temperatures, and crystal specifications. With quality components, 20 pF load capacitors (C_L) are usually adequate for any frequency above 1 MHz.

Noise spikes on the XTAL1 or XTAL2 pin can cause a miscount in the internal clock-generating circuitry. Capacitive coupling between the crystal oscillator and traces carrying fast-rising digital signals can introduce noise spikes. To reduce this coupling, mount the crystal oscillator and capacitors near the device and use short, direct traces to connect to XTAL1, XTAL2, and V_{SS}. To further reduce the effects of noise, use grounded guard rings around the oscillator circuitry and ground the metallic crystal case.



NOTE:
Keep oscillator components close to chip and use short, direct traces to XTAL1, XTAL2 and Vss. When using crystals, C1=C2≈20pF. When using ceramic resonators, consult manufacturer for recommended oscillator circuitry.

A0077-C0

Figure 11-4. External Crystal Connections

11.3.1.2. USING CERAMIC RESONATORS

In cost-sensitive applications, you may choose to use a ceramic resonator instead of a crystal oscillator. Ceramic resonators may require slightly different load capacitor values and circuit configurations. Consult the manufacturer's data sheet for the required oscillator circuitry.

11.3.2. Using an External Clock Signal

To use an external clock source, apply a clock signal to XTAL1 and let XTAL2 float. An example of this circuit is shown in Figure 11-5. See the data sheet for the required XTAL1 voltage drive levels.

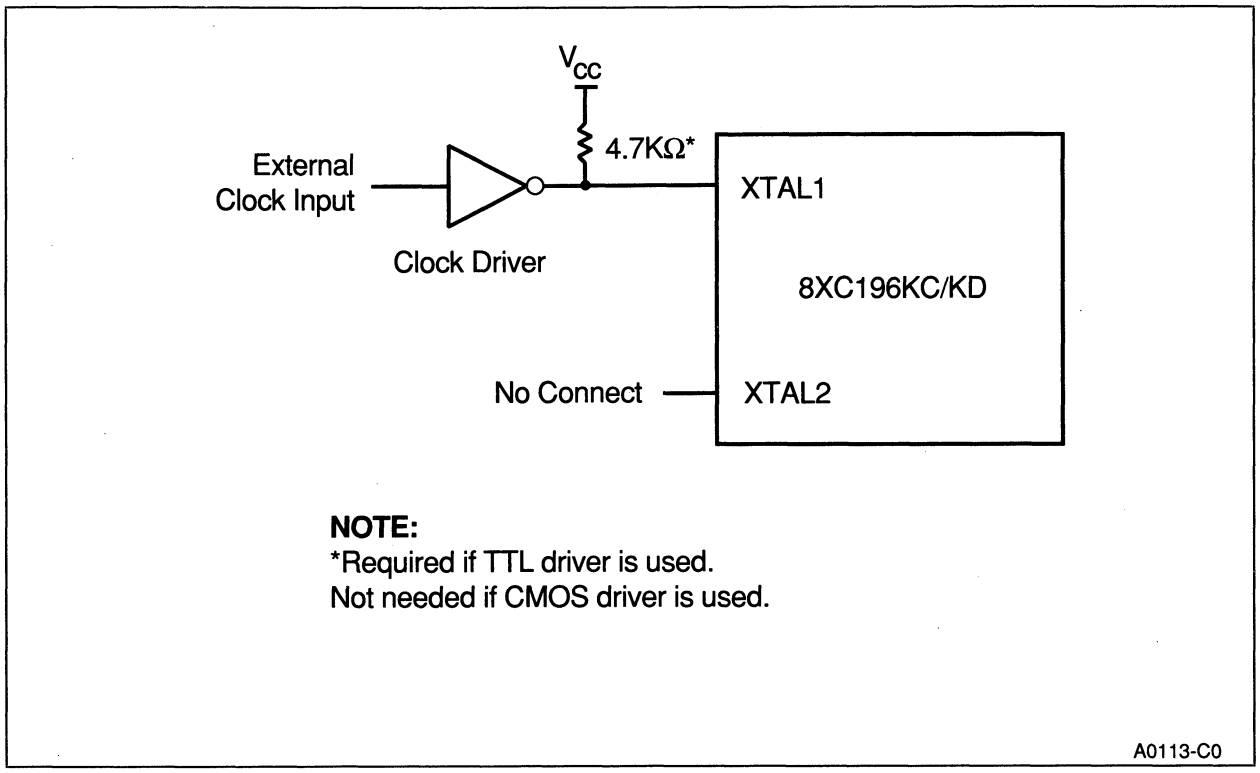


Figure 11-5. External Clock Connections

The external clock source must meet the minimum high and low times (T_{XHXX} and T_{XLXX}) and the maximum rise and fall transition times (T_{XLHX} and T_{XHXL}) to ensure proper operation (see Figure 11-6). The longer the rise and fall times, the higher the probability that external noise will affect the clock generator circuitry and cause unreliable operation. See the data sheet for actual specifications.

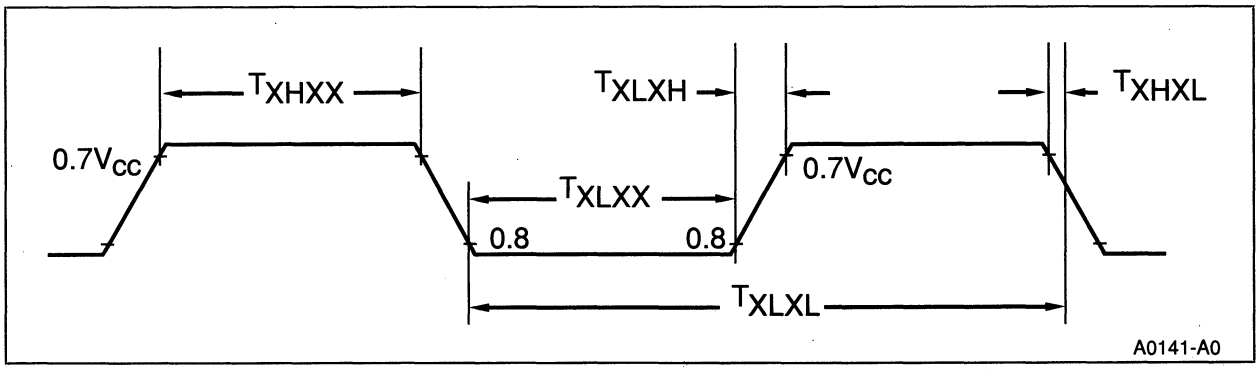


Figure 11-6. External Clock Drive Waveforms

At power-on, the interaction between the internal amplifier and its feedback capacitance (i.e., the Miller effect) may cause a load of up to 100 pF at the XTAL1 pin, if the signal at XTAL1 is small, such as might be the case during start-up of the external oscillator. This problem will go away when the XTAL1 input signal meets the V_{IL} and V_{IH1} specifications (see the data sheet). If these specifications are met, the XTAL1 pin capacitance will not exceed 20 pF.

11.4. RESETTING THE 8XC196KC/KD

Reset forces the 8XC196KC/KD into a known state. (See Table B-3 in Appendix B and Table C-3 in Appendix C for pin and register status after reset.) The device remains in its reset state until RESET# is deasserted. When RESET# is deasserted, the bus controller fetches the Chip Configuration Byte (CCB) from location 2018H, loads it into the Chip Configuration Register (CCR), and then fetches the first instruction at location 2080H. Figure 11-7 shows the reset-sequence timing. Depending upon when RESET# is brought high, the CLKOUT signal may become out of phase with the PH1 internal clock. When this occurs, the clock generator immediately resynchronizes CLKOUT as shown in Case 2.

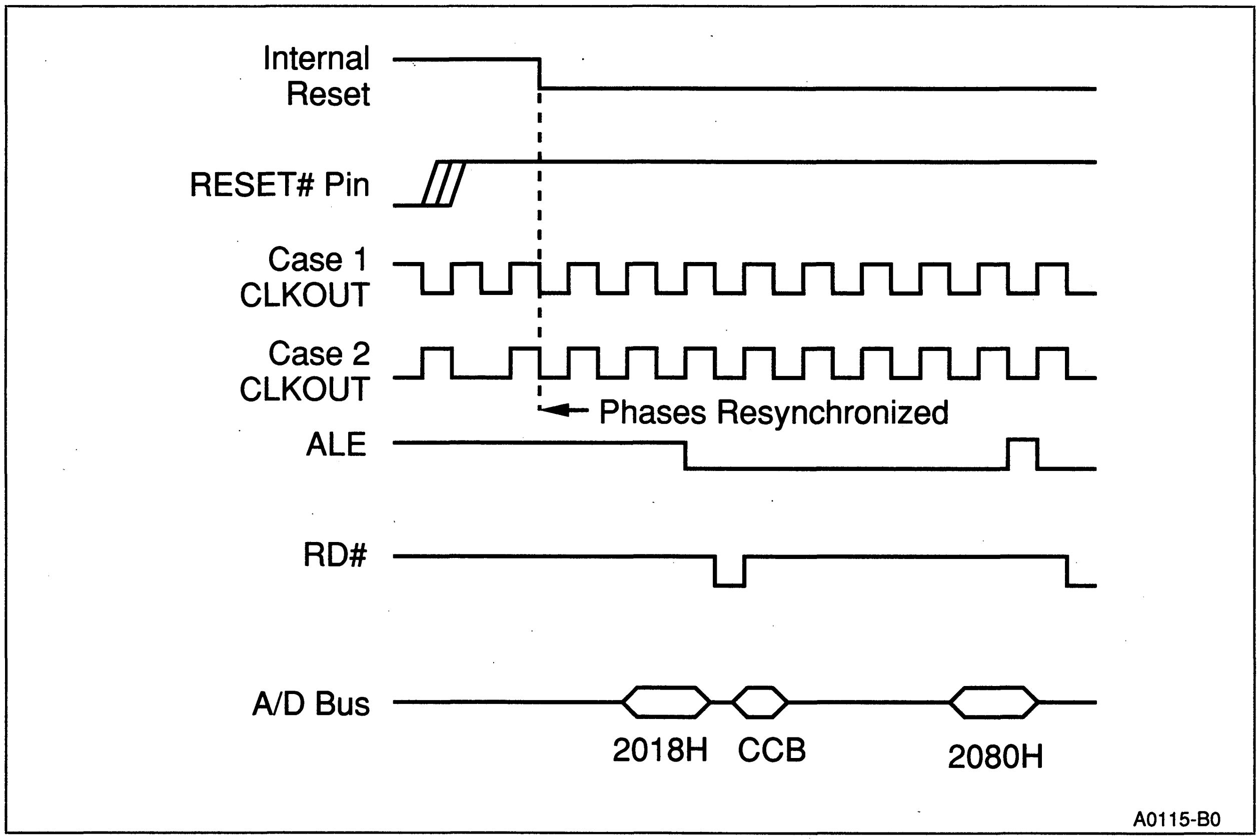


Figure 11-7. Reset Timing Sequence

The following events will reset the 8XC196KC/KD (see Figure 11-8):

- An external device drives the RESET# pin low
- The CPU issues the Reset (RST) instruction
- The CPU issues an IDLPD instruction with an illegal key operand
- The Watchdog timer (WDT) overflows

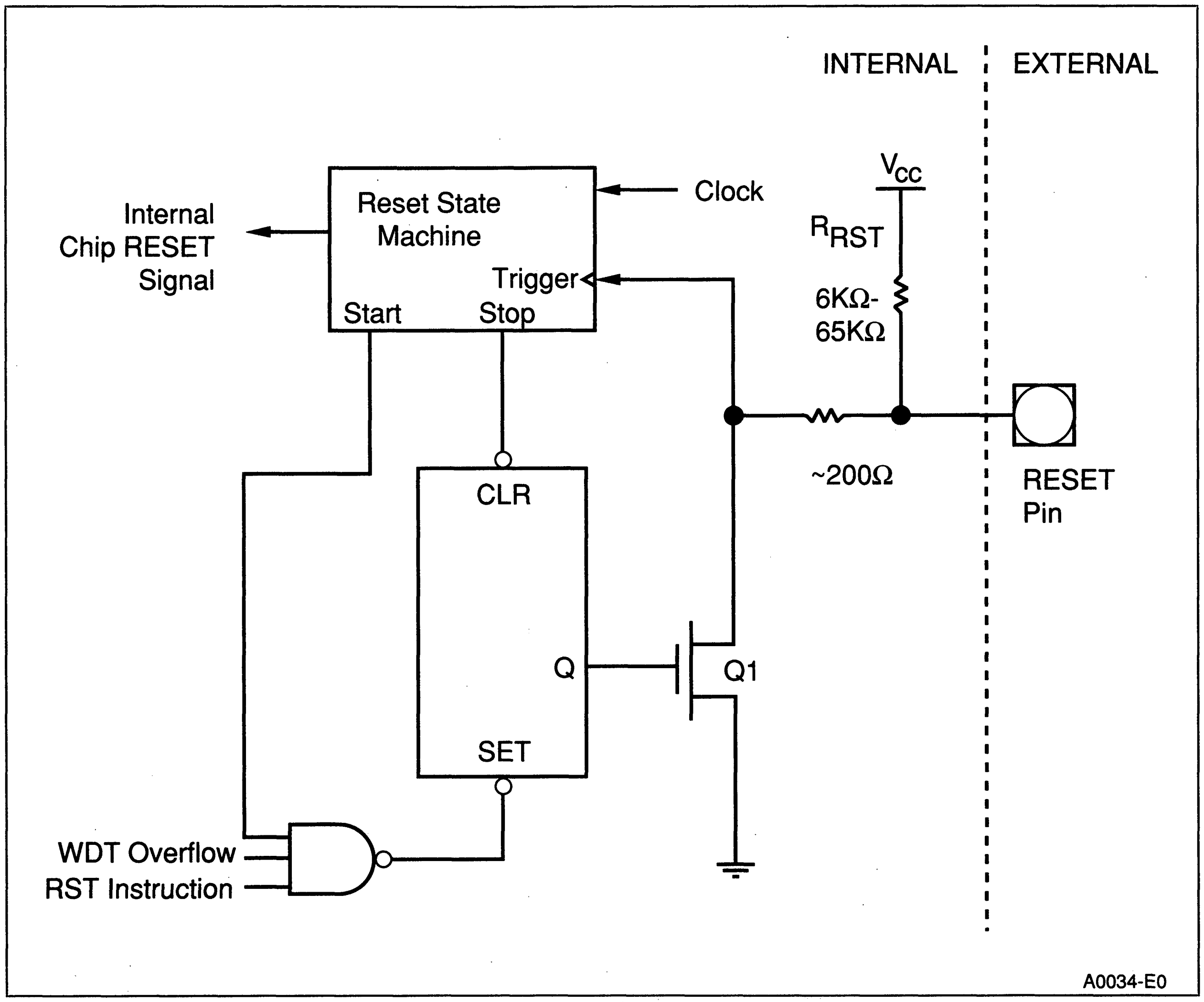


Figure 11-8. Reset Pin Internal Circuit

11.4.1. Pulling the RESET# Pin Low

To reset the device, hold the RESET# pin low for at least one state time after the power supply is within tolerance and the oscillator has stabilized. When RESET# is first asserted, the device turns on a pull-down transistor (Q1) for 16 state times. This enables the RESET# signal to function as the system reset.

11.4.1.1. EXTERNAL RESET CIRCUITS

The simplest way to reset an 8XC196KC/KD is to insert a capacitor between the RESET# pin and V_{SS}, as shown in Figure 11-9. The 8XC196KC/KD has an internal pull-up which has a value between 6 KΩ and 65 KΩ (R_{RST}). RESET# should remain asserted for at least one state time after V_{CC} and XTAL1 have stabilized and meet the operating conditions specified in the data sheet. A 4.7 µF or greater capacitor should provide sufficient reset time, as long as V_{CC} rises quickly.

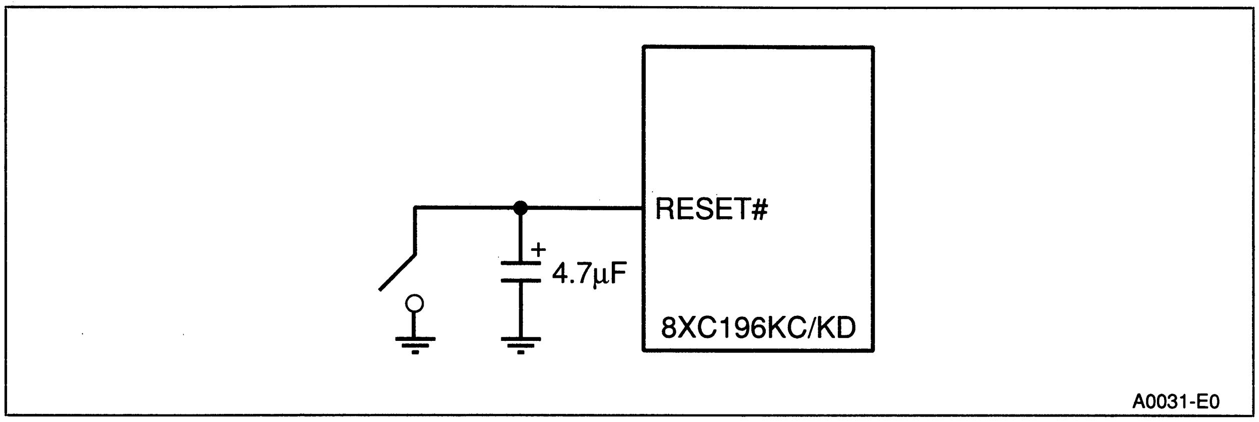


Figure 11-9. Minimum Reset Circuit

This minimum circuit is not adequate when the RESET# pin provides the system reset. The other devices may not be reset because the capacitor will keep the voltage above V_{IL},. Since RESET# is asserted for only 16 state times, it may be necessary to lengthen and buffer the system-reset pulse. Figure 11-10 shows an example of a system-reset circuit. In this example, D2 creates a wired-OR gate connection to the reset pin. An internal 8XC196KC/KD reset, system power up, or the closing of SW1 will generate the system-reset signal.

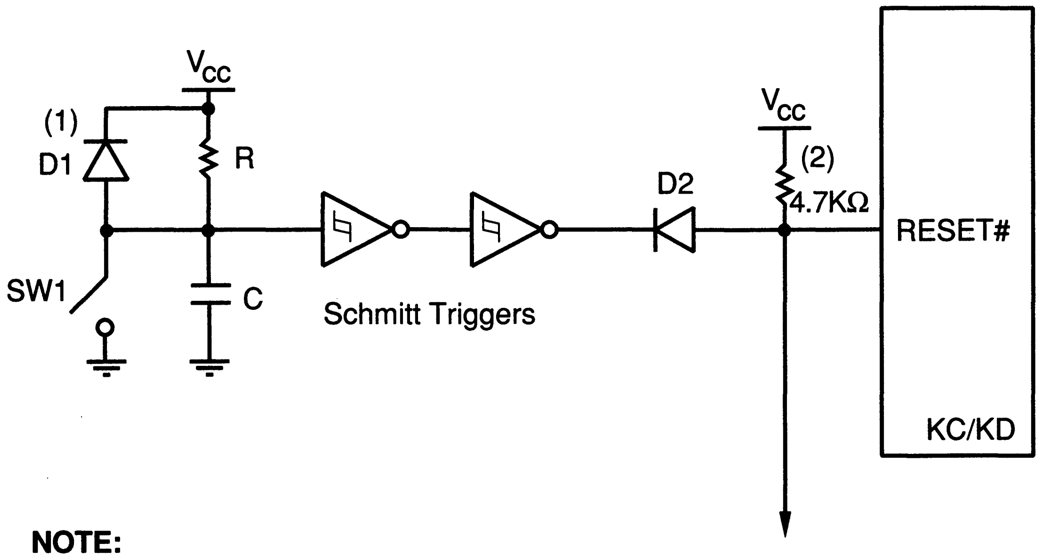
11.4.2. Issuing the Reset (RST) Instruction

The RST instruction (opcode FFH) resets the 8XC196KC/KD by pulling RESET# low for 16 state times. It also clears the program status word (PSW), sets the master program counter (PC) to 2080H, and resets the Special Function Registers (SFRs). See Table C-3 in Appendix C for the reset values of the SFRs.

Putting pull-ups on the address/data bus causes unimplemented areas of memory to be read as 0FFH. If unused internal OTPROM memory is set to 0FFH, then execution from any unused memory locations will reset the 8XC196KC/KD.

11.4.3. Issuing an Illegal IDLPD Key Operand

The 8XC196KC/KD resets itself if an illegal key operand is used with the IDLE/POWERDOWN (IDLPD) command. The legal keys are "1" for Idle mode and "2" for Powerdown mode. If any other value is used, the device executes a reset sequence. (See Appendix A for a description of the IDLPD command.)



NOTE:
1. D1 provides a faster cycle time for repetitive power-on resets.
2. Optional pull-up for faster recovery.

Figure 11-10. Example System Reset Circuit

11.4.4. Enabling the Watchdog Timer

The Watchdog timer (WDT) is a 16-bit ripple counter that resets the device when the counter overflows. When enabled, the Watchdog timer (WDT) monitors the execution of the user program. If the counter is not periodically cleared within 64K state times, it will reset the device. This prevents a runaway process from hanging up the system.

To clear the WATCHDOG register, send a "1EH" followed immediately by an "E1H" to location 0AH. Clearing this register the first time enables the WDT with an initial value of 0000H, which is incremented once every state time. If the counter overflows to 0000H, it drives the RESET# signal low and holds it low for 16 state times (see Figure 11-8). Once enabled, only a reset can disable the WDT.

If enabled, the WDT continues to run in Idle mode. The user program must service the WDT before it times out or it will reset the device, which causes it to exit Idle mode.

Special Operating Modes

12

CHAPTER 12
SPECIAL OPERATING MODES

The 8XC196KC/KD supports three special operating modes: Idle, Powerdown, and On-Circuit Emulation (ONCE). Idle and Powerdown modes reduce power consumption; the On-Circuit Emulation (ONCE) mode is a test mode that electrically isolates the chip from the other system components. This chapter provides an overview of each mode and then describes how to enter and exit each mode. (Refer to Appendix A for descriptions of the instructions discussed in this chapter, to Appendix B for information about the signals, and to Appendix C for details about the registers.)

12.1. IDLE MODE

In Idle mode, the CPU clocks are frozen at logic state zero, but the peripheral clocks and CLKOUT remain active. The CPU stops executing instructions, but the internal RAM, timer/counters, serial port, and interrupt system continue functioning. Power consumption is reduced to about 40% of the normal consumption.

The bus-control pins (ALE, RD#, WR#, INST, and BHE#) are driven to their inactive states. If they are being used as I/O ports, Ports 3 and 4 will retain the value present in their data latches. If these ports are being used as the address/data bus, the pins will float.

If enabled, the Watchdog timer continues to run in Idle mode. The device must be awakened within every 64K state times to clear the WATCHDOG (0AH) register or the timer will reset the device.

12.1.1. Entering Idle Mode

To enter Idle mode, execute the IDLPD #1 instruction.

12.1.2. Exiting Idle Mode

Either an interrupt or a hardware reset will bring the device out of Idle mode. Because all peripherals remain active in Idle mode, any enabled interrupt source can generate the interrupt. When an interrupt occurs, the CPU exits Idle mode and begins executing the corresponding interrupt service routine. After completing the interrupt service routine, the CPU fetches and then executes the instruction that follows the IDLPD #1 instruction.

12.2. POWERDOWN MODE

Powerdown mode places the 8XC196KC/KD into a very low power state. If V_{CC} is maintained, the Special Function Registers (SFRs) and register RAM retain their data. All peripherals are powered down, all internal clocks are frozen at logic state zero, and the oscillator is shut off. Power consumption drops into the microwatt range (refer to the data sheet for exact specifications). I_{CC} is reduced to device leakage.

During Powerdown, the internal oscillator and clock generator are disabled. The bus-control pins (ALE, RD#, WR#, INST, and BHE#) are driven to their inactive states. All of the output pins hold the values in their data latches. If ports 3 and 4 were being used as I/O ports (EA# = 0), their pins will hold the last output value. If ports 3 and 4 were being used as an address/data bus (EA# = 1), their pins will float.

Figure 12-1 is a simplified drawing of the internal powerdown circuitry. Table 12-1 describes the internal signals shown in the figure. The IDLPD #2 instruction sets Q1, turning off the internal phase clocks, and clears Q2, turning off the internal oscillator.

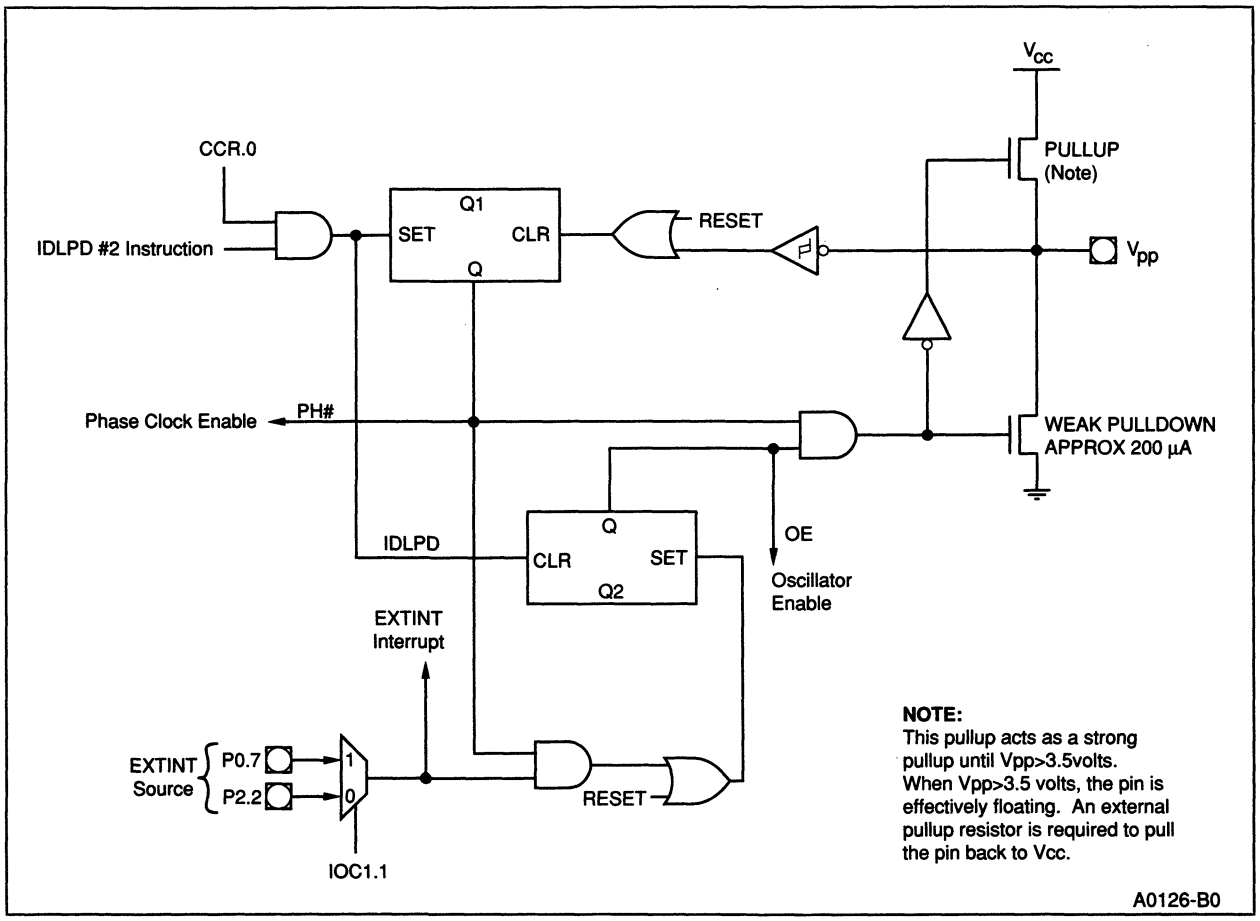

Vcc
CCR.0
PULLUP (Note)
IDLPD #2 Instruction
Q1
SET
CLR
Q
RESET
Vpp
Phase Clock Enable
PH#
WEAK PULLDOWN APPROX 200 µA
Q
IDLPD
CLR
SET
Q2
OE
Oscillator Enable
EXTINT Interrupt
EXTINT Source
P0.7
P2.2
1
0
RESET
IOC1.1
NOTE:
This pullup acts as a strong pullup until Vpp>3.5volts. When Vpp>3.5 volts, the pin is effectively floating. An external pullup resistor is required to pull the pin back to Vcc.
A0126-B0

Figure 12-1. Simplified Powerdown Circuit

Table 12-1. Internal Powerdown Signals

Signal Name	Description
IDLPD	Idle/Powerdown. This signal causes the device to enter the Powerdown mode. It disables the phase clock enable (PH#) and oscillator enable (OE) signals. The IDPLD #2 instruction generates this signal.
OE	Oscillator Enable. This signal enables the oscillator during normal operating mode. During Powerdown, it disables the internal oscillator.
PH#	Phase Clock Enable. This signal enables the internal phase clocks during normal operation. During Powerdown, it disables the internal phase clocks.
RESET	Internal Reset. This signal is an inverted version of the external RESET# signal.

12.2.1. Disabling Powerdown Mode

The Powerdown mode is disabled when the PD bit (CCR.0) is cleared. The CCR is loaded from the Chip Configuration Byte (location 2018H) when the device is reset.

12.2.2. Entering Powerdown Mode

Before entering Powerdown, complete the following tasks:

- Complete all serial port transmissions or receptions. When the device exits Powerdown, the serial port activity will continue where it left off and incorrect data may be transferred or received.
- Complete all analog conversions. If Powerdown occurs during the conversion, the result will be incorrect.
- If the Watchdog timer (WDT) is enabled, clear the WATCHDOG register just before issuing the Powerdown instruction. This ensures that the device can exit Powerdown cleanly. Otherwise, the WDT could reset the 8XC196KC/KD before the oscillator stabilizes. (The WDT cannot reset the device during Powerdown because the clock is stopped.)

Upon completion of these tasks, execute the IDLPD #2 instruction to enter Powerdown mode. The IDLPD #2 instruction sets Q1, turning off the internal phase clocks, and clears Q2, turning off the internal oscillator (see Figure 12-1).

NOTE

The EXTINT pin must be held low while the device is in Powerdown.

12.2.3. Exiting Powerdown Mode

Powerdown is exited by driving the V_{PP} pin low or by asserting either the RESET# or the EXTINT signal.

12.2.3.1. DRIVING V_{PP} LOW

The fastest way to exit Powerdown mode is to drive the V_{PP} pin low for at least 50 ns. This clears Q1 (see Figure 12-1), which enables the internal phase clocks, but it does not set Q2. The internal oscillator remains disabled, so use this method only when an external clock provides the clock signal.

12.2.3.2. ASSERTING RESET#

Another way to exit Powerdown mode is to assert the RESET# signal. RESET# must be held low until the oscillator has stabilized. The oscillator design must be characterized to determine how long it takes to stabilize. When using an external clock, RESET# must remain low for at least one state time. When RESET# is asserted, the internal reset signal clears Q1 and sets Q2, immediately enabling both the internal phase clocks and the internal oscillator (see Figure 12-1).

12.2.3.3. ASSERTING EXTINT

The final way to exit Powerdown mode is to assert the EXTINT signal for at least 50 ns. EXTINT is normally a sampled interrupt input. However, the Powerdown circuitry uses it as a level-sensitive input. The EXTINT interrupt need not be enabled for it to bring the device out of Powerdown. The EXTINT_SRC bit (IOC1.1) defines the source of EXTINT. Setting IOC1.1 selects P0.7 as the source; clearing IOC1.1 selects P2.2 as the source. Figure 12-2 shows the power-up and power-down sequence when using EXTINT to exit Powerdown.

If the EXTINT signal will be used to exit from Powerdown mode, we recommend that you connect the external RC circuit shown in Figure 12-3 to the V_{PP} pin. The discharging of the capacitor causes a delay that allows the oscillator to stabilize before the internal phase clocks are enabled.

When an EXTINT signal is received, Q2 is set, which enables the internal oscillator circuitry and turns on the weak pull-down (see Figure 12-1). This weak pull-down causes the external capacitor (C_1) to begin discharging at a typical rate of 100µA to 200 µA. When the V_{PP} voltage drops below the threshold voltage (about 2.5 V), the Schmitt trigger clears Q1, which enables the phase clocks (PH# = 0), and the device resumes code execution.

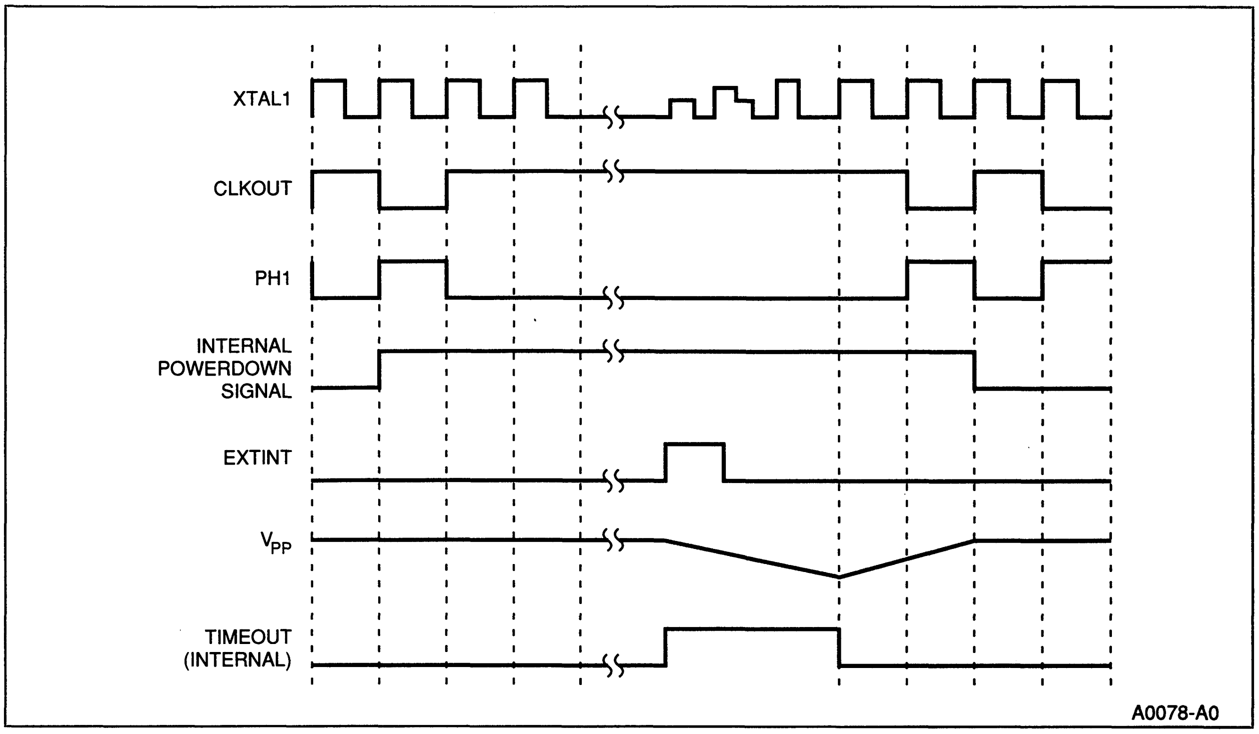


Figure 12-2. Power-Up and Power-Down Sequence

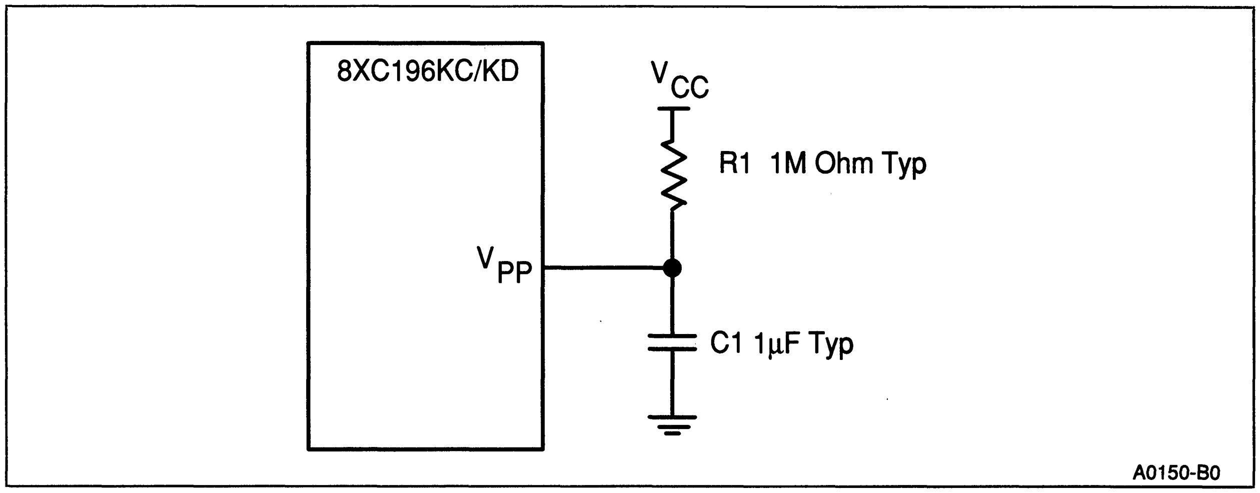


Figure 12-3. V_{PP} External RC Circuit

The pull-up transistor turns on when the V_{PP} voltage drops below the threshold and quickly pulls the pin back up to about 3.5 V. The pull-up becomes ineffective and the external resistor (R_1) takes over and pulls the voltage up to V_{CC} (see recovery time in Figure 12-4). The time constant follows an exponential charging curve; if C_1 = 1 µF and R_1 = 1 MΩ, the recovery time will take several seconds.

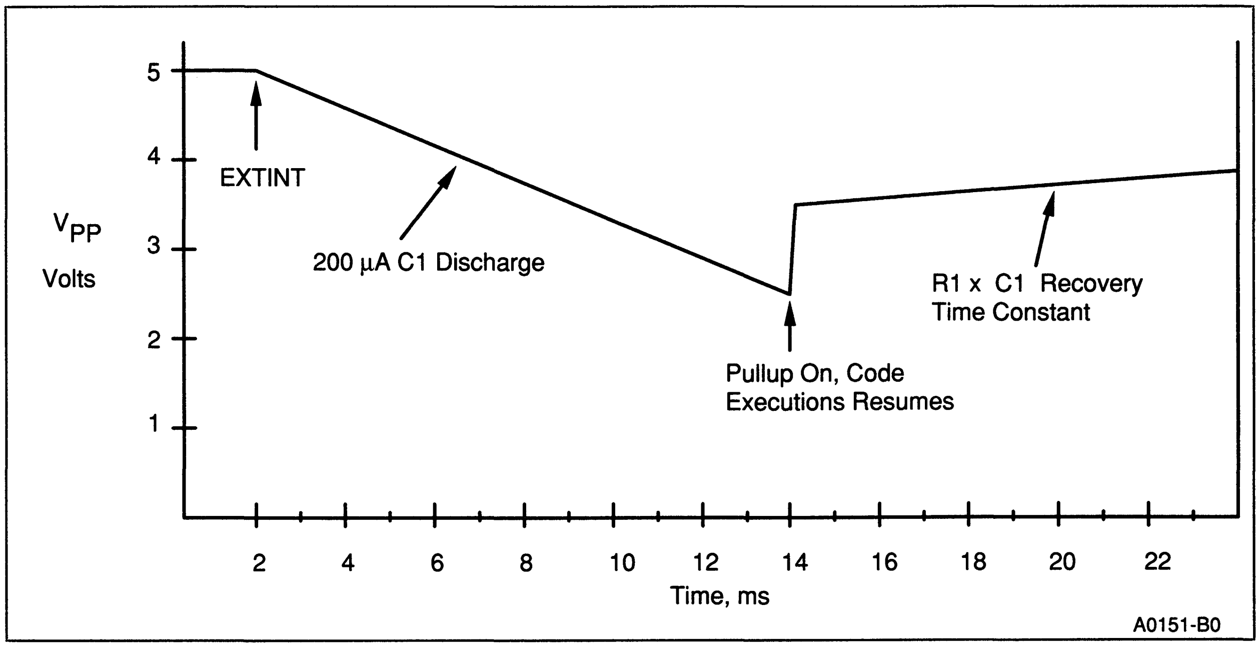


Figure 12-4. Typical V_{PP} Voltage While Exiting Powerdown

12.2.3.3.1. Selecting R_1 and C_1

Select components that produce a sufficient discharge time to permit the internal oscillator circuitry to stabilize. Because many factors can influence the discharge time requirement, you should always fully characterize your design under worst-case conditions to verify proper operation.

Select a resistor that will not interfere with the discharge current. An R_1 value between 200 KΩ and 1 MΩ should perform satisfactorily.

When selecting the capacitor, determine the worst-case discharge time needed for the oscillator to stabilize, then use this formula to calculate an appropriate value for C_1:

$$C_1 = \frac{T_{DIS} \times I}{\Delta V}$$

where:

C_1	is the capacitor value, in Farads
T_{DIS}	is the worst-case discharge time, in seconds
I	is the discharge current, in amperes
ΔV	is the threshold voltage

NOTE

If Powerdown is re-entered and exited before C_1 charges to V_{CC}, it will take less time for the voltage to ramp down to the threshold (because ΔV is less). The device will take less time to exit Powerdown.

For example, assume that the oscillator needs at least 12.5 ms to stabilize (T_{DIS} = 12.5 ms), ΔV is 2.5 V, and the discharge current is 200 µA. The minimum C_1 capacitor size is 1 µF.

$$C_1 = \frac{.0125 \times .0002}{2.5} = 1\ \mu F$$

When using an external oscillator, the value of C1 can be very small, to allow rapid recovery from Powerdown. For example, a 100 pF capacitor discharges in 1.25 µs.

$$T_{DIS} = \frac{C_1 \times \Delta V}{I} = \frac{1.0e-10 \times 2.5}{.0002} = 1.25\ \mu s$$

12.3. ONCE MODE

ONCE mode is a test mode that electrically isolates the 8XC196KC/KD from the other devices on the Printed Circuit Board (PCB). It is the only test mode available to users. ONCE mode is typically used to isolate the 8XC196KC/KD while programming discrete EEPROMs on the circuit board.

All pins, except XTAL1 and XTAL2, have weak pull-ups or pull-downs and are not truly high impedance (see Table B-3 in Appendix B). RESET# must remain high during ONCE. If RESET# is asserted during ONCE, the device will exit ONCE mode and enter the reset state.

12.3.1. Entering ONCE Mode

The ONCE mode is entered if P2.0 is held low during the rising edge of RESET#. The DC specifications for I_{OH1} and I_{IL1} (see the data sheet) define the voltage levels required to place the device into ONCE mode or to ensure that ONCE is not inadvertently activated.

12.3.2. Exiting ONCE Mode

Exit the ONCE Mode by asserting the RESET# signal and allowing P2.0 to float or be pulled high. When RESET# rises, normal operation resumes.

12.4. ENTERING RESERVED TEST MODES

When held high in combination with other pins, PWM0 (P2.5) invokes reserved test modes. To avoid entering the reserved test modes, do not hold PWM0 high during reset.

Interfacing with External Memory

13

CHAPTER 13
INTERFACING WITH EXTERNAL MEMORY

The 8XC196KC/KD can interface with a variety of external memory devices. It supports either a fixed 8-bit bus width or a dynamic 8-bit/16-bit bus width, internal Ready control for slow external memory devices, a bus-hold protocol that enables external devices to take over the bus, and several bus-control modes. These features provide a great deal of flexibility when interfacing with external memory devices. This chapter lists the external memory signals and describes the registers that control the external memory interface and the various external bus modes and features.

13.1. EXTERNAL MEMORY INTERFACE SIGNALS

Table 13-1 describes the external memory interface signals. Many of these signals have alternate functions. When applicable, the "Alternate Functions" column lists the alternate functions and the "Selected by" column lists the register bit and value that selects the function listed in column one.

Table 13-1. External Memory Interface Signals

Function Name	Alternate Functions	Selected by	Type	Description
AD0–AD7 AD8–AD15	P3.0–P3.7 P4.0–P4.7	Bus access to external address	I/O	System Address/Data Bus. These pins provide a multiplexed address and data bus. During the address phase of the bus cycle, address bits 0–15 are output onto the bus and can be latched using ALE or ADV#. During the data phase, 8- or 16-bit data is transferred. When a bus access is not occurring, these pins revert to their I/O port function.
ADV#	ALE	CCR.3 = 0	O	Address Valid. This active-low output signal is asserted only during external memory accesses. ADV# indicates that valid address information is available on the system address/data bus. The signal remains low while a valid bus cycle is in progress and is returned high as soon as the bus cycle completes. An external latch can use this signal to demultiplex the address from the address/data bus. A decoder can also use this signal to generate chip selects for external memory.
ALE	ADV#	CCR.3 = 1	O	Address Latch Enable. This active-high output signal is asserted only during external memory accesses. ALE indicates that valid address information is available on the system address/data bus and signals the start of a valid bus cycle. ALE differs from ADV# in that it does not remain active during the entire bus cycle. An external latch can use this signal to demultiplex the address from the address/data bus.

Table 13-1. External Memory Signals (Continued)

Function Name	Alternate Functions	Selected by	Type	Description
BHE#	WRH#	CCR.2 = 1	O	Byte High Enable. This active-low output signal is asserted only during word writes and high byte writes to external memory. Used in conjunction with A0 to select the memory byte to be written.
BREQ#	P1.5	WSR.7 = 1	O	Bus Request. This active-low output signal is asserted during a HOLD cycle when the bus controller has a pending external memory cycle. The bus controller can assert BREQ# at the same time it asserts HLDA#. BREQ# remains asserted until HOLD# is removed. When this function is active, the pin acts as a standard output (not quasi-bidirectional).
BUSWIDTH	—	CCR.1 = 1	I	Bus Width. When CCR.1 = 1, this signal selects the bus width during external accesses. When high BUSWIDTH selects a 16-bit bus width; when low it selects an 8-bit bus width. When CCR.1 = 0, the bus width is always 8 bits and the BUSWIDTH signal is ignored.
EA#	Programming mode select (EA# = V_{EA})	—	I	External memory access. This active-low signal directs memory accesses to off-chip memory. When high, it selects on-chip OTPROM.
HLDA#	P1.6	WSR.7 = 1	O	Bus Hold Acknowledge. This active-low output indicates that the bus controller has relinquished control of the bus. This occurs in response to an external device asserting the HOLD# signal.
HOLD#	P1.7	WSR.7 = 1	I	Bus Hold. This active-low signal indicates that an external device is requesting control of the bus. When this function is active, the pin acts as a standard input (not quasi-bidirectional).
INST	—	—	O	Instruction Fetch. The signal is valid only during external memory read cycles. When high, INST indicates that an instruction is being fetched; when low, it indicates that data is being read. INST can be used in applications that require separate memory banks for instructions and data. (Note that CCB bytes and interrupt vectors are considered data.)
RD#	—	—	O	External Read. This active-low output signal is the external memory read signal.
READY	—	—	I	Ready. This active-high input signal indicates that external memory is ready to send or receive data.
WR#	WRL#	CCR.2 = 1	O	External Write. This active-low output signal is the external memory write signal.

Table 13-1. External Memory Signals (Continued)

Function Name	Alternate Functions	Selected by	Type	Description
WRH#	BHE#	CCR.2 = 0	O	Write High Byte. In 16-bit bus mode, WRH# is asserted during high byte writes and word writes. In 8-bit bus mode (CCR.1 = 0), WRH# is asserted for high byte, low byte, and word writes.
WRL#	WR#	CCR.2 = 0	O	Write Low Byte. In 16-bit bus mode, WRL# is asserted during low byte writes and word writes. In 8-bit bus mode (CCR.1 = 1), WRL# is asserted for high byte, low byte, and word writes.

13.2. CHIP CONFIGURATION REGISTER

The Chip Configuration Register (CCR) controls the bus width, write strobe signal generation, address valid signal generation, and the number of wait states that can be inserted while the READY pin is held low. It is the first byte fetched from memory after a device reset. Figure 13-1 shows the format of the CCR.

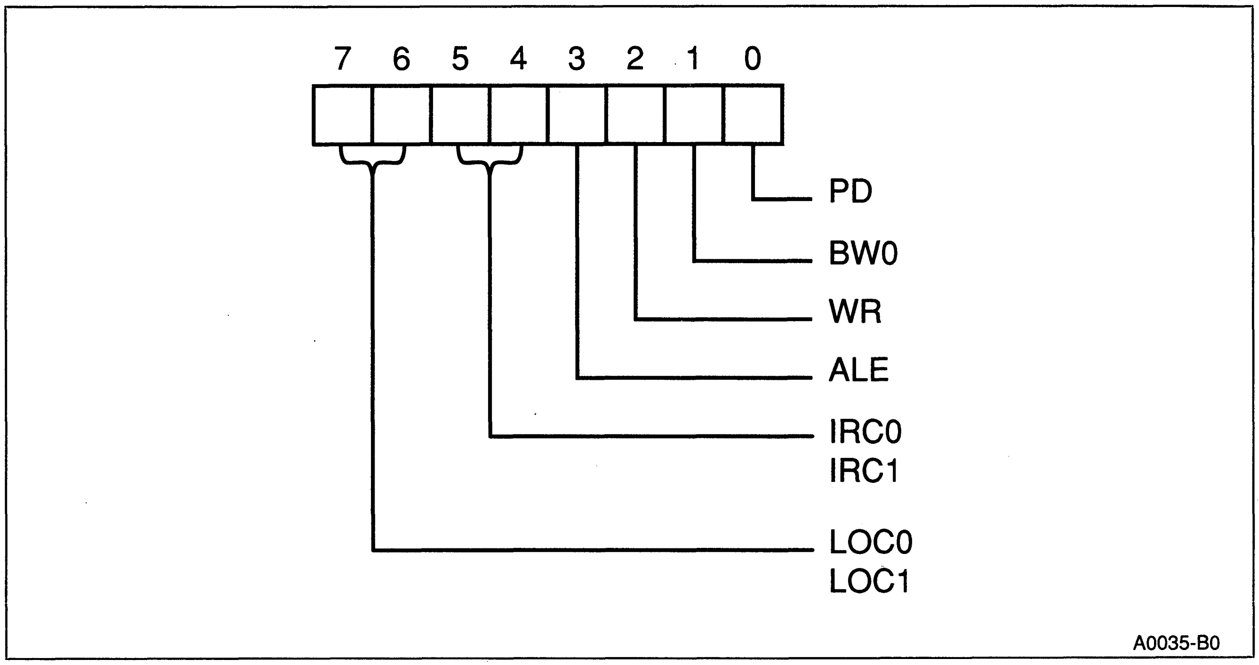


Figure 13-1. Chip Configuration Register

When the device is reset, the bus controller fetches the Chip Configuration Byte (CCB) from location 2018H and loads it into the CCR. The address is always strongly driven during a CCB fetch, just as in normal operation. The CCB is fetched from internal OTPROM if the EA# signal is high and from external memory if the EA# signal is low. The CCR is loaded only during the reset sequence. Once it is loaded, the CCR cannot be changed until the next device reset. Typically, the CCB is programmed once when the user program is compiled and is not redefined during normal operation.

The default setting of the CCR allows the READY and BUSWIDTH signals to control the timing and bus width during the CCB fetch. If the CCB is stored in slow external memory devices, pull the READY signal low. This causes the bus controller to automatically insert up to three wait states into the CCB fetch. (CCR.4 and CCR.5 control the number of wait states; the default configuration specifies a maximum of three wait states.) If the CCB is located in 16-bit external memory, drive the BUSWIDTH signal high; if the CCB is located in 8-bit external memory, drive BUSWIDTH low.

Some early designs may hold the BUSWIDTH pin low (8-bit bus) during the CCB fetch even when 16-bit external memory devices are used. To prevent bus contention, we recommend that location 2019H be loaded with 20H. Under these conditions, the 8XC196KC/KD will strongly drive 20H out onto Port 4 during the entire CCB fetch. If the external memory outputs 20H on its high byte, there will be no bus contention.

After the CCB is loaded into the CCR, the bus width is configured as either fixed 8-bit or dynamically controlled by the BUSWIDTH signal, as specified by CCR.1.

13.3. BUS WIDTH AND MEMORY CONFIGURATIONS

The 8XC196KC/KD external bus can operate as either an 8-bit or 16-bit multiplexed address/data bus (see Figure 13-2). The value of CCR.1 defines the bus width as either a fixed 8-bit bus width or a dynamic 16-bit/8-bit bus width controlled by the BUSWIDTH signal. If CCR.1 is clear, the bus controller is locked into an 8-bit bus mode. Expect some performance degradation when executing code from an 8-bit bus. Word reads and writes to external memory take an extra bus cycle, and the prefetch queue is not fully utilized.

If CCR.1 is set, the BUSWIDTH signal controls the bus width. The bus is 16 bits wide when BUSWIDTH is high and 8 bits wide when BUSWIDTH is low. The BUSWIDTH signal is sampled after the address is on the bus, as shown in Figure 13-3.

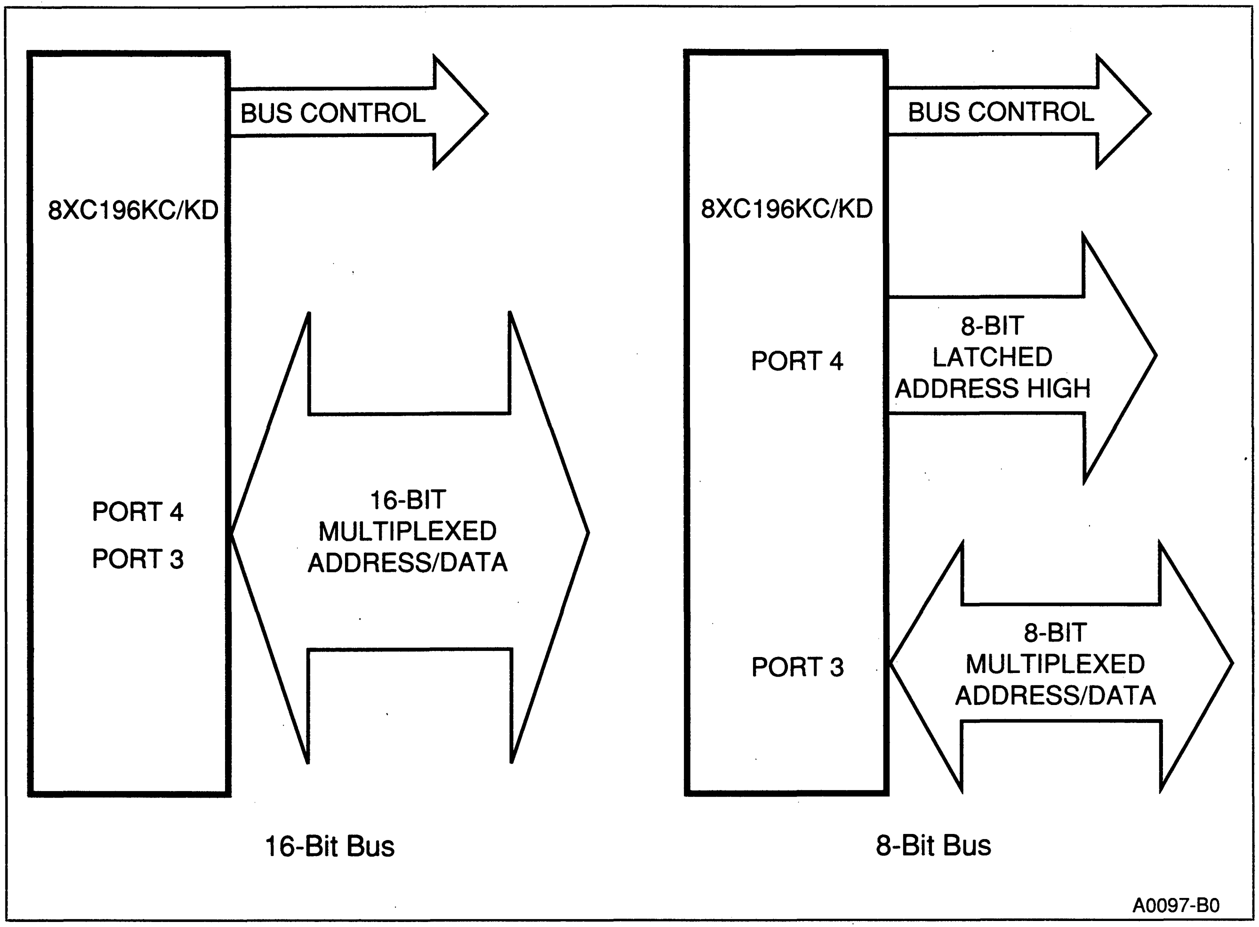


Figure 13-2. Bus Width Options

The BUSWIDTH signal can be used in numerous applications. For example, a system could store code in a 16-bit memory device and data in 8-bit memory device. The BUSWIDTH signal could be tied to the chip-select input of the 8-bit memory device as shown in Figure 13-11 on page 13-16. When BUSWIDTH is low, it enables 8-bit bus mode and selects the 8-bit memory device. When BUSWIDTH is high, it enables 16-bit bus mode and deselects the 8-bit memory device.

13.3.1. Timing Requirements for BUSWIDTH

When using BUSWIDTH to dynamically change between 8-bit and 16-bit bus widths, setup and hold timings must be met for proper operation (see Figure 13-3). Because a decoded, valid address is used to generate the BUSWIDTH signal, the setup time is specified relative to the address being valid. This specification, T_{AVGV}, indicates how much time one has to decode the valid address and generate a valid BUSWIDTH signal.

BUSWIDTH must be held valid until the minimum hold specification, T_{CLGX}, has been met. Typically this hold time is 0 ns minimum after CLKOUT goes low. In all cases, refer to the data sheet for current specifications for T_{AVGV} and T_{CLGX}.

NOTE

Earlier HMOS devices used a BUSWIDTH setup timing that was referenced to the falling edge of ALE (T_{LLGV}). This specification is not meaningful for CMOS devices, which use an internal two-phase clock; it is included for comparison only.

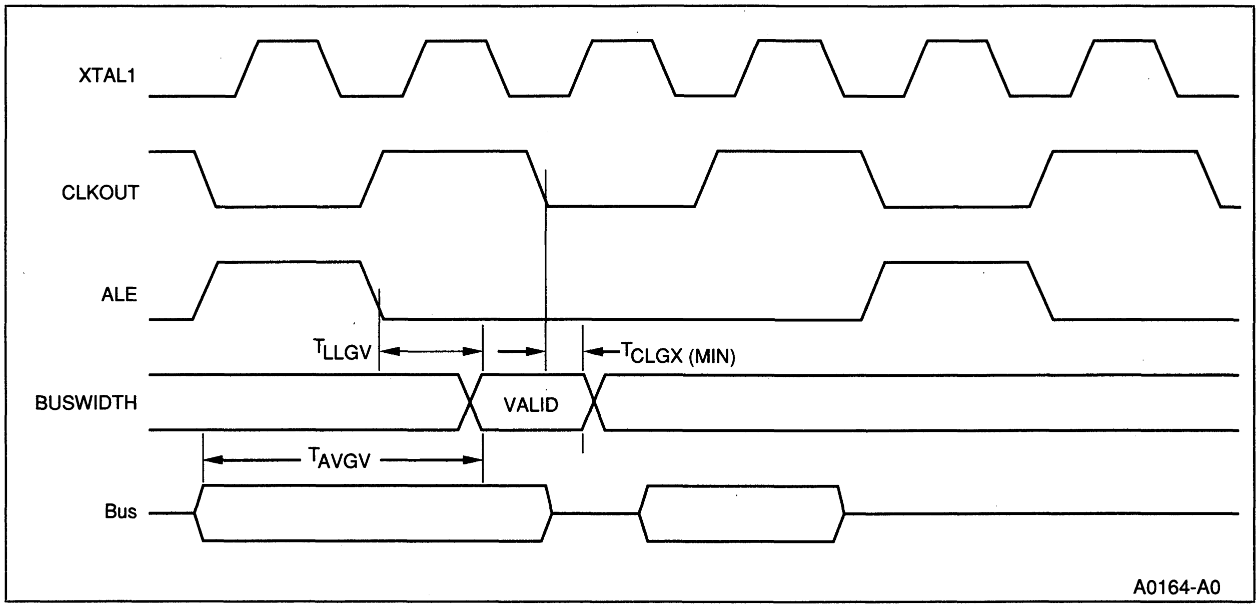


Figure 13-3. BUSWIDTH Timing Diagram

13.3.2. 16-Bit Bus Width

When the 8XC196KC/KD is configured to operate in the 16-bit bus-width mode (that is, when CCR.1 is set and the BUSWIDTH signal is high), lines AD0–AD15 form a 16-bit multiplexed address/data bus. The address/data bus shares pins with Ports 3 and 4 (see Table 13-1 on page 13-1). Figure 13-4 shows an idealized timing diagram for the external read and write cycles. See the data sheet for a list of specifications that must be met by external memory devices.

Address Latch Enable (ALE) demultiplexes the address bus by strobing transparent latches (such as a 74AC373). The address ("Address Out" on Bus) will be valid on the bus before ALE drops.

13.3.2.1. 16-BIT READ CYCLES

The bus controller floats the bus and then drives RD# low so that it can receive data. The external memory must put data ("Data In") onto the bus before the rising edge of RD#. The data sheet specifies the maximum time the memory device has to output valid data after RD# is asserted. When INST is asserted, it indicates that the read operation is an instruction fetch.

13.3.2.2. 16-BIT WRITE CYCLES

The bus controller drives WR# low, then puts data onto the bus. The rising edge of WR# signifies that data is valid. At this time, the external system must latch the data.

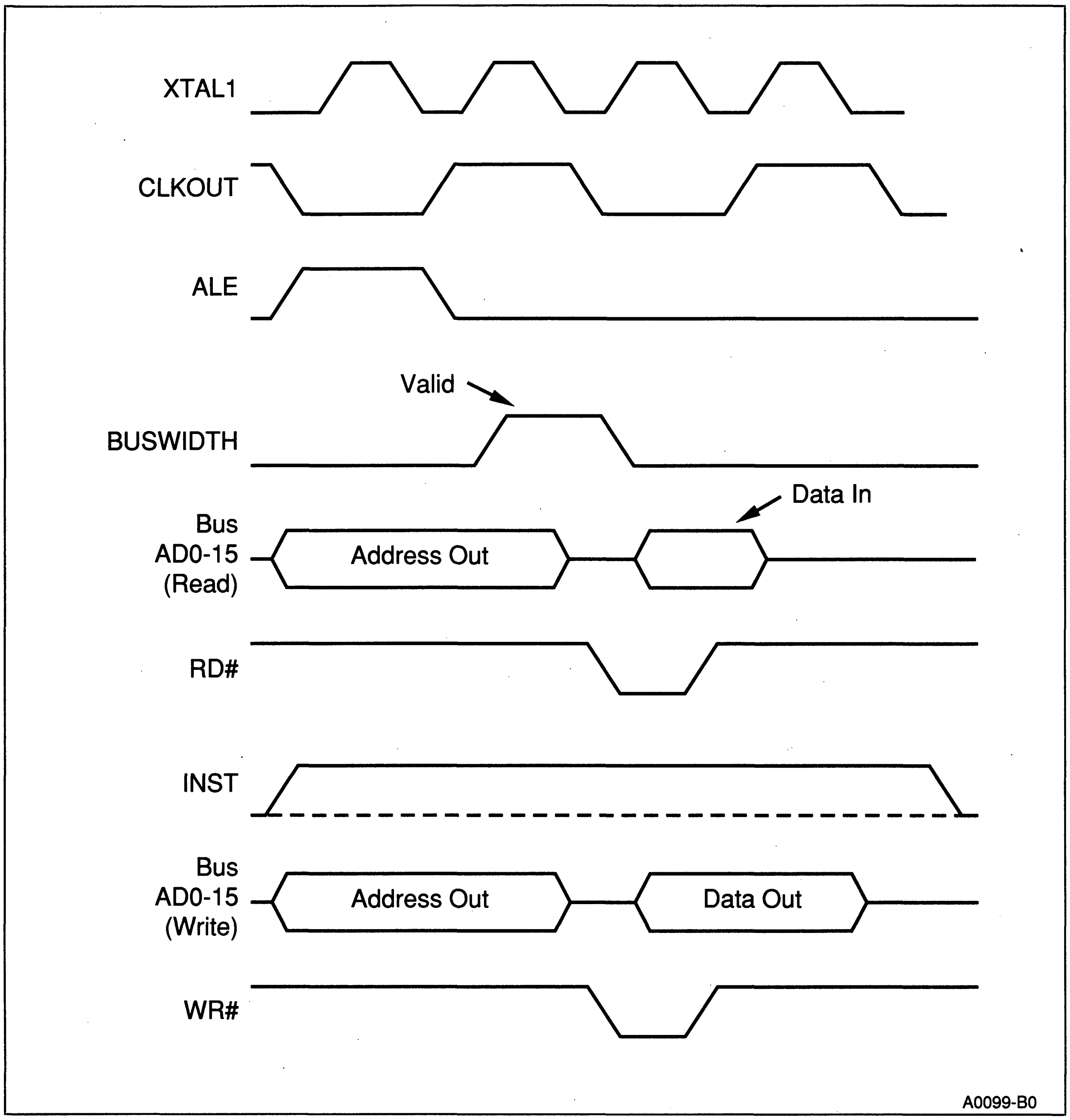


Figure 13-4. Idealized Bus Timings for 16-Bit External Bus

13.3.3. 8-Bit Bus Width

When the 8XC196KC/KD is configured to operate in the 8-bit bus mode, lines AD0–AD7 form a multiplexed lower address and data bus. Lines AD8–AD15 are not multiplexed; the upper address is latched and remains valid throughout the bus cycle. Figure 13-5 shows an idealized timing diagram for the external read and write cycles.

The ALE signal is used to demultiplex the lower address by strobing a transparent latch (such as a 74AC373).

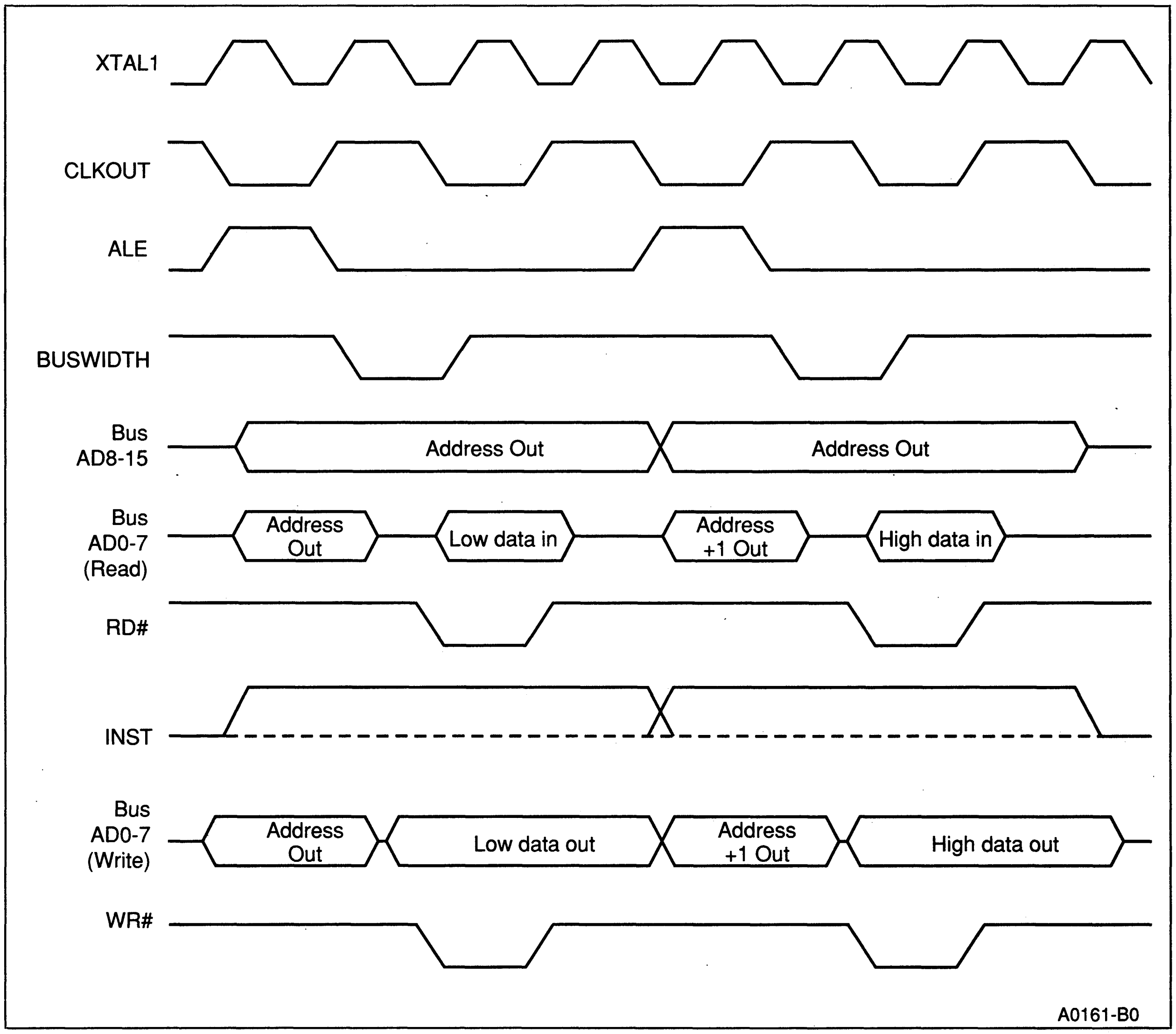


Figure 13-5. Idealized Bus Timings for 8-Bit External Bus

13.3.3.1. 8-BIT BUS READ CYCLES

After ALE falls, the bus controller floats the bus and drives the RD# signal low. The external memory then must put its data on the bus. That data must be valid at the rising edge of the RD# signal. To read a data word, the bus controller performs two consecutive reads, reading the low byte first, followed by the high byte.

13.3.3.2. 8-BIT BUS WRITE CYCLES

After ALE falls, the bus controller outputs data on AD0–7 and then drives WR# low. The external memory must latch the data by the time WR# goes high. That data will be valid on the bus until slightly after WR# goes high. To write a data word, the bus controller performs two consecutive writes, writing the low byte first, followed by the high byte.

13.4. READY CONTROL

The internal Ready control circuitry allows slow external memory devices to increase the length of the 8XC196KC/KD's read and write bus cycles. If the external memory device is not ready for access, it pulls the READY signal low and holds it low until it is ready to complete the operation. While READY is low, the bus controller inserts wait states into the bus cycle.

CCR bits 4 and 5 (IRC0 and IRC1) define the maximum number of wait states that will be inserted (see Table 13-2). When both bits are set, the bus controller inserts wait states until the external memory device asserts the READY signal. Otherwise, the bus controller inserts wait states until either the external memory device asserts the READY signal or the number of wait states equals the number (1, 2, or 3) defined by CCR.4 and CCR.5.

Table 13-2. Wait State Control

CCR.5	CCR.4	Maximum Wait States
0	0	Limit to 1
0	1	Limit to 2
1	0	Limit to 3
1	1	Infinite (controlled by READY signal)

When using the READY signal to control the number of wait states, be sure to add external hardware to count wait states and release READY within a specified period of time. (See the data sheet for READY specifications.) Otherwise, a defective memory device could tie up the address/data bus indefinitely.

NOTE

Ready control is valid only for external memory; you cannot add wait states when accessing internal RAM and OTPROM space.

13.4.1. Timing Requirements for READY

Setup and hold timings must be met when using the READY signal to insert wait states into a bus cycle (see Figure 13-6). Because a decoded, valid address is used to generate the READY signal, the setup time is specified relative to the address being valid. This specification, T_{AVYV} , indicates how much time one has to decode and assert READY from when the address is valid. The READY signal must be held valid until the T_{CLYX} timing specification is met. Typically, this is a minimum of 0 ns from the time CLKOUT goes low. Do not exceed the maximum T_{CLYX} specification or additional (unwanted) wait states might occur. In all cases, refer to the data sheets for the current specifications for T_{AVYV} and T_{CLYX}.

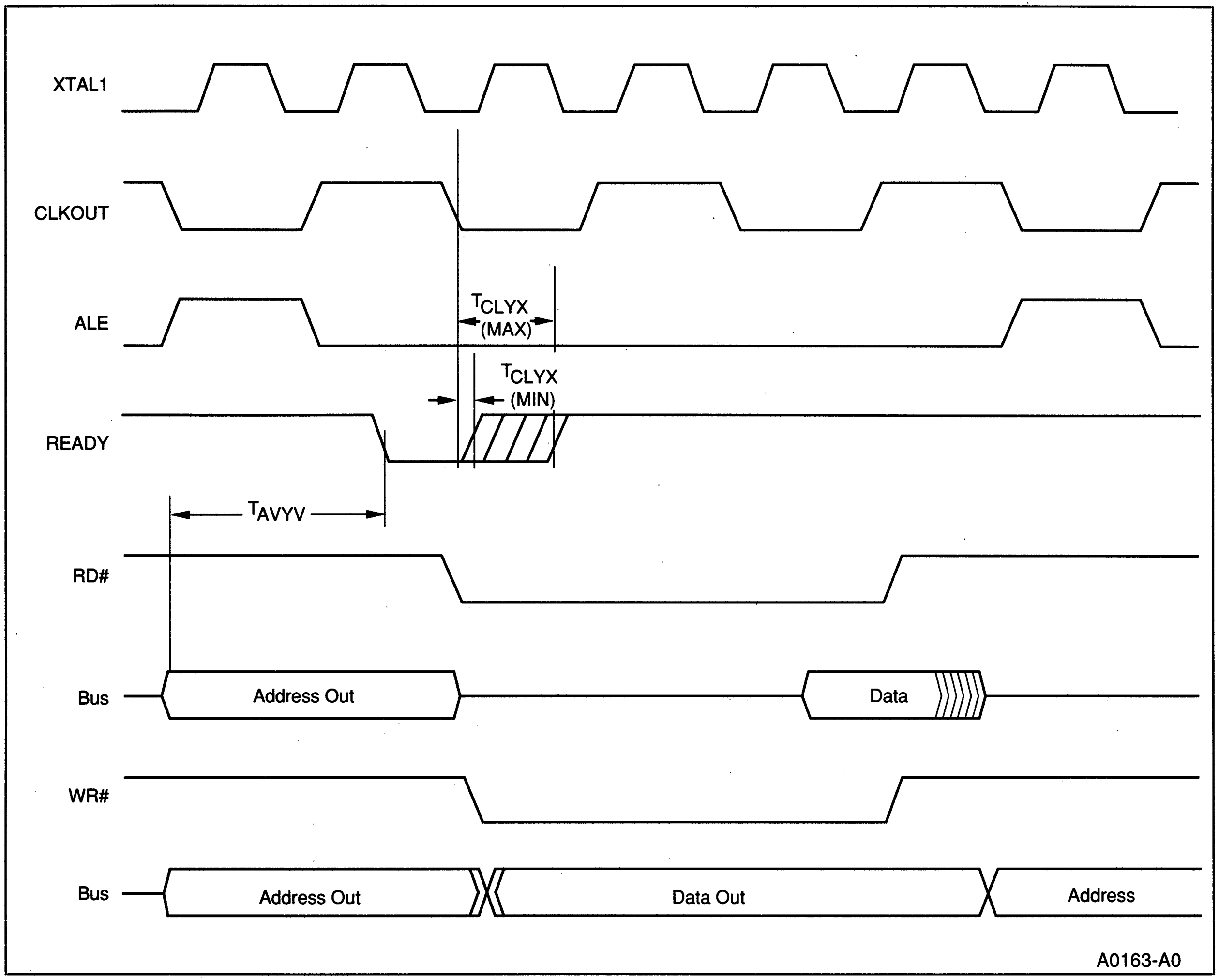


Figure 13-6. READY Timing Diagram

When the CCR is programmed for limited wait states (i.e., 1, 2, or 3 maximum), you may want to tie READY directly to the chip enable of slow external memory device. This is a simple way to insert 1–3 wait states when the device is selected.

NOTE

Earlier HMOS devices specified a READY setup time that was referenced to the falling edge of ALE (T_{LLYV}). This specification is not meaningful for CMOS devices, which use an internal 2-phase clock; it is included for comparison only.

13.5. BUS-HOLD PROTOCOL

The 8XC196KC/KD supports a bus-hold protocol that allows external devices to gain control of the address/data bus. The protocol uses three signals, HOLD# (Hold Request), HLDA# (Hold Acknowledge), and BREQ# (Bus Request), which are Port 1 alternate functions. When a device wants to use the 8XC196KC/KD bus, it asserts the HOLD# signal. HOLD# is sampled while CLKOUT is low. The 8XC196KC/KD responds by releasing the bus and asserting HLDA#. The address/data bus floats and ALE, RD#, WR#, and BHE# are weakly held in their inactive states during this hold time. Figure 13-7 shows the timing for bus-hold protocol. Refer to the data sheet for specific timing requirements.

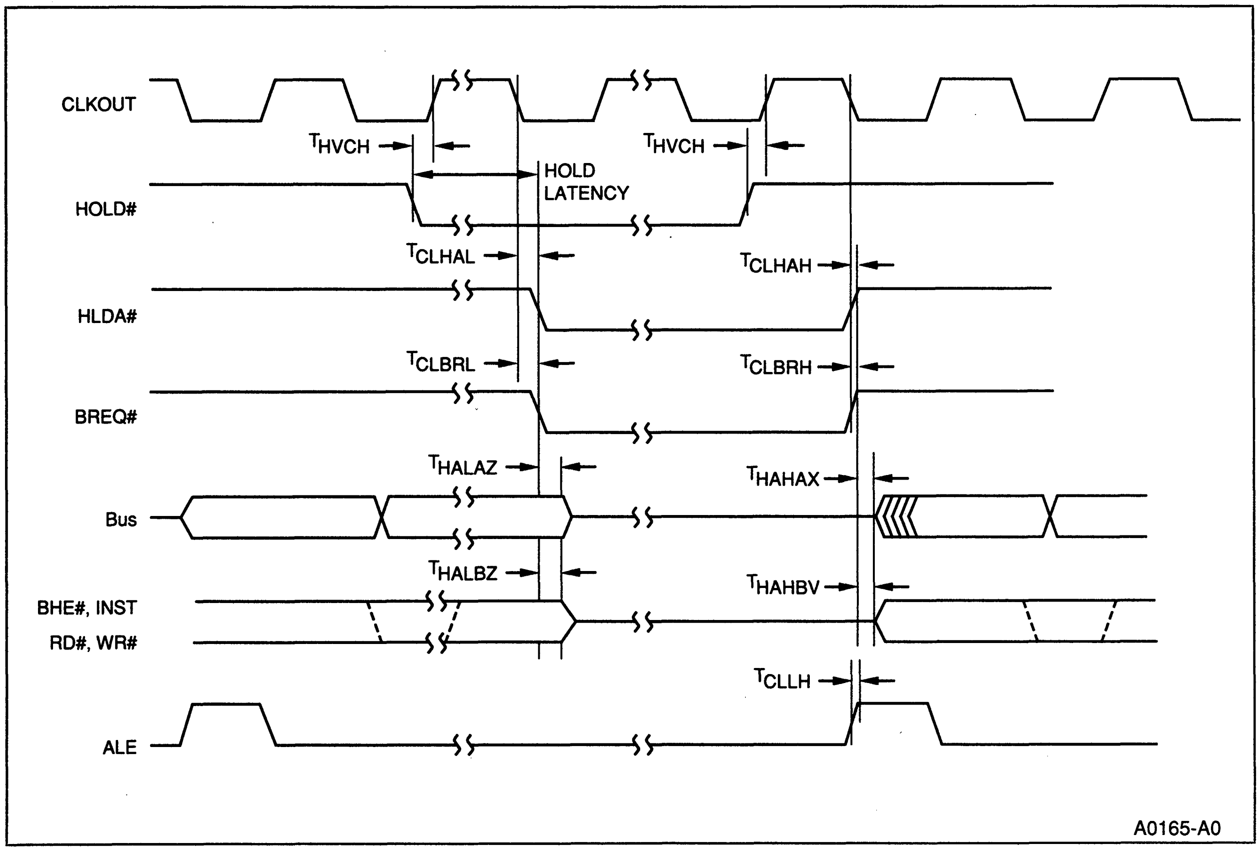


Figure 13-7. HOLD#, HLDA# Timings

When the external device is finished with the bus, it relinquishes control by driving HOLD# high. In response, the 8XC196KC/KD drives HLDA# high and assumes control of the bus. While the 8XC196KC/KD is in hold, it can request control of the bus by asserting BREQ#. After the external device responds by driving HOLD# high, the 8XC196KC/KD exits hold and then deasserts BREQ# and HLDA#.

13.5.1. Enabling the Bus-Hold Protocol

The HLDEN bit in the Window Select register (WSR.7) enables the bus-hold protocol. Setting this bit enables the protocol and selects the alternate functions of port pins P1.7 (HOLD#), P1.6 (HLDA#), and P1.5 (BREQ#). Once the alternate functions are enabled, the pins act as standard (not quasi-bidirectional) inputs or outputs. Once the bus-hold protocol has been selected, the port function cannot be reselected without resetting the device. However, the hold function can be dynamically enabled and disabled as described below.

13.5.2. Disabling the Bus-Hold Protocol

To disable hold requests, clear WSR.7. The 8XC196KC/KD does not take over the bus immediately after HLDEN is cleared. Instead, it waits for the current HOLD# request to finish and then disables the bus-hold feature and ignores any new requests until the bit is set again.

Sometimes it is important to prevent another device from taking control of the bus while a block of code is executing. One way to protect a block of code is to clear WSR.7, and then execute a JBC instruction to check the status of the HLDA# signal. The JBC instruction prevents the RALU from executing the protected block until current HOLD# requests are serviced and the hold feature is disabled.

```
        DI                          ;Disable interrupts to prevent
                                    ;code interruption
        ANDB WSR, #7FH              ;Disable hold requests
WAIT:   JBC IOPORT1,6, WAIT         ;Check the HLDA# signal
                                    :If set, execute
                                    ;protected instruction

        ORB WSR,#80H                ;Enable hold requests
        EI                          :Enable interrupts
```

13.5.3. Hold Latency

When an external device asserts HOLD#, the 8XC196KC/KD finishes the current bus cycle and then asserts HLDA#. The time it takes the 8XC196KC/KD to assert HLDA# after the external device asserts HOLD# is called *hold latency* (see Figure 13-7). Table 13-3 lists the maximum hold latency for each type of bus cycle.

Table 13-3. Maximum Hold Latency

Bus Cycle Type	Maximum Hold Latency
Internal Execution or Idle Mode	1.5 State Times
16-Bit External Execution	2.5 State Times
8-Bit External Execution	4.5 State Times

13.5.4. Regaining Bus Control

While HOLD# is asserted, the 8XC196KC/KD can execute code out of internal memory. When it needs to access external memory, it asserts BREQ# and waits for the external device to deassert HOLD#. After asserting BREQ#, the 8XC196KC/KD cannot respond to any interrupt requests, including NMI, until the external device deasserts HOLD#. One state time after HOLD# goes high, the 8XC196KC/KD deasserts HLDA# and resumes control of the bus.

If the 8XC196KC/KD is reset while in hold, bus contention can occur. For example, a CPU-only device would try to fetch the Chip Configuration Byte from external memory after RESET# is brought high. Bus contention would occur because both the external device and the 8XC196KC/KD would attempt to access memory. One solution is to use the RESET# signal as the system reset; then all bus masters (including the 8XC196KC/KD) are reset at once. Chapter 11, "Minimum Hardware Considerations," shows system reset circuit examples.

13.6. BUS-CONTROL MODES

CCR.2 and CCR.3 define which bus-control signals will be generated during external read and write cycles. Table 13-4 lists the four bus-control modes and shows the CCR.3 and CCR.2 settings for each.

Table 13-4. Bus-Control Mode

Bus-Control Mode	Bus-Control Signals	CCR.3	CCR.2
Address Valid with Write Strobe mode	ADV#, RD#, WRL#, WRH#	0	0
Address Valid Strobe mode	ADV#, RD#, WR#, BHE#	0	1
Write Strobe mode	ALE#, RD# WRL#, WRH#	1	0
Standard Bus-Control mode	ALE, RD#, WR#, BHE#	1	1

13.6.1. Standard Bus-Control Mode

In the standard bus-control mode, the 8XC196KC/KD generates the standard bus-control signals: ALE, WR#, RD# and BHE# (see Figure 13-8). ALE is asserted while the address is driven, and it can be used to latch the address externally. When low, BHE# selects the bank of memory that is addressed by the high byte of the data bus. RD# is asserted for every external memory read, and WR# is asserted for every external memory write.

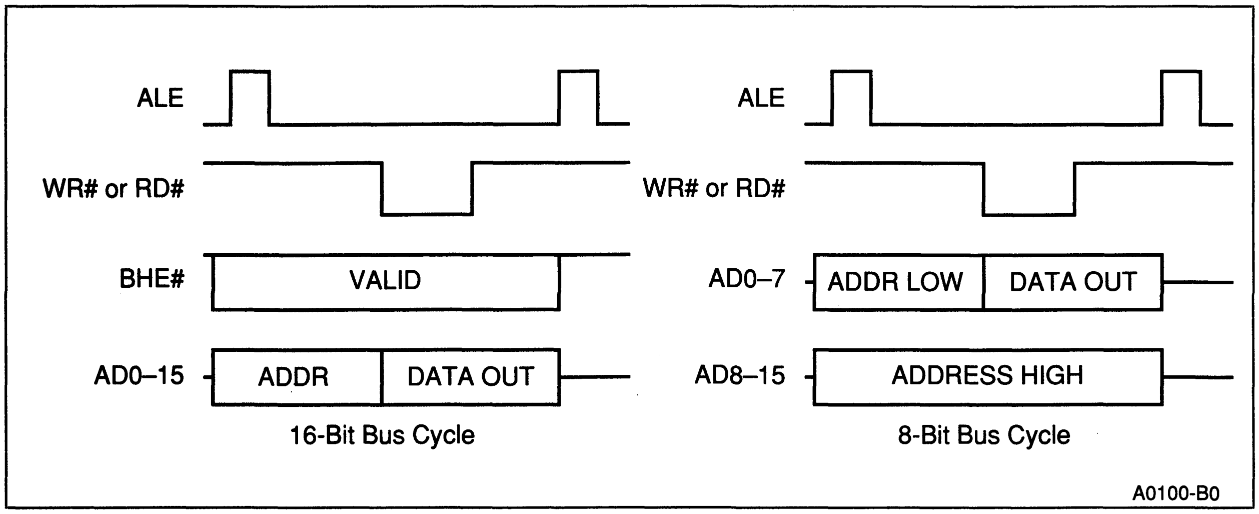


Figure 13-8. Standard Bus Control

When the device is configured to use a 16-bit bus, separate low- and high-byte write signals must be generated for single-byte writes. Figure 13-9 shows a sample circuit that combines BHE# and A0 to produce these signals (WRL# and WRH#). A similar pair of signals for read is unnecessary. For a single-byte read with the 16-bit bus, both bytes are put onto the data bus and the processor discards the unwanted byte.

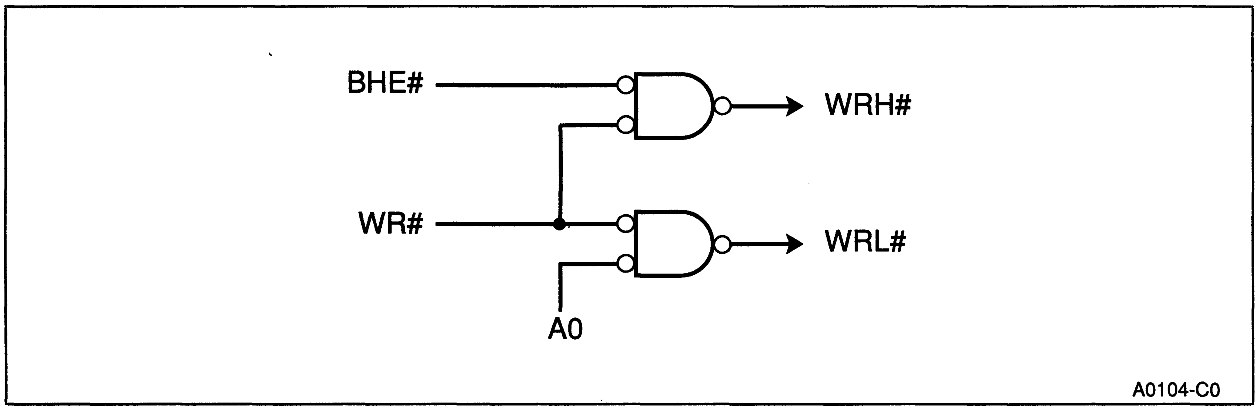


Figure 13-9. Decoding WRL# and WRH#

Figure 13-10 shows an 8-bit system with both EPROM and RAM. The EPROM is the lower half of memory, and the RAM is the upper half. This system configuration uses the most-significant address bit (A15) as the chip-select signal and ALE as the address-latch signal.

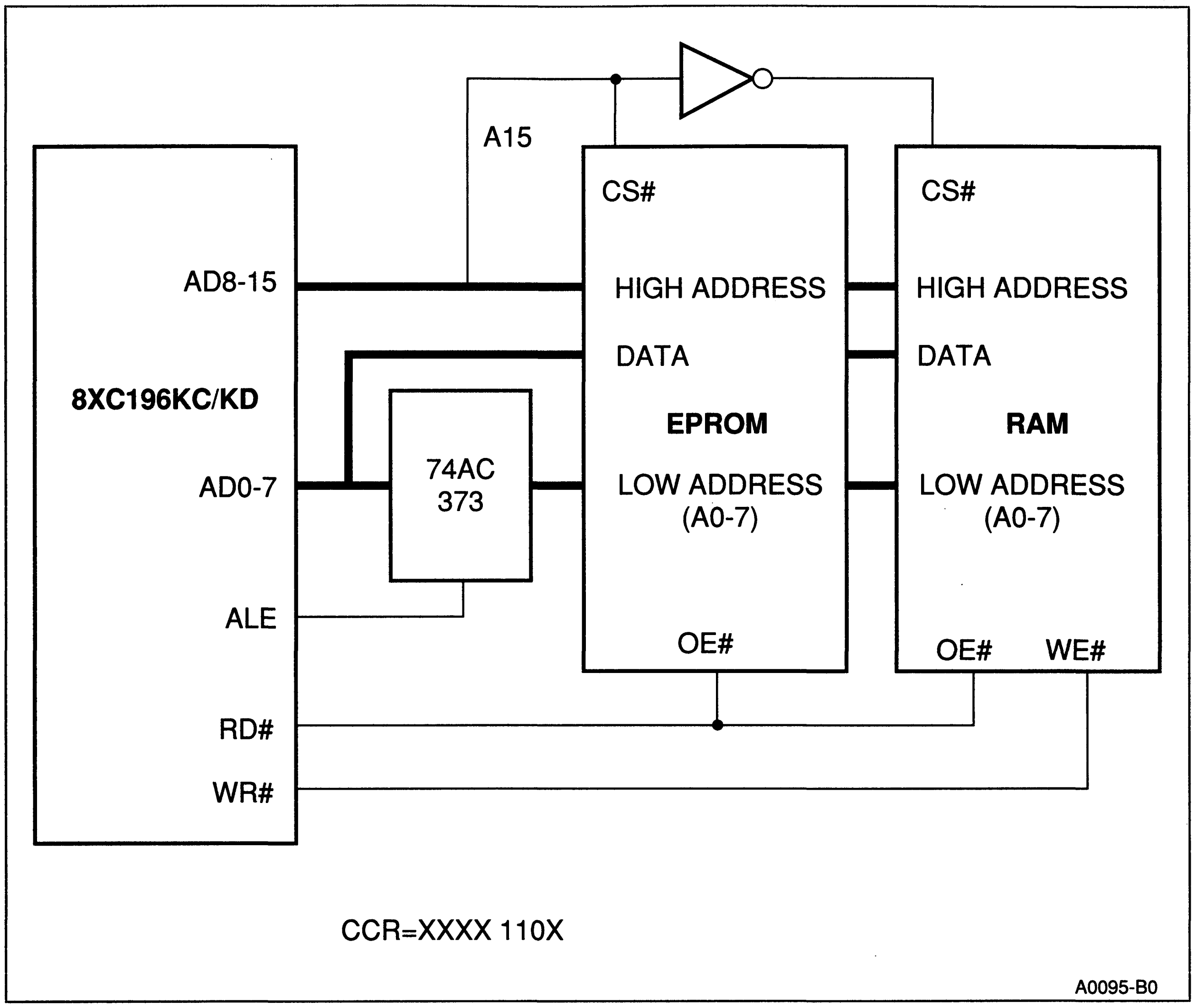


Figure 13-10. 8-Bit System with EPROM and RAM

Figure 13-11 shows a system that uses the dynamic bus-width feature (CCR.1 set). Code is executed from the two EPROMs and data is stored in the byte-wide RAM. The RAM is the high memory and is selected by driving A15 high, which also selects the 8-bit bus width mode by driving the BUSWIDTH signal low.

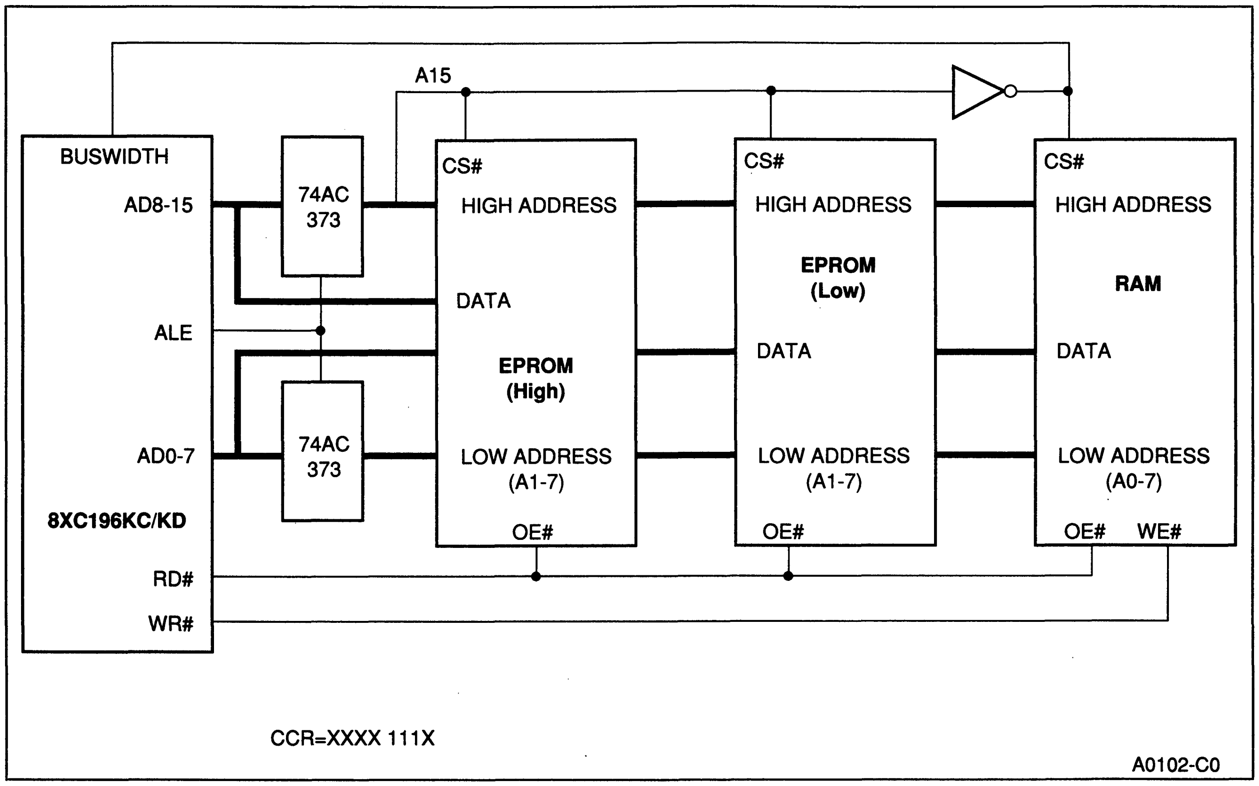


Figure 13-11. 16-Bit System with Dynamic Bus Width

13.6.2. Write Strobe Mode

The Write Strobe mode eliminates the need to externally decode high- and low-byte writes to external 16-bit RAM in 16-bit bus mode. When the Write Strobe mode is selected, the 8XC196KC/KD generates WRL# and WRH# instead of WR# and BHE#. WRL# is asserted for all low byte writes (even addresses) and all word writes. WRH# is asserted for all high byte writes (odd addresses) and all word writes. In the 8-bit bus mode, WRH# and WRL# are asserted for both even and odd addresses. The Write Strobe mode timing is shown in Figure 13-12.

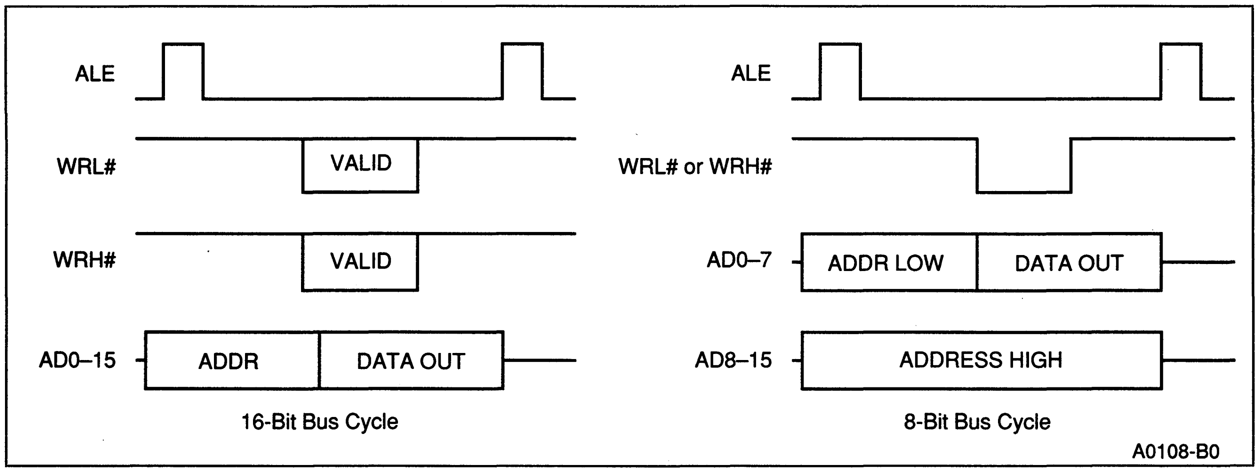


Figure 13-12. Write Strobe Mode

Figure 13-13 shows a 16-bit system that uses two EPROMs and two RAMs. This example is configured to use the Write-Strobe mode. ALE latches the address, and A15 is the chip select for the EPROMs and RAMs. WRL# is asserted during low byte writes and word writes. WRH# is asserted during high byte writes and word writes. Note that RAM devices do not use A0. WRL# and WRH# determine whether the low byte (A0 = 0) or high byte (A0 = 1) is selected.

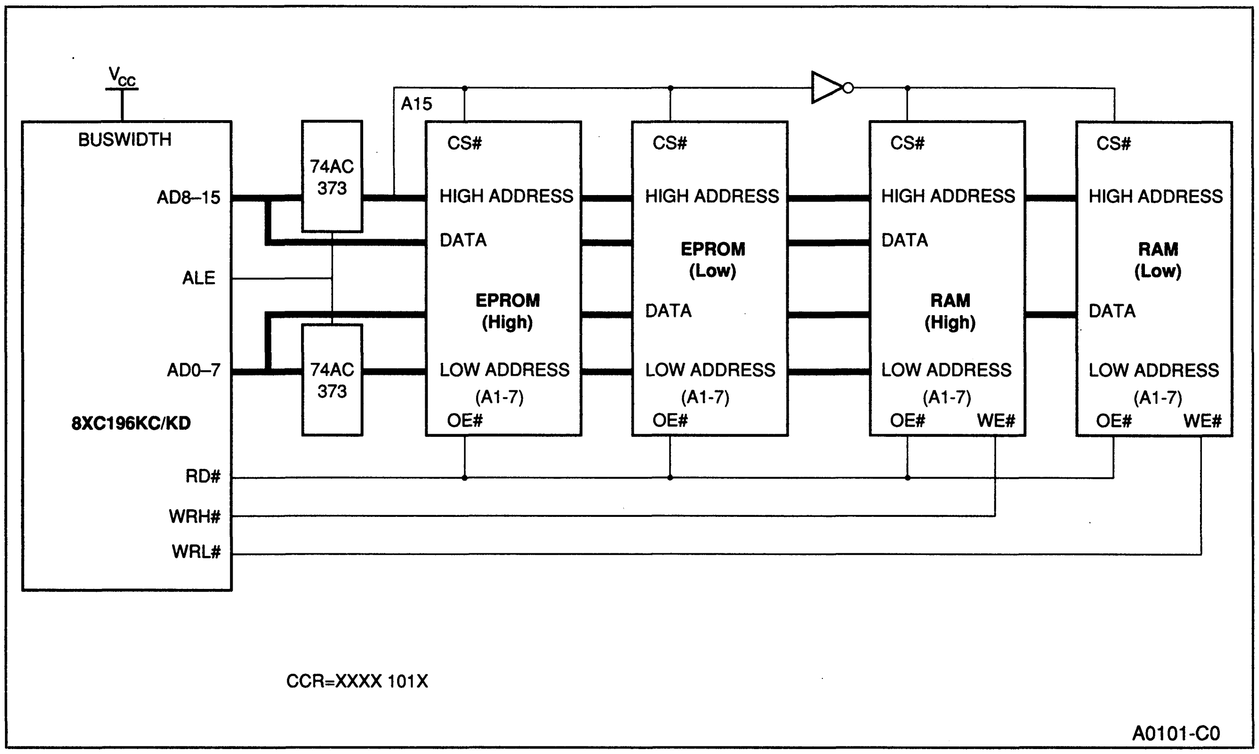


Figure 13-13. 16-Bit System with Single-Byte Writes to RAM

13.6.3. Address Valid Strobe Mode

When the Address Valid Strobe mode is selected, the 8XC196KC/KD generates the Address Valid signal (ADV#) instead of the Address Latch Enable signal (ALE). ADV# is asserted after an external address is valid (see Figure 13-14). This signal can be used to latch the valid address and simultaneously enable an external memory device.

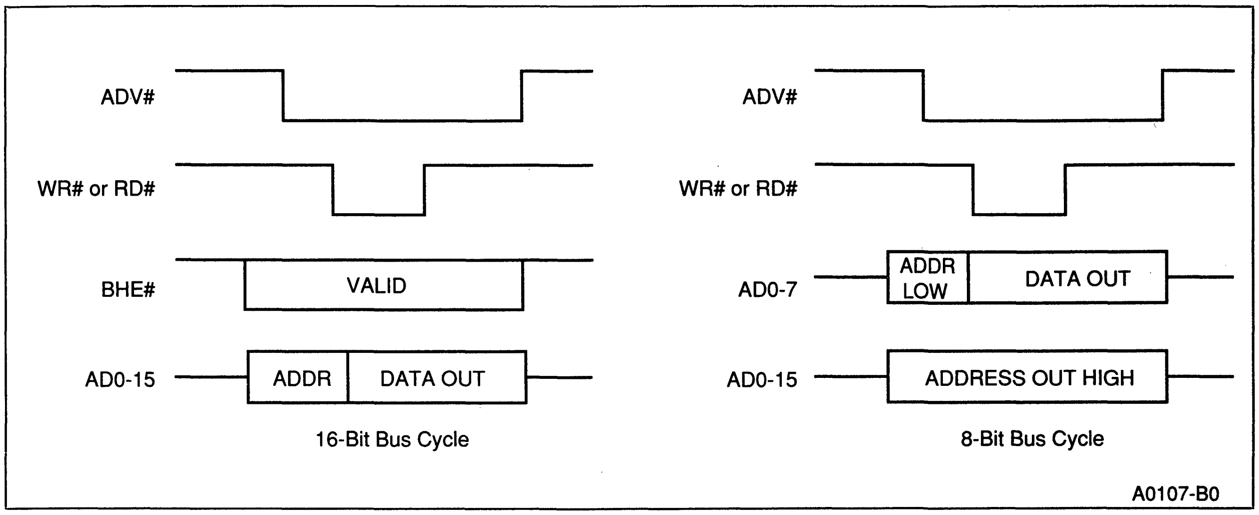


Figure 13-14. Address Valid Strobe Mode

The difference between ALE and ADV# is that ADV# is asserted for the entire bus cycle, not just to latch the address. Figure 13-15 shows the difference between ALE and ADV# for a single read or write cycle. Note that for back-to-back bus access, the ADV# function will look identical to the ALE function. The difference becomes apparent only when the bus is idle. Because ADV# is high during these periods, external memory will be disabled, thus saving power. Figures 13-16 and 13-17 show sample circuits that use Address Valid Strobe Mode.

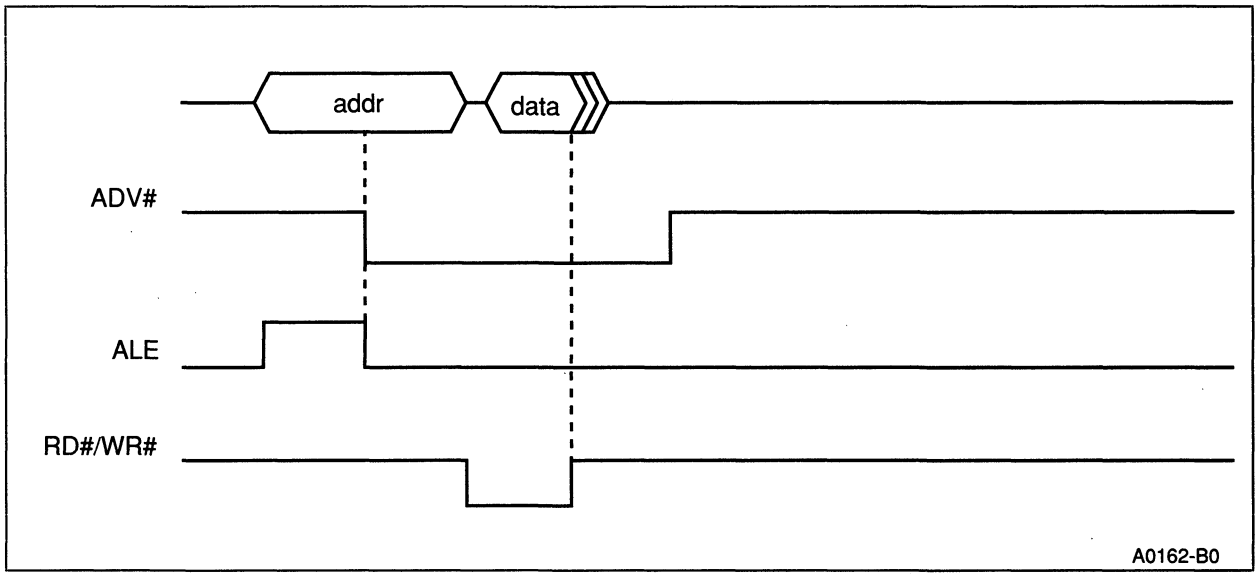


Figure 13-15. Comparison of ALE and ADV# Bus Cycles

Figure 13-13 shows a 16-bit system that uses two EPROMs and two RAMs. This example is configured to use the Write-Strobe mode. ALE latches the address, and A15 is the chip select for the EPROMs and RAMs. WRL# is asserted during low byte writes and word writes. WRH# is asserted during high byte writes and word writes. Note that RAM devices do not use A0. WRL# and WRH# determine whether the low byte (A0 = 0) or high byte (A0 = 1) is selected.

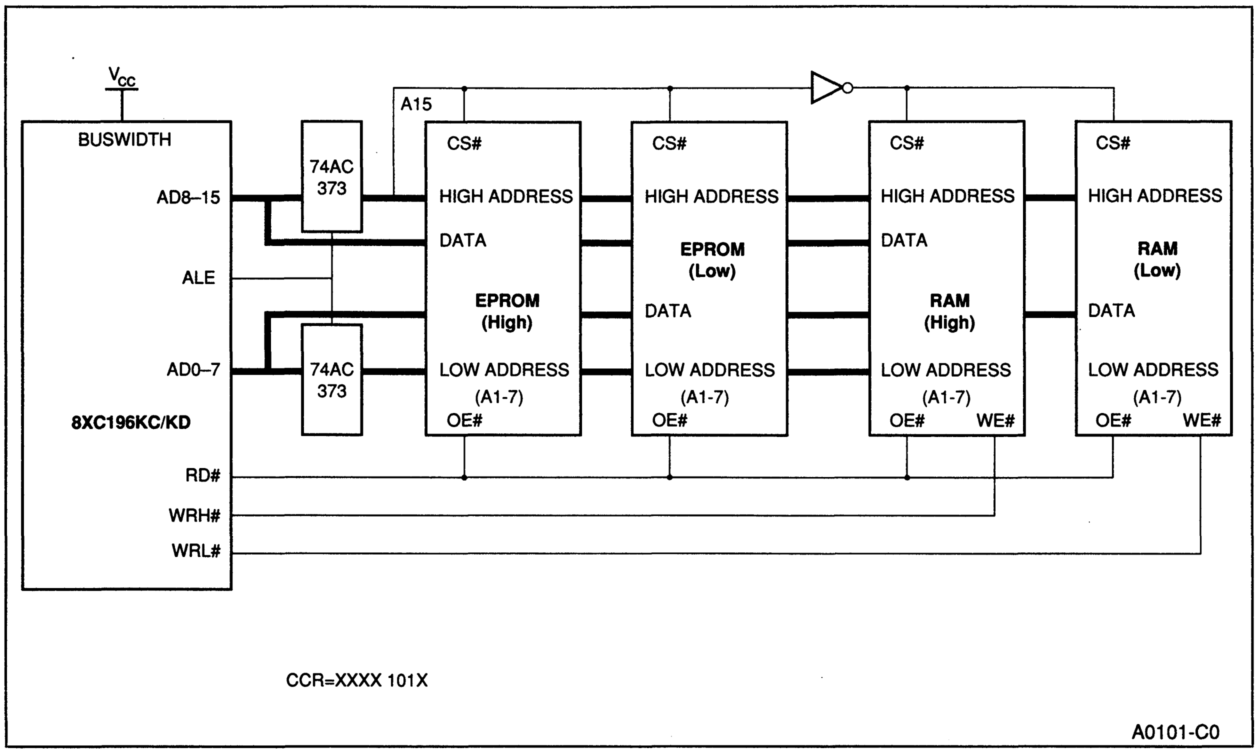


Figure 13-13. 16-Bit System with Single-Byte Writes to RAM

13.6.3. Address Valid Strobe Mode

When the Address Valid Strobe mode is selected, the 8XC196KC/KD generates the Address Valid signal (ADV#) instead of the Address Latch Enable signal (ALE). ADV# is asserted after an external address is valid (see Figure 13-14). This signal can be used to latch the valid address and simultaneously enable an external memory device.

Figure 13-16 shows a simple 8-bit system with a single EPROM. It is configured for the Address Valid Strobe mode. This system configuration uses the ADV# signal as both the EPROM chip-select signal and the address-latch signal.

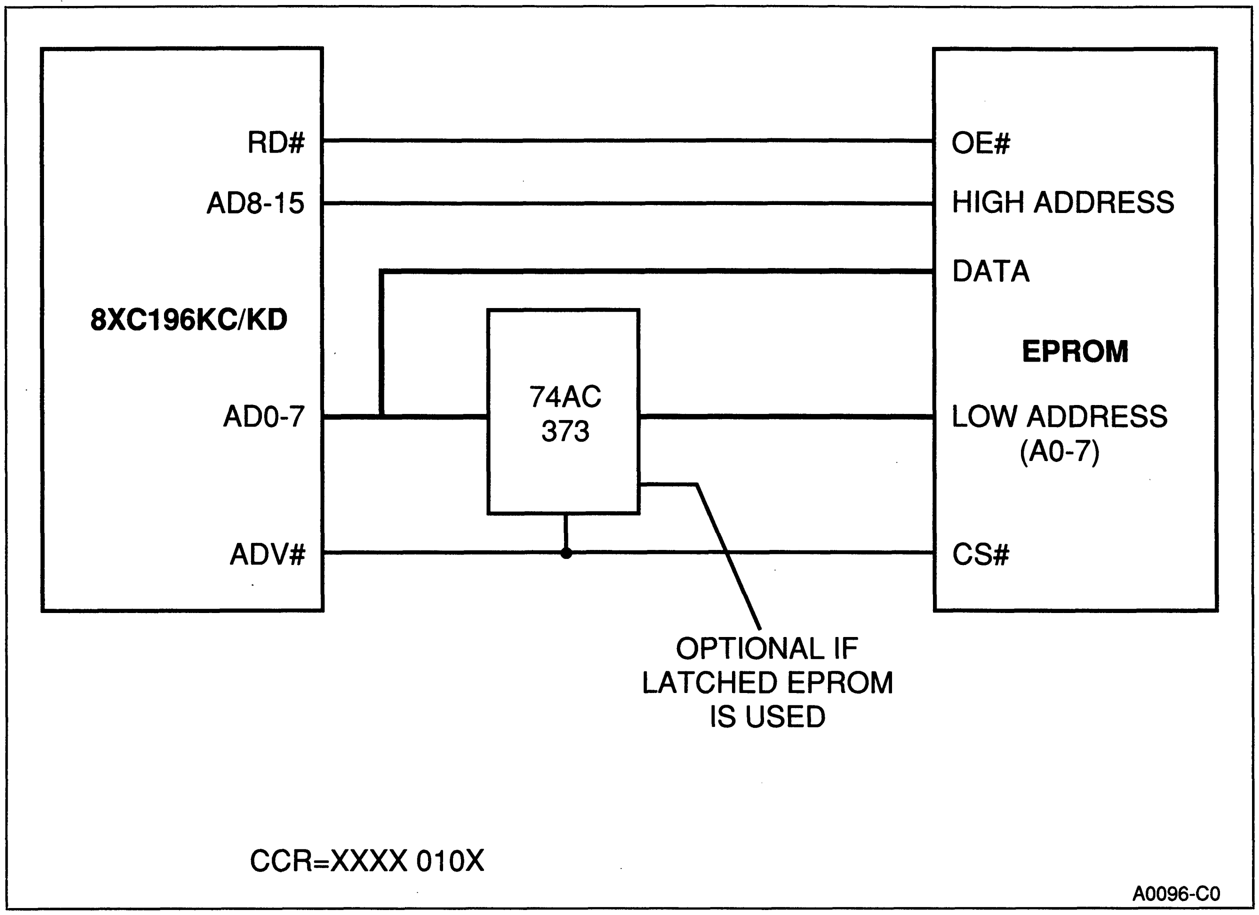


Figure 13-16. 8-Bit System with EPROM

Figure 13-17 shows a 16-bit system with two EPROMs. This system configuration uses the ADV# signal as both the EPROM chip-select signal and the address-latch signal.

13.6.4. Address Valid with Write Strobe Mode

When the Address Valid with Write Strobe mode is selected, the 8XC196KC/KD generates the ADV#, WRL#, and WRH# bus-control signals. This mode is used for a simple system using external 16-bit RAM. Figure 13-18 shows the timing. The RD# signal (not shown) is similar to WRL#, WRH# and WR#. The example system of Figure 13-19 uses Address Valid with Write Strobe.

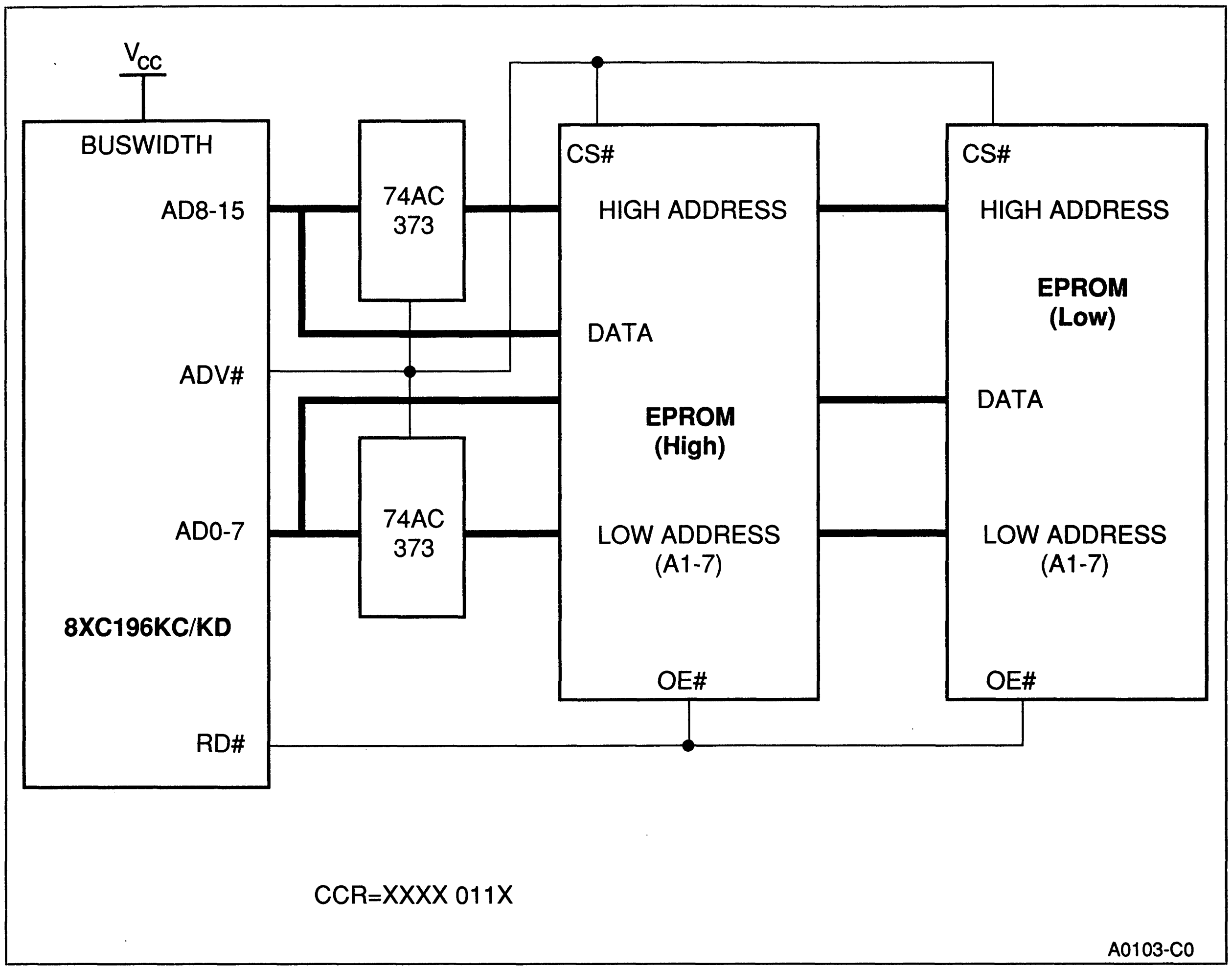


Figure 13-17. 16-Bit System with EPROM

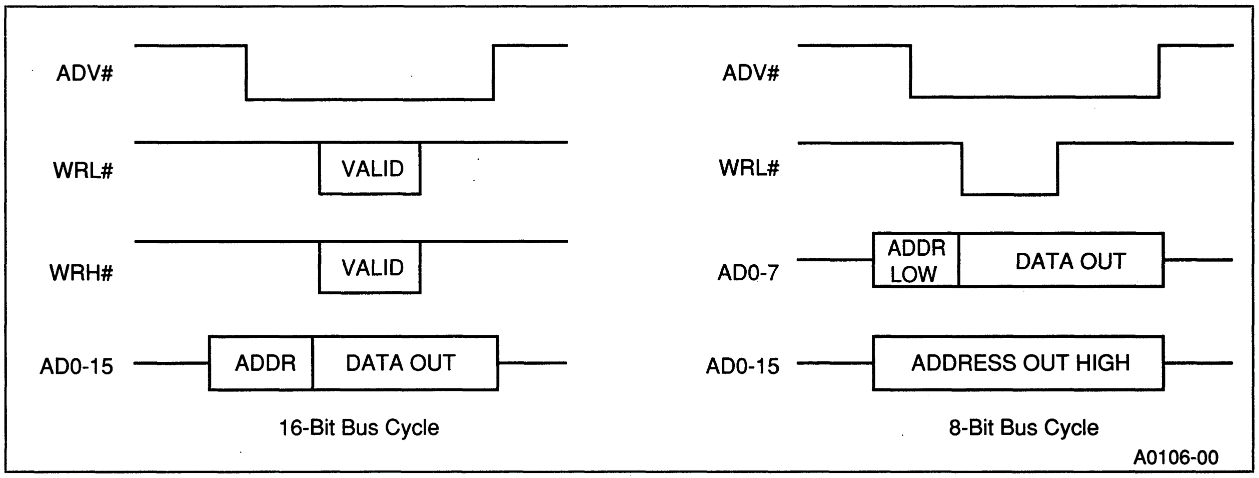


Figure 13-18. Timings of Address Valid with Write Strobe Mode

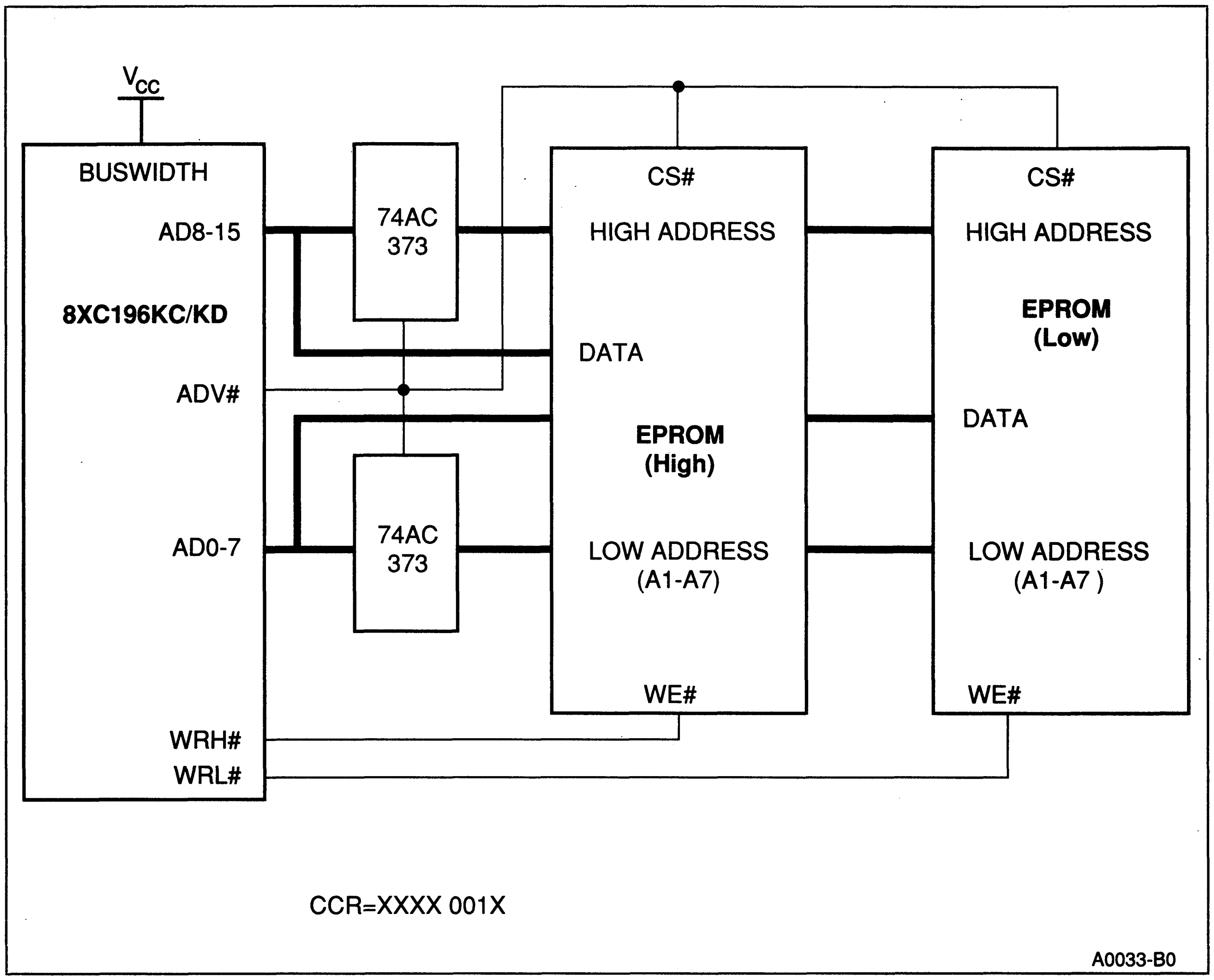


Figure 13-19. 16-Bit System with RAM

13.7. SYSTEM BUS AC TIMING SPECIFICATIONS

Refer to the latest data sheet for the AC timings to make sure your system meets specifications. The major system bus timing specifications are shown in Figure 13-20 and described in Table 13-5.

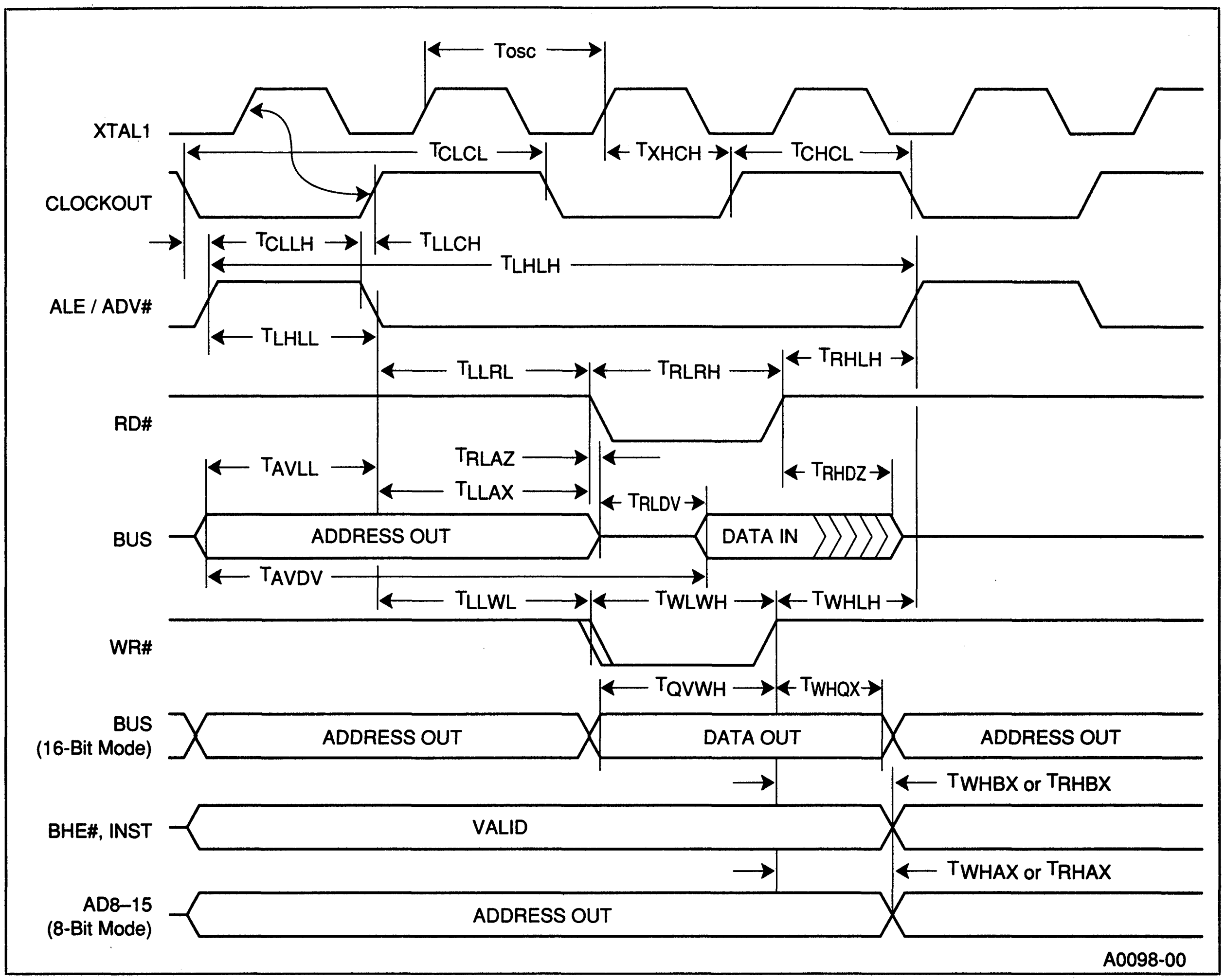


Figure 13-20. System Bus Timing

13.7.1. Explanation of AC Symbols

Each symbol is two pairs of letters prefixed by "T" (for time). The characters in a pair indicate a signal and its condition, respectively. Symbols represent the time between the two signal/condition points.

Conditions:	**Signals:**		
H— High	A— Address	G— BUSWIDTH	Q— Data Out
L— Low	B— BHE#	H— HOLD#	R— RD#
V— Valid	BR— BREQ#	HA— HLDA#	X— XTAL1
X— No Longer Valid	C— CLKOUT	L— ALE/ADV#	Y— READY
Z— Floating	D— DATA	W— WR#/WRH#/WRL#	

Table 13-5. AC Timing Definitions

TIMING THE MEMORY SYSTEM MUST PROVIDE:	
T_{AVYV}	Address Valid to READY Setup: Maximum time the memory system has to assert READY after the address is output by the 8XC196KC/KD to guarantee that at least one wait state will occur.
T_{LLYH}*	ALE Low to READY Setup: Maximum time the memory system has to assert READY after ALE falls to guarantee that at least one wait state will occur.
T_{CLYX}	READY Hold after CLKOUT Low: Minimum hold time is always 0 ns. If maximum specification is exceeded, additional wait states will occur.
T_{LLYX}	READY Hold after ALE Low: Minimum time the level of the READY signal must be valid after ALE falls. If the maximum value is exceeded, additional wait states will occur.
T_{AVGV}	Address Valid to BUSWIDTH Valid: Maximum time after address is valid until BUSWIDTH must be valid. If this specification is exceeded, the 8XC196KC/KD may not respond with the specified bus cycle.
T_{LLGV}	ALE Low to BUSWIDTH Valid: Maximum time after ALE/ADV falls until BUSWIDTH must be valid. If this specification is exceeded, the 8XC196KC/KD may not respond with the specified bus cycle.
T_{CLGX}	BUSWIDTH Hold after CLKOUT Low: Minimum time BUSWIDTH must be held valid after CLKOUT falls. Always 0 ns on the 8XC196KC/KD.
T_{AVDV}	Address Valid to Input Data Valid: Maximum time the memory device has to output valid data after the 8XC196KC/KD outputs valid address.
T_{RLDV}	RD# Low to Input Data Valid: Maximum time the memory system has to output valid data after the 8XC196KC/KD asserts RD#.
T_{CLDV}	CLKOUT Low to Input Data Valid: Maximum time the memory system has to output valid data after CLKOUT falls.
T_{RHDZ}	RD# High to Input Data Float: Time after RD# is inactive until the memory system must float the bus. If this timing is not met, bus contention will occur.
T_{RHDX}	Data Hold after RD# High: Time after RD# is inactive that the memory system must hold data on the bus. Always 0 ns on the 8XC196KC/KD.
F_{XTAL}	Frequency on XTAL: Frequency of the signal input on the XTAL1 input. The internal bus speed of the 8XC196KC/KD device is 1/2 F_{XTAL}.
T_{OSC}	1/F_{XTAL} : All AC Timings are referenced to T_{OSC}.
T_{XHCH}	XTAL1 High to CLKOUT High or Low.
T_{CLCL}	CLKOUT Cycle Time: Normally 2 T_{OSC}.
T_{CHCL}	CLKOUT High Period: Needed in systems which use CLKOUT as clock for external devices.
T_{CLLH}	CLKOUT Falling to ALE/ADV# Rising: Use to derive other timings.
T_{LLCH}	ALE/ADV# Falling to CLKOUT Rising: Use to derive other timings.

* Included only for comparison with HMOS device timings.

Table 13-5 AC Timing Definitions (Continued)

TIMING THE 8XC196KC/KD WILL PROVIDE:	
T_{LHLH}	ALE Cycle Time: Minimum time between ALE pulses.
T_{LHLL}	ALE/ADV# High Period. Use this specification when designing the external latch.
T_{AVLL}	Address Setup to ALE/ADV# Low: Length of time ADDRESS is valid before ALE/ADV# falls. Use this specification when designing the external latch.
T_{LLAX}	Address Hold after ALE/ADV# Low: Length of time ADDRESS is valid after ALE/ADV# falls. Use this specification when designing the external latch.
T_{LLRL}	ALE/ADV# Low to RD# Low: Length of time after ALE/ADV# falls before RD# is asserted. Could be needed to ensure proper memory decoding takes place before a device is enabled.
T_{RLCL}	RD# Low to CLKOUT Low: Length of time from RD# asserted to CLKOUT falling edge.
T_{RLRH}	RD# Low to RD# High: RD# pulse width.
T_{RHLH}	RD# High to ALE/ADV# Asserted: Time between RD# going inactive and the next ALE/ADV#. Useful in calculating time between inactive and next Address valid.
T_{RLAZ}	RD# Low to Address Float: Used to calculate when the 8XC196KC/KD stops driving Address on the bus.
T_{LLWL}	ALE/ADV# Low to WR# Low: Length of time after ALE/ADV# falls before WR# is asserted. Could be needed to ensure proper memory decoding takes place before a device is enabled.
T_{CLWL}	CLKOUT Low to WR# Low: Time between CLKOUT going low and WR# being asserted.
T_{QVWH}	Data Valid to WR# High: Time between data being valid on the bus and WR# going inactive. Memory devices must meet this specification.
T_{CHWH}	CLKOUT High to WR# High: Time between CLKOUT going high and WR# going inactive.
T_{WLWH}	WR# Low to WR# High: WR# pulse width.
T_{WHQX}	Data Hold after WR# High: Length of time after WR# rises that the data stays valid on the bus. Memory devices must meet this specification.
T_{WHLH}	WR# High to ALE/ADV# High: Time between WR# going inactive and next ALE/ADV#. Also used to calculate WR# inactive and next Address valid.
T_{WHBX}	BHE#, INST Hold after WR# High: Minimum time these signals will be valid after WR# inactive.
T_{RHBX}	BHE#, INST Hold after RD# High: Minimum time these signals will be valid after RD# inactive.
T_{WHAX}	AD8–15 Hold after WR# High: Minimum time the high byte of the address in 8-bit mode will be valid after WR# inactive.
T_{RHAX}	AD8–15 Hold after RD# High: Minimum time the high byte of the address in 8-bit mode will be valid after RD# inactive.

Programming the Nonvolatile Memory

14

CHAPTER 14
PROGRAMMING THE NONVOLATILE MEMORY

The 8XC196KC/KD contains One-Time-Programmable Read-Only Memory (OTPROM), a version of EPROM. The 8XC196KC has 16 Kbytes at locations 2000H–5FFFH. The 8XC196KD has 32 Kbytes at locations 2000H–9FFFH. You may program the OTPROM yourself, or have the factory do it for you in the guise of a ROM product.

This chapter contains procedures and guidelines to help you program the device. You have several options. You may use Auto or Slave Programming mode or Run-Time Programming to help you program the OTPROM. Later, you may select ROM-Dump mode to verify the OTPROM contents. You may use the UPROM or PCCB Programming mode to establish part of the security scheme. This chapter describes the programming modes, how to select them, and how to use them.

14.1. SPECIFICATIONS

When you select a programming mode (except Run-Time) the device responds by executing an internal algorithm, which is located in the test ROM. In general, the internal algorithm performs the programming function by prompting the outside world for information.

Your programming-mode selection determines which algorithm the internal test-ROM code executes. To execute programs properly, the device must have these minimum hardware connections: XTAL1 driven, unused input pins strapped, and power and grounds applied. Follow the data sheet operating-condition specifications.

Later in this chapter, a section about each programming mode discusses specific requirements for that mode. The sections include figures illustrating the circuit diagrams we recommend.

14.2. PROGRAMMING MODES

The 8XC196KC/KD supports three methods of programming the OTPROM program memory: Auto Programming mode, Slave Programming mode, and Run-Time Programming.

- Auto Programming mode enables the 8XC196KC/KD to program itself from an external EPROM, without a specialized programmer. This mode allows you to construct a prototype application-programming circuit.
- Slave Programming mode supports programming with a specialized programmer. While using this programming mode, you can program and verify single or multiple words in the OTPROM.

- Run-Time Programming allows you to program individual OTPROM locations during normal program execution, under complete software control. You do not have to place the device into programming mode to perform this type of programming.

ROM-Dump mode and UPROM and PCCB Programming modes support other aspects of programming the OTPROM.

- ROM-Dump mode allows verification of the OTPROM contents by writing the entire OTPROM array to external memory.
- UPROM Programming mode enables programming of two unerasable bits that prevent the bus controller from fetching external instructions (DEI bit) or external data (DED bit).
- PCCB Programming mode enables programming of the Programming Chip Configuration Byte (PCCB). The PCCB is a non-memory-mapped location in the test ROM that establishes hardware security during programming modes.

14.2.1. Programming Mode Selection

Use the PMODE signal to select a programming mode. Table 14-1 describes the PMODE function and lists the PMODE values and programming modes.

Place the 8XC196KC/KD into programming mode by applying V_{PP} voltage (typically +12.5V) to EA# during the rising edge of RESET#. (The "Power-Up and Power-Down Sequences" section on page 14-6 contains more details.)

Table 14-1. PMODE Description and Values

PMODE Description	PMODE Value	Programming Mode
The PMODE value selects an 8XC196KC/KD programming	0H–4H	Reserved
mode (one of five, not including Run-Time Programming).	5H	Slave Programming
Pins P0.4–P0.7 define the hex value of PMODE.	6H	ROM Dump
	7H–8H	Reserved
The five programming modes are Slave, ROM-Dump,	9H	UPROM Programming
UPROM, Auto, and PCCB.	0AH–0BH	Reserved
	0CH	Auto Programming
PMODE is sampled on the rising edge of RESET#.	0DH	PCCB Programming
You must reset the device to switch modes.	0EH–0FH	Reserved

14.3. PROGRAMMING MODE PINS

To support the various programming modes, some device pins have new functions. Table 14-2 describes the new pin functions. Figure 14-1 shows the renamed pins.

Table 14-2. Programming-Mode Pin-Function Description

Function Name	Type	Programming Mode	Description
AINC# P2.4	I	Slave	Auto Increment. During Slave Programming mode, an active-low input enables the auto-increment mode. (Auto increment allows reading or writing of sequential OTPROM locations, without requiring address transactions across the PBUS for each read or write.)
CPVER P2.6	O	Slave	Cumulative Program Verification. During Slave Programming mode, a high signal indicates that all locations programmed correctly.
EA# Programming Mode Select	I	All (except Run-Time)	External Access. Controls program mode entry as follows. If EA# is at V_{PP} voltage on the rising edge of RESET#, the device enters programming mode. EA# is sampled and latched only on the rising edge of RESET#. Changing the level of EA# after reset has no effect.
PACT# P2.7	O	Auto	Programming Active. During Auto Programming mode, a low signal indicates programming is in progress and a high signal indicates programming is complete.
PALE# P2.1	I	Slave	Programming ALE Input. During Slave Programming mode, a falling edge causes the 8XC196KC/KD to read address/command information on Ports 3 and 4.
PBUS PORTS 3, 4	I/O	All (except Run-Time)	Address/Command/Data Bus. Used as a bidirectional port with open-drain outputs to pass commands, addresses, and data to or from the 8XC196KC/KD. Used as a regular system bus to access external memory during ROM-Dump and Auto Programming mode. Slave Programming mode requires pull-ups to V_{CC}.
PMODE P0.4-7	I	All (except Run-Time)	Programming Mode Select. Determines which OTPROM programming algorithm is performed. PMODE is sampled after a chip reset and must be static while the part is operating. (Table 14-1 lists the PMODE values and programming modes.)
PROG# P2.2	I	Slave	Programming. During Slave Programming mode, a falling edge latches data on PBUS and begins programming and a rising edge ends programming.
PVER P2.0	O	Slave Auto	Programming Verification. During Slave or Auto Programming mode, a high output signal indicates successful programming of a location and a low signal indicates a detected error.
V_{PP}	I	All	Programming Voltage. During programming, the V_{PP} pin is typically at +12.5V (V_{PP} voltage). Data sheets list the exact values for different devices. Exceeding the maximum V_{PP} voltage specification may result in device damage.

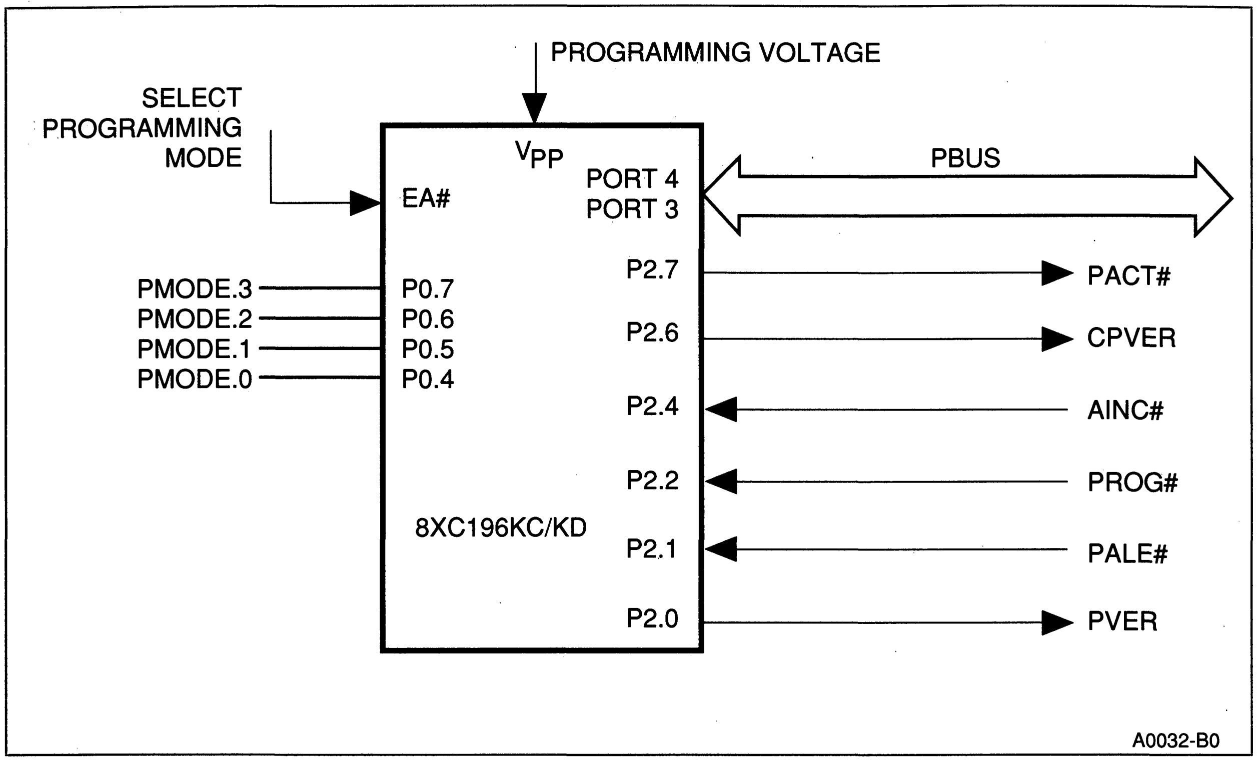


Figure 14-1. Programming-Mode Pin Functions

14.4. MEMORY MAP

The program memory, locations 2080H–5FFFH (8XC196KC) and 2080H–9FFFH (8XC196KD), and the special-purpose memory, locations 2000H–207FH, are user-specified. Figure 14-2 illustrates the special-purpose memory map. Table 14-3 compares the 8XC196KC and 8XC196KD special-purpose memory addresses and provides starting and ending addresses in both hexadecimal and decimal. (Chapter 4, "Memory Partitions," contains a complete set of 8XC196KC/KD memory maps and address tables).

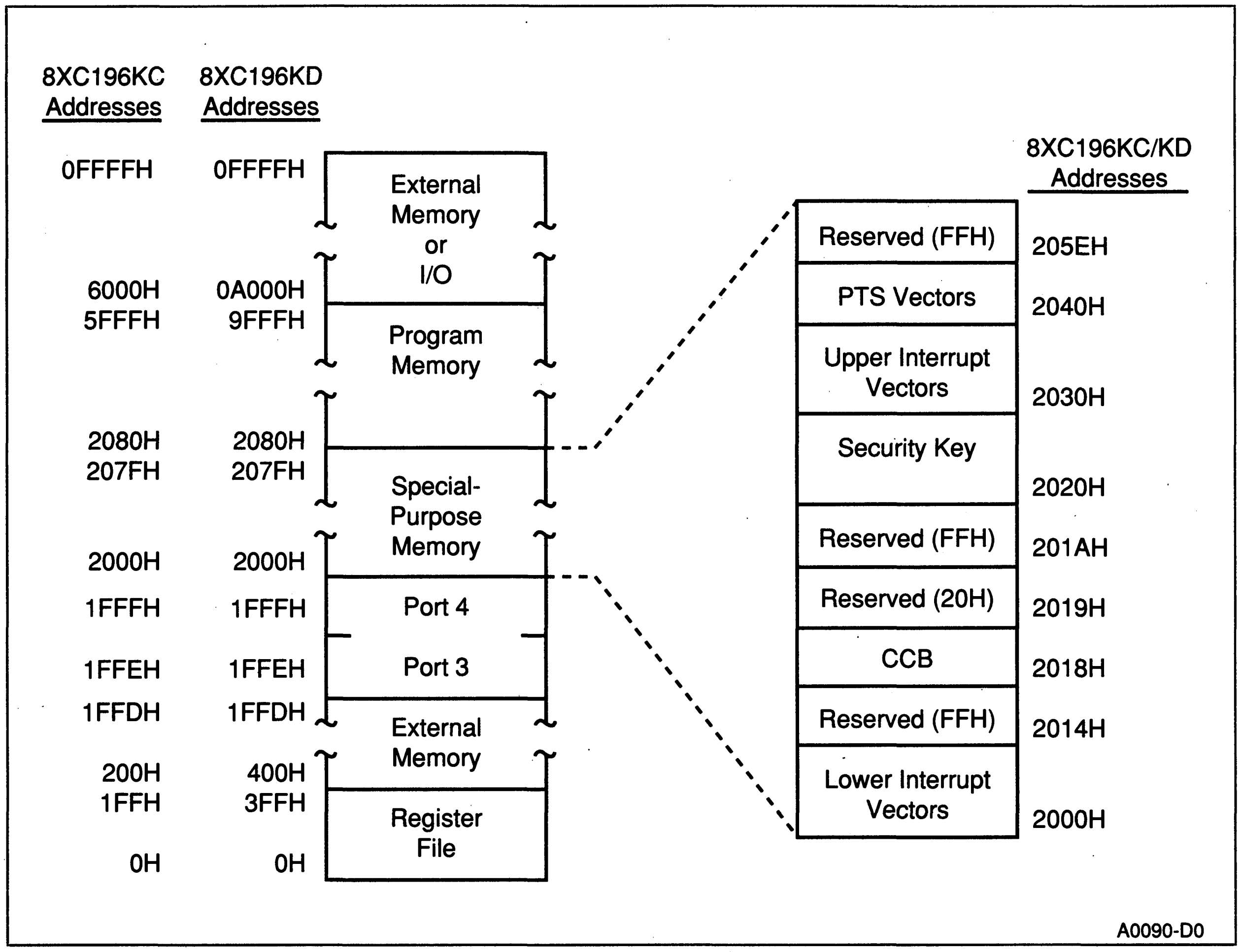


Figure 14-2. 8XC196KC/KD Special-Purpose Memory Map

Table 14-3. 8XC196KC/KD Special-Purpose Memory Addresses

Description	8XC196KC/KD Address Range	
	Hexadecimal	Decimal
Reserved (must contain 0FFH)	205EH–207FH	8286–8319
PTS Vectors	2040H–205DH	8256–8285
Upper Interrupt Vectors	2030H–203FH	8240–8255
Security Key	2020H–202FH	8224–8239
Reserved (must contain 0FFH)	201AH–201FH	8218–8223
Reserved (must contain 20H)	2019H	8217
CCB	2018H	8216
Reserved (must contain 0FFH)	2014H–2017H	8212–8215
Lower Interrupt Vectors	2000H–2013H	8192–8211

14.5. POWER-UP AND POWER-DOWN SEQUENCES

When you are ready to begin programming, follow these procedures to avoid device damage.

WARNINGS

Follow these rules or permanent device damage will result:

- Do not apply voltage to V_{PP} while V_{CC} is low.
- Do not exceed the maximum limit on V_{PP}.
- The power supplies to the V_{CC}, V_{PP}, EA# and RESET# pins must be well regulated and free of glitches and spikes.
- All V_{SS} pins must be well grounded.

14.5.1. Power-Up Sequence

1. Hold the RESET# pin low while V_{CC} stabilizes. Allow V_{PP} and EA# to float during this time.
2. After V_{CC} and the oscillator stabilize, while continuing to hold the device in reset, apply V_{PP} voltage to EA#. After EA# stabilizes, apply V_{PP} voltage to the V_{PP} pin. (Data sheets list the exact V_{PP} voltage for different devices.)
3. Set the PMODE value to select a programming algorithm. (Table 14-1 on page 14-2 identifies the PMODE value for each programming mode.)
4. Bring the RESET# pin high.
5. Complete the selected programming algorithm.

NOTE

The individual Programming Mode sections (later in this chapter) further describe the algorithms.

14.5.2. Power-Down Sequence

1. Assert the RESET# signal (RESET# = 0). RESET# must be held low throughout the power-down sequence.
2. Remove the V_{PP} voltage from the V_{PP} pin and allow the pin to float.
3. Remove the V_{PP} voltage from the EA# pin and allow the pin to float.
4. Turn off the V_{CC} supply and allow time for it to reach 0 volts.

14.6. MEMORY PROTECTION

NOTE

The developers have made a substantial effort to provide an adequate program protection scheme. However, Intel cannot and does not guarantee that these protection methods will always prevent unauthorized access.

Hardware and software protection are each part of the memory protection scheme. The hardware protection prevents OTPROM or external reads or writes in the following manner. Clearing the Chip Configuration Register security-lock bits, CCR.6 (LOC0) and CCR.7 (LOC1), disables OTPROM reads or writes. An attempt to write causes the bus controller to cycle through the write sequence; however, a write does not take place. An attempt to read does not access the internal memory. Setting the UPROM Special Function Register bits, USFR.2 (DED) and USFR.3 (DEI), disables the bus controller from external instruction or data access. If the bus controller attempts an external instruction or data fetch, a device reset occurs.

The software protection prevents unauthorized programming or verifying of the OTPROM. Software support of the memory protection scheme works in the following manner. A software security-key mechanism restricts access to internal memory. If you cleared either CCB.6 or CCB.7 and the device is in programming mode, the programming-mode algorithm requires a security key verification before it executes. If the verification fails, the device enters a program loop that continues until a reset occurs.

14.6.1. CCR Security-Lock Bits

The Chip Configuration Register (CCR) establishes the bus structure and other options of the device during programming. (Chapter 13, "Interfacing with External Memory," and Appendix C, "8XC196KC/KD Registers," contain information about the Chip Configuration Register.) Figure 14-3 illustrates the Chip Configuration Register, including the security-lock bits.

If the 8XC196KC/KD is in normal program execution mode, the reset sequence loads the CCR from location 2018H, the Chip Configuration Byte (CCB). This determines security during normal program execution. Clearing CCB.6 or CCB.7 affects Run-Time Programming and any attempts to access the internal OTPROM from outside the device.

If the device is in programming mode, the reset sequence loads the CCR from a non-memory-mapped test-ROM location (accessible only in programming mode), the Programming Chip Configuration Byte (PCCB). This establishes the hardware security during the programming modes. Clearing these bits affects OTPROM access in all the programming modes. The default value of the PCCB is 0FFH, which disables the hardware security in programming mode.

Figure 14-4 illustrates the CCB and PCCB relationship to the Chip Configuration Register.

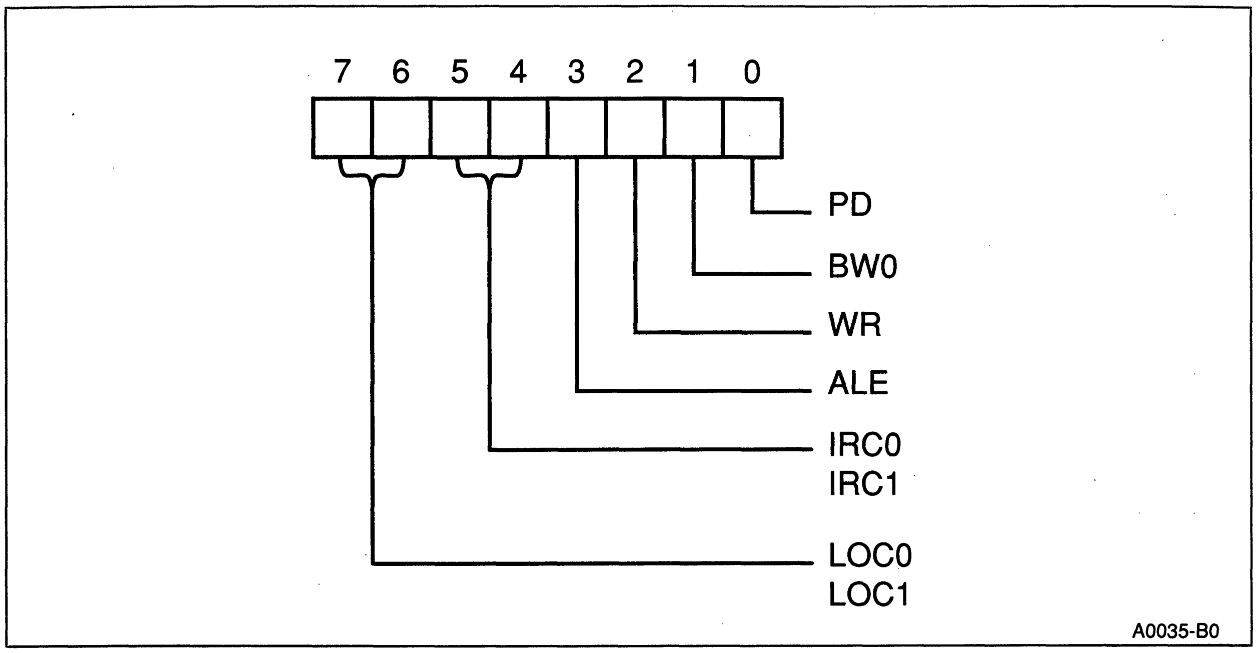


Figure 14-3. Chip Configuration Register

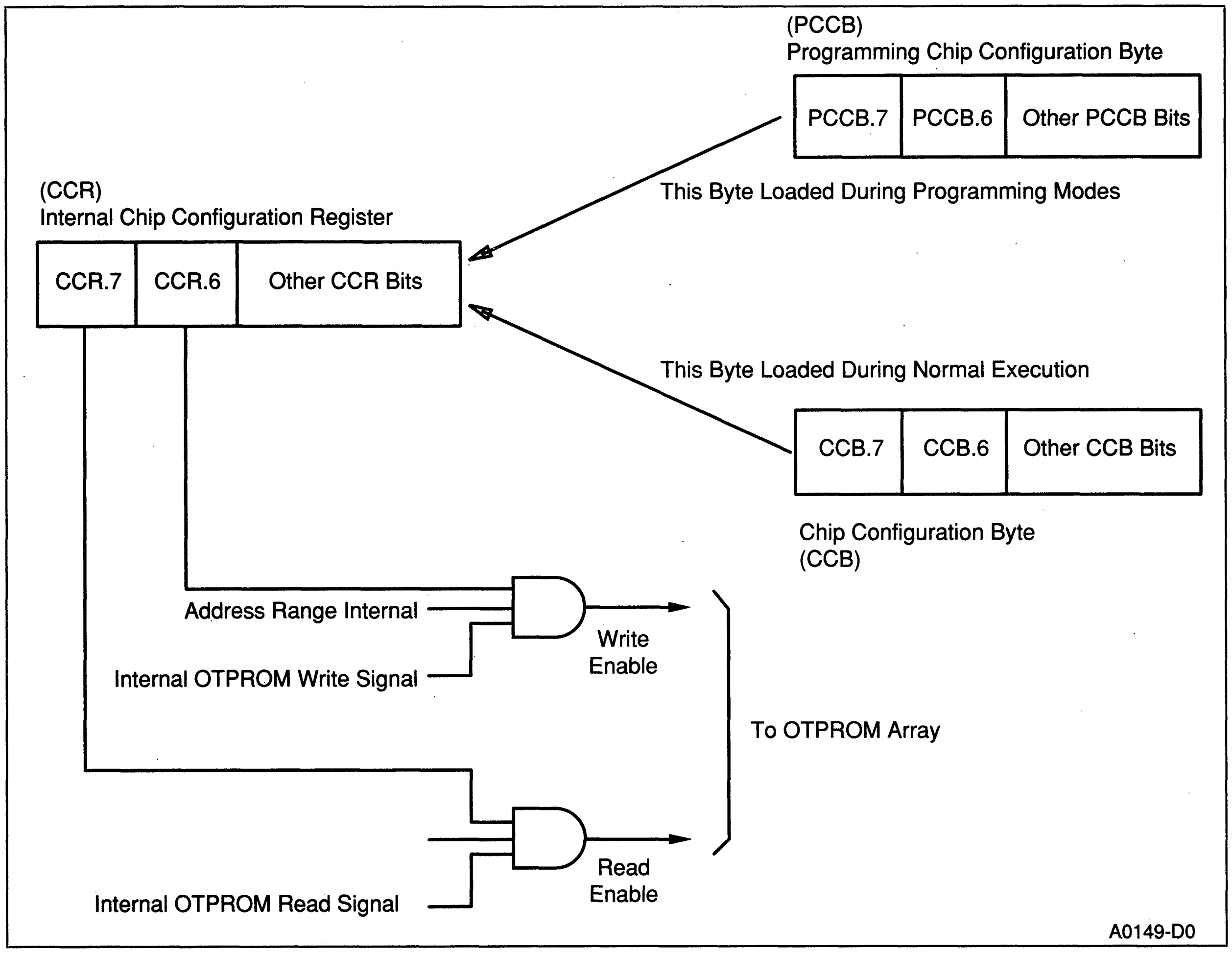

(PCCB)
Programming Chip Configuration Byte
PCCB.7
PCCB.6
Other PCCB Bits
This Byte Loaded During Programming Modes
(CCR)
Internal Chip Configuration Register
CCR.7
CCR.6
Other CCR Bits
This Byte Loaded During Normal Execution
CCB.7
CCB.6
Other CCB Bits
Chip Configuration Byte
(CCB)
Address Range Internal
Internal OTPROM Write Signal
Write Enable
To OTPROM Array
Internal OTPROM Read Signal
Read Enable
A0149-D0

Figure 14-4. Chip Configuration Byte

Clear CCR.6 to disable writing and prevent further programming of the OTPROM array, 2000H–5FFFH (8XC196KC) or 2000H–9FFFH (8XC196KD). An attempt to write causes the bus controller to cycle through the write sequence; however, a write does not take place.

Clear CCR.7 to disable reading and prevent a program in external memory space from accessing the contents of the OTPROM. An attempt to read does not access the internal memory.

A data read can occur if the slave program counter (slave PC) is in the range of 2000–5FFFH (8XC196KC) or 2000–9FFFH (8XC196KD). The slave PC may be as much as 4 bytes ahead of the actual program execution. For this reason, an instruction located after 5FFAH (8XC196KC) or 9FFAH (8XC196KD) may not access protected memory. The "Memory Controller" section of Chapter 2 discusses the slave PC. The interrupt vectors and CCBs are not read protected because interrupts can occur even when executing out of external memory.

If you enable the security-lock bits, CCR.6 or CCR.7, some programming modes require security-key verification before executing and some modes will not execute. See Table 14-4 and each programming section for more information about the effects of enabling the lock bits. Table 14-4 summarizes the protection options.

Table 14-4. Protection Options

PCCB.6	CCB.6	Write Protection Options
1	1	No protection.
1	0	Run-Time Programming not allowed. Programming modes allowed after security-key verification.
0	1	Programming modes not allowed. Run-Time Programming allowed.
0	0	No programming allowed.
PCCB.7	**CCB.7**	**Read Protection Options**
1	1	No protection.
1	0	Programming modes allowed after security-key verification. Reads of internal OTPROM are not allowed if program execution is external.
0	1	Programming allowed.
0	0	Programming modes allowed after security-key verification. Reads of internal OTPROM are not allowed if program execution is external.

14.6.1.1. PCCB PROGRAMMING MODE

You can program the CCB in the same manner that you program any other OTPROM location, using Auto, Slave, or Run-Time Programming mode. You can program the PCCB during Slave Programming or in the PCCB Programming mode. If you need to program only the PCCB, you may use the PCCB Programming mode.

Drive PALE# low to begin PCCB Programming. The algorithm programs the Port 3 data into the PCCB. When programming ends, a verify occurs. The device drives PVER high if the bytes programmed correctly and low if they did not. You can pulse PALE# to repeat programming. Table 14-5 shows the function of the programming-mode signals during PCCB Programming mode.

Table 14-5. PCCB and UPROM Programming Mode Pin Functions

Function Name	Type	Description
PALE# (P2.1)	I	Programming ALE. Clear PALE# to begin reading from Port 3 into the CCB and PCCB.
PVER (P2.0)	O	Program Verification. Updated after each programming pulse is issued. A high signal indicates successful programming and a low signal indicates a detected error.

For PCCB Programming mode, we recommend the circuit illustrated in Figure 14-5.

Enter the PCCB Programming mode by using the standard power-up sequence (page 14-6), after setting the following values:

PMODE = 0DH;

Port 4 = 0FFH;

Port 3 = the data for PCCB.

The algorithm reads the value on Port 3/4 then programs the PCCB by sending 5 or 25 separate 100 μs programming cycles to the appropriate internal location.

Complete the procedure by following the standard power-down sequence (page 14-6).

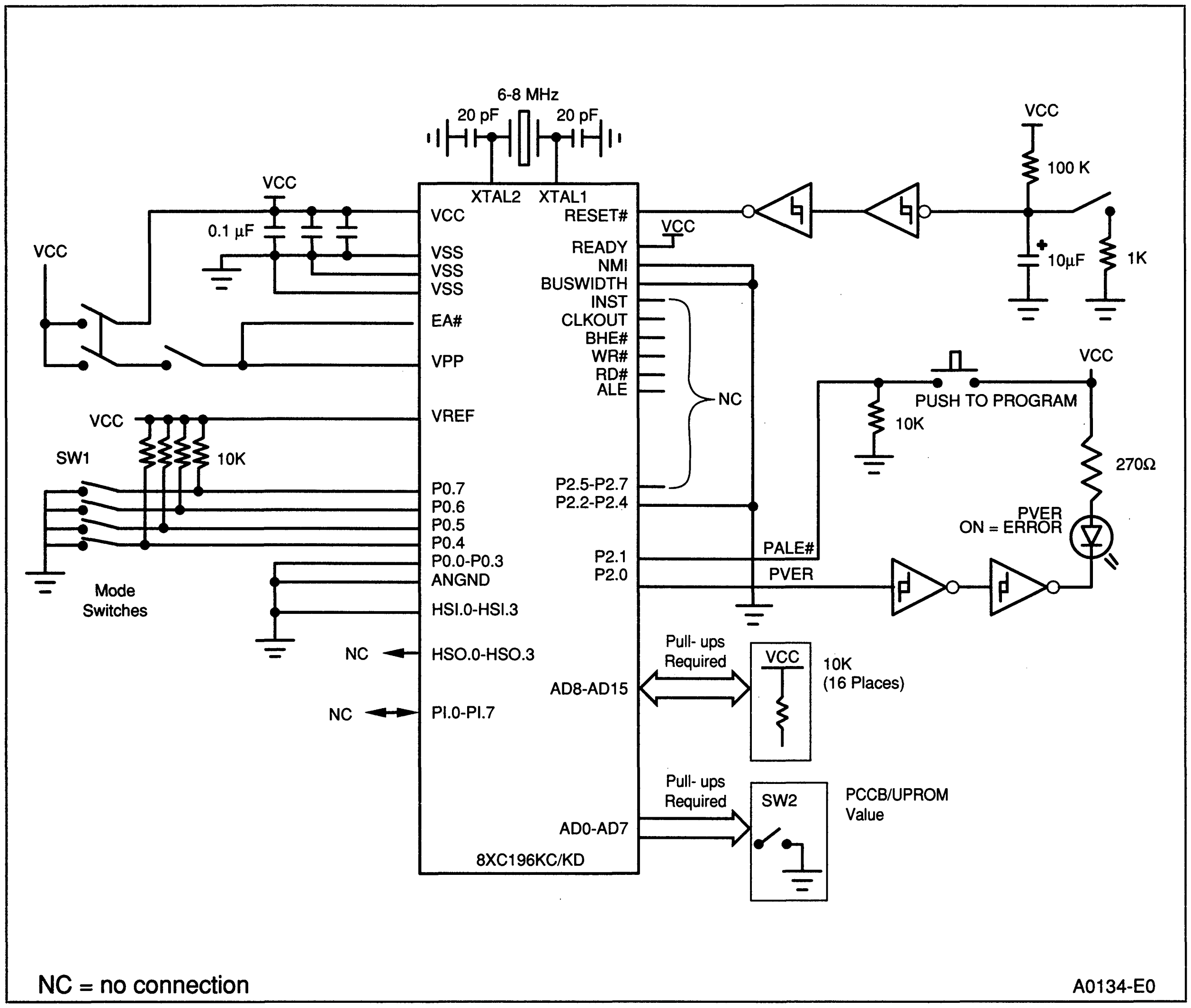


Figure 14-5. PCCB and UPROM Programming Mode Circuit

14.6.2. UPROM Security Bits

The 8XC196KC/KD includes two Unerasable-PROM (UPROM) bits implemented as an additional security feature. Figure 14-6, the UPROM Special Function Register (USFR), illustrates that USFR.2 and USFR.3, Disable External Data fetch and Disable External Instruction fetch (DED and DEI) are the UPROM security bits. Program these bits by writing a one to the appropriate location, using Slave Programming mode or UPROM Programming mode. Figure 14-5 illustrates the circuit we recommend for UPROM Programming mode.

WARNING

Programming these bits makes dynamic failure analysis impossible. For this reason, you cannot return a device to Intel for failure analysis if it has programmed UPROM bits.

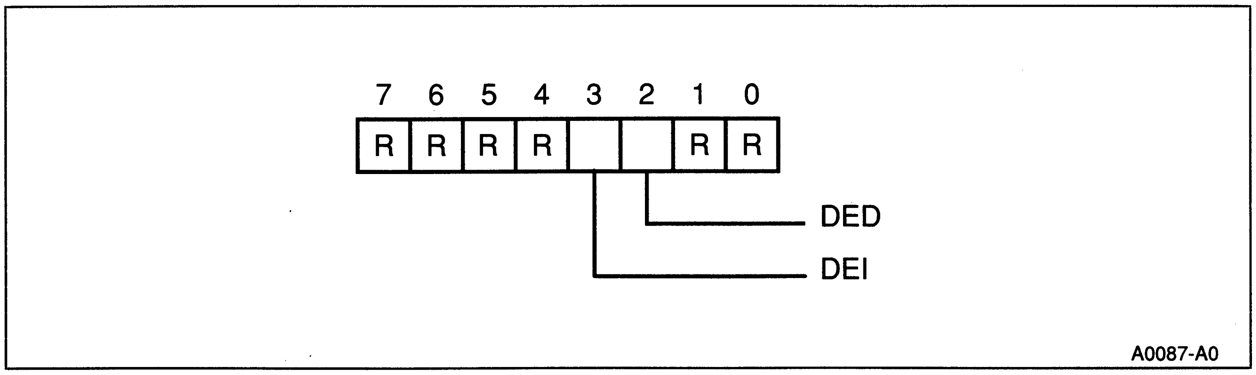


Figure 14-6. UPROM Special Function Register (USFR)

Table 14-6. UPROM Programming Memory Map

To set this bit	While in this Programming Mode	Write this value	To this location
DEI	Slave	08H	0718H
DEI	UPROM	08H	Port 3
DED	Slave	04H	0758H
DED	UPROM	04H	Port 3
DEI and DED	UPROM	0CH	Port 3

Setting USFR.2 (DEI) prevents the bus controller from executing external instruction fetches. If you enable this feature, the prefetch queue in the bus controller prevents code execution from the last four bytes of internal memory. Any attempt to load the slave program counter with an external address causes the device to reset itself. The automatic reset also gives extra protection against runaway code.

Setting USFR.3 (DED) prevents the bus controller from executing external data reads and writes. Any attempt to access data through the bus controller causes the 8XC196KC/KD to reset itself.

You can verify a UPROM bit to make sure it programmed. You cannot erase UPROM bits. For this reason Intel cannot test the bits before shipment. Intel does test the features enabled by the UPROM bit so the only undetectable defects are those (highly unlikely) defects within the UPROM cells themselves.

14.6.2.1. UPROM PROGRAMMING MODE

You can program the UPROM during Slave Programming or in the UPROM Programming mode. You may use the UPROM Programming mode if you need to program only the UPROM.

Drive PALE# low to begin UPROM Programming. The algorithm programs the Port 3 data into the PCCB. When programming ends, a verify occurs. The device drives PVER high if the bytes programmed correctly and low if they did not. You can pulse PALE# to repeat programming. Table 14-5 shows the function of the programming-mode signals during UPROM Programming mode.

For UPROM Programming mode, we recommend the circuit illustrated in Figure 14-5.

Enter the UPROM Programming mode by using the standard power-up sequence (page 14-6), after setting the following values:

PMODE = 9H;

Port 4 = 0FFH;

Port 3 = the data for UPROM.

The algorithm reads the value on Port 3/4 then programs the UPROM by sending 5 or 25 separate 100 μs programming cycles to the appropriate internal location.

Complete the procedure by following the standard power-down sequence (page 14-6).

14.6.3. Security Key

The security key is a 128-bit number located in internal memory at locations 2020H–202FH. Program a password, your security key, into these locations. The Auto Programming, Slave Programming, and ROM-Dump mode sections contain information about how each mode verifies the key. In each instance, if you do not match the security key the device enters an endless loop and must be reset.

14.6.3.1. PROGRAMMING THE SECURITY KEY

If a glitch or reset occurs during programming of the security key, an unknown security key might accidentally be written, rendering the device inaccessible for further programming. To prevent this possibility during Slave Programming, program the rest of the OTPROM array before you program the CCB security-lock bits (CCB.6 and CCB.7).

To prevent the possibility of accidentally writing an unknown security key during Auto Programming, follow these sequences. The Auto programming algorithm skips all locations with a 0FFH value. For this reason, the first sequence programs all of the array except the CCB; the second sequence programs the CCB.

1. Follow these steps to program the entire OTPROM array, except the CCB. (These steps program, but do not enable, the security key.)
   a. Write 0FFH to the external CCB location (4018H = 0FFH).
   b. Place the Programming Pulse Width (PPW) in external location 2014H–2015H.
   c. Place the user code in the appropriate external EPROM locations (see Table 14-8).
   d. Initiate the Auto Programming algorithm, Figure 14-9. The algorithm skips the CCB and programs the rest of the OTPROM array.
2. Follow these steps to program only the CCB (location 2018H). (These steps enable the security key.)
   a. Place the appropriate CCB value in external location 4018H.
   b. Place the Programming Pulse Width (PPW) in external location 2014H–2015H.
   c. Write 0FFH to all other external EPROM locations.
   d. Initiate the Auto Programming algorithm, Figure 14-9. This time the algorithm programs the CCB and skips the rest of the OTPROM array.

NOTE

If you are absolutely sure that a glitch will not occur, you can program the lock bits (CCB.6 and CCB.7) at the same time that you program the security key.

14.7. MODIFIED QUICK-PULSE ALGORITHM

The code that implements the Modified Quick-Pulse Algorithm programs each OTPROM location by sending 5 or 25 separate 100 µs programming cycles to each location. After the 5th (25th) pulse, a verification routine compares the location's contents to the input data. Intel guarantees lifetime data retention for a device programmed with the Modified Quick-Pulse Algorithm.

The Auto, PCCB, and UPROM programming algorithms produce the appropriate number of programming pulses, either 5 (late 8XC196KC and all 8XC196KD devices) or 25 (early 8XC196KC devices).

During Slave Programming mode, the external programmer controls the number of programming pulses at each OTPROM location. Design the external programmer to perform the Modified Quick-Pulse Algorithm as described in Figure 14-7 on page 14-16.

14.8. SIGNATURE WORD

The 8XC196KC/KD contains a Signature Word at location 0070H. The Signature Word identifies device type. The word can be accessed in the Slave Programming mode by executing a Dump-Word command at memory location 0070H. The programming voltages are determined by reading locations 0072H and 0073H in Slave Programming mode. The Signature Word values and voltage levels are shown in Table 14-7. The voltages are calculated by using the following equation.

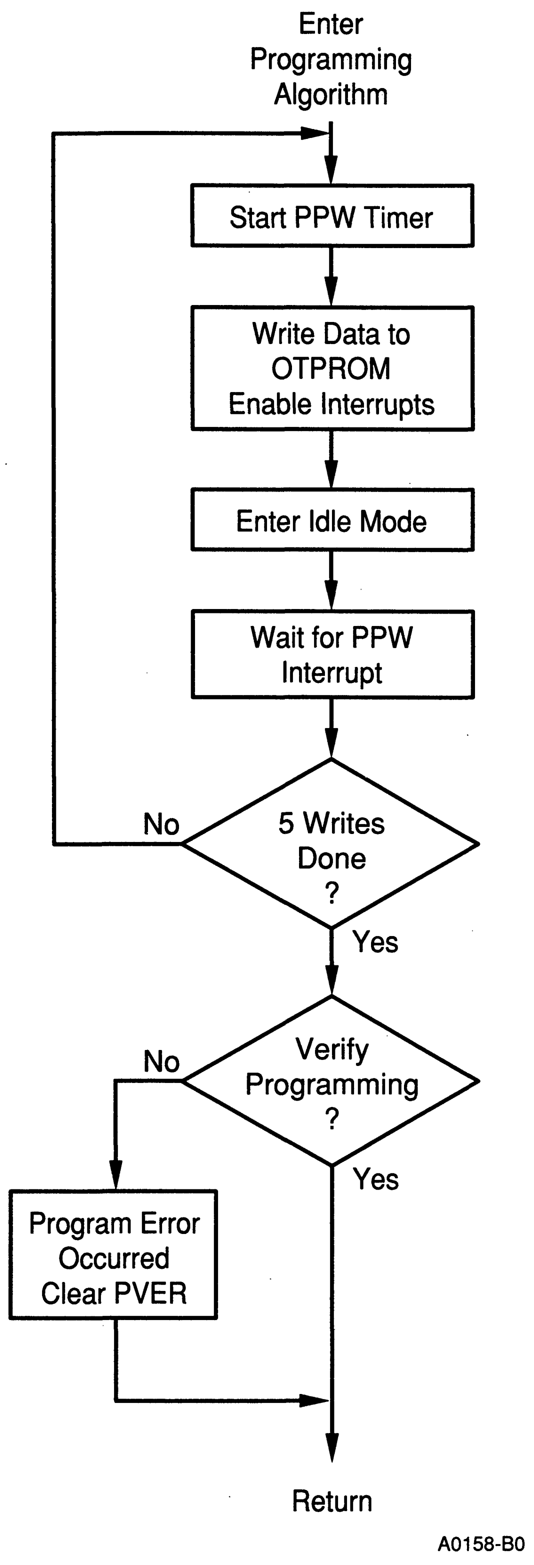
Voltage = 20 / 256 x (test ROM value)

Table 14-7. Signature Word and Voltage Levels

Description	Location*	Early 8XC196KC Value**	Late 8XC196KC Value	8XC196KD Value
Signature Word	0070H	879CH	879CH	879DH
Programming V_{CC}	0072H	0A0H	40H	40H
Programming V_{PP}	0073H	40H	0A0H	0A0H

* Location accessed with a Word Dump during Slave Programming mode.

** An error in the early 8XC196KC device reversed the V_{CC} and V_{PP} bytes.



1. Start the Programming Pulse Width timer.

2. Write the data word to the OTPROM. (Turns on internal programming voltage and begins programming the location.)

3. Enable interrupts and enter Idle mode.

4. Sleep until the PPW causes an interrupt to the CPU that terminates the programming pulse.

5. Read the loop counter. If ≠ 5 (or 25) reiterate. If = 5 (or 25) continue. (Slave Programming: loop counter = 1)

6. Verify programming. If programming verifies, return to the main program. If programming does not verify, clear PVER (P2.0 = low) then return to the main program.

NOTE

A verification error does not stop the programming process.

Figure 14-7. Programming Algorithm

14.9. AUTO PROGRAMMING MODE

The Auto Programming mode is a low-cost programming alternative. Using this programming mode, you can program the 8XC196KC/KD without using an EPROM programmer. The device programs itself with the data from an external EPROM, external locations 4000H–BFFFH (8XC196KD) or 4000H–7FFFH (8XC196KC). The Auto Programming algorithm uses the Programming Pulse Width (PPW) value (discussed on page 14-21) and the Modified Quick-Pulse Algorithm (discussed on page 14-15). It takes approximately 10 seconds to program 32 Kbytes of OTPROM (8XC196KD).

In the Auto Programming mode, the PCCB loads the Chip Configuration Register. The 8XC196KC/KD gets programming data through the external bus, so the PCCB must correctly correspond to the memory system in the programming setup, which is not necessarily the memory system of the application. The "Memory Protection" section beginning on page 14-7 discusses the PCCB.

If the CCB's security-lock bits (CCB.6 and CCB.7) are enabled, a security-key verification occurs at the beginning of the Auto Programming algorithm when you enter the Auto Programming mode. If the verification fails, the device enters an endless internal loop and you must reset. A security-key programming section begins on page 14-14.

14.9.1. Auto Programming Circuit

Figure 14-8 shows the circuit diagram we recommend for Auto Programming mode. (This circuit replaces previously recommended circuits, which did not use P1.0–P1.2 to generate the upper address bits.) This circuit functions with both the 8XC196KC and 8XC196KD devices. It remaps the external EPROM address as shown in Table 14-8.

Table 14-8. Auto Programming Memory Map

Device	External EPROM Address	Internal OTPROM Address	Description
8XC196KC/KD	2014H	N/A	PPW Least-Significant Bit
8XC196KC/KD	2015H	N/A	PPW Most-Significant Bit
8XC196KC	4000H–7FFFH	2000H–5FFFH	Reserved locations for code and data.
8XC196KD	4000H–BFFFH	2000H–9FFFH	Reserved locations for code and data.
8XC196KC/KD	E020H–E02FH	2020H–202FH	Security key, during verification.

The circuit in Figure 14-8 uses an 8-bit external bus (because BUSWIDTH is low). P1.0, P1.1, and P1.2 generate the three high-order bits of the external EPROM address. Hardwire P0.4–P0.7 to V_{SS} and V_{CC} to determine the programming mode. (In this case 1100B = 0CH = Auto Programming mode.) PACT# and PVER are status outputs, buffered by the 74HC14. They drive LEDs that indicate programming active (PACT#) and programming verification (PVER). All unused inputs are connected to ground (V_{SS}), and unused outputs are left floating. READY, NMI, and BUSWIDTH are active; connect them as indicated. Auto programming is specified for a crystal frequency of 6 to 8 MHz. At 8 MHz, a 27(C)512 EPROM with T_{ACC} = 250 ns and T_{OE} = 100 ns or faster specifications should be used. Figure 14-2 and Table 14-3 (see page 14-5) list and describe the device's memory map and addresses.

Table 14-9 shows the function of the programming-mode signals during Auto Programming mode.

Table 14-9. Auto Programming Mode Pin Function

Function Name	Type	Description
PACT# (P2.7)	O	Programming Active. A zero indicates that programming activity is occurring.
PBUS (Ports 3 and 4)	I/O	Address/Command/Data Bus. Used as a regular system bus to access external memory.
PVER (P2.0)	O	Program Verification. Updated after each programming pulse is issued. A high signal indicates successful programming; a low signal indicates a detected error.
P1.0–P1.2	0	Block select lines. Forms the upper address for the external EPROM.

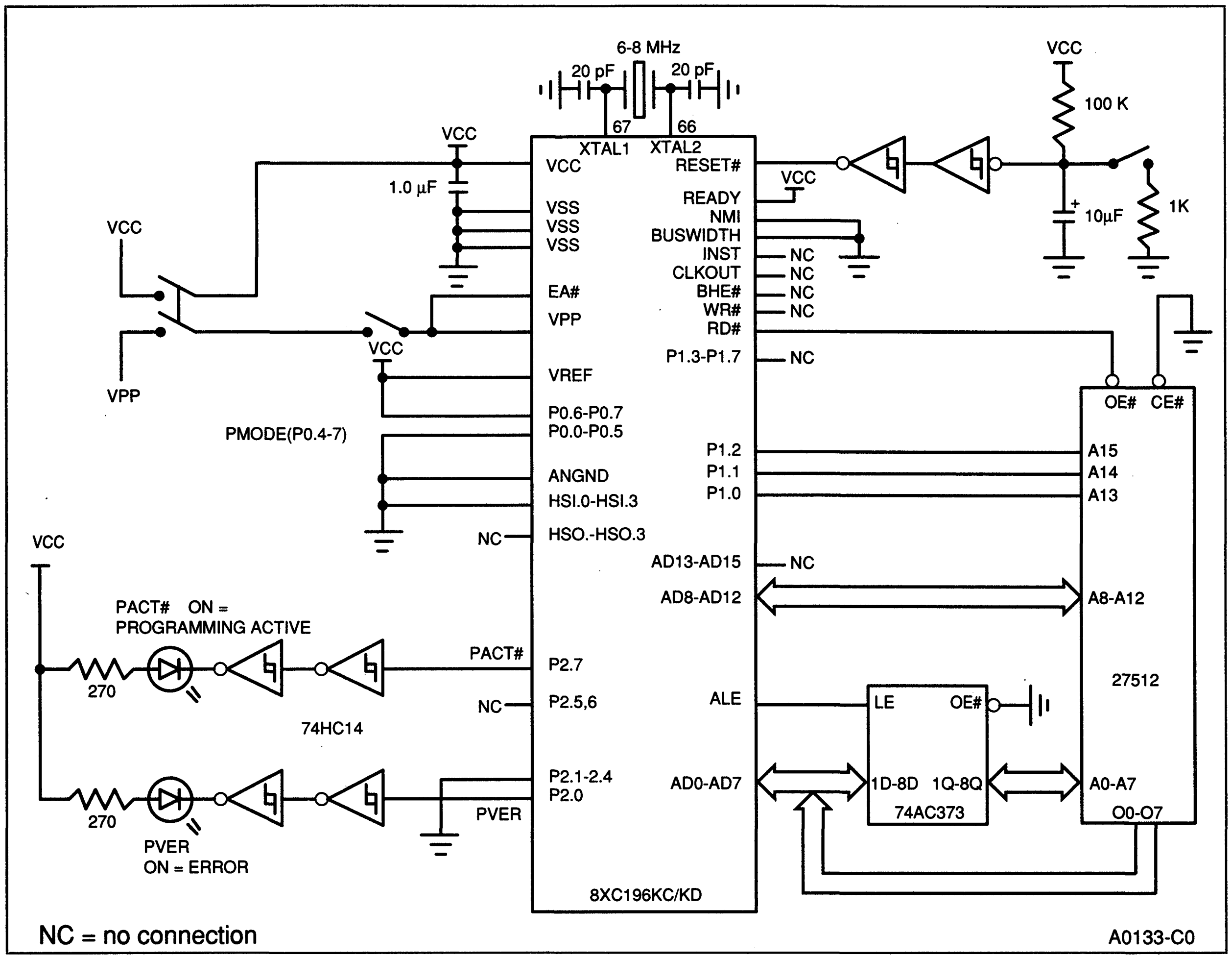


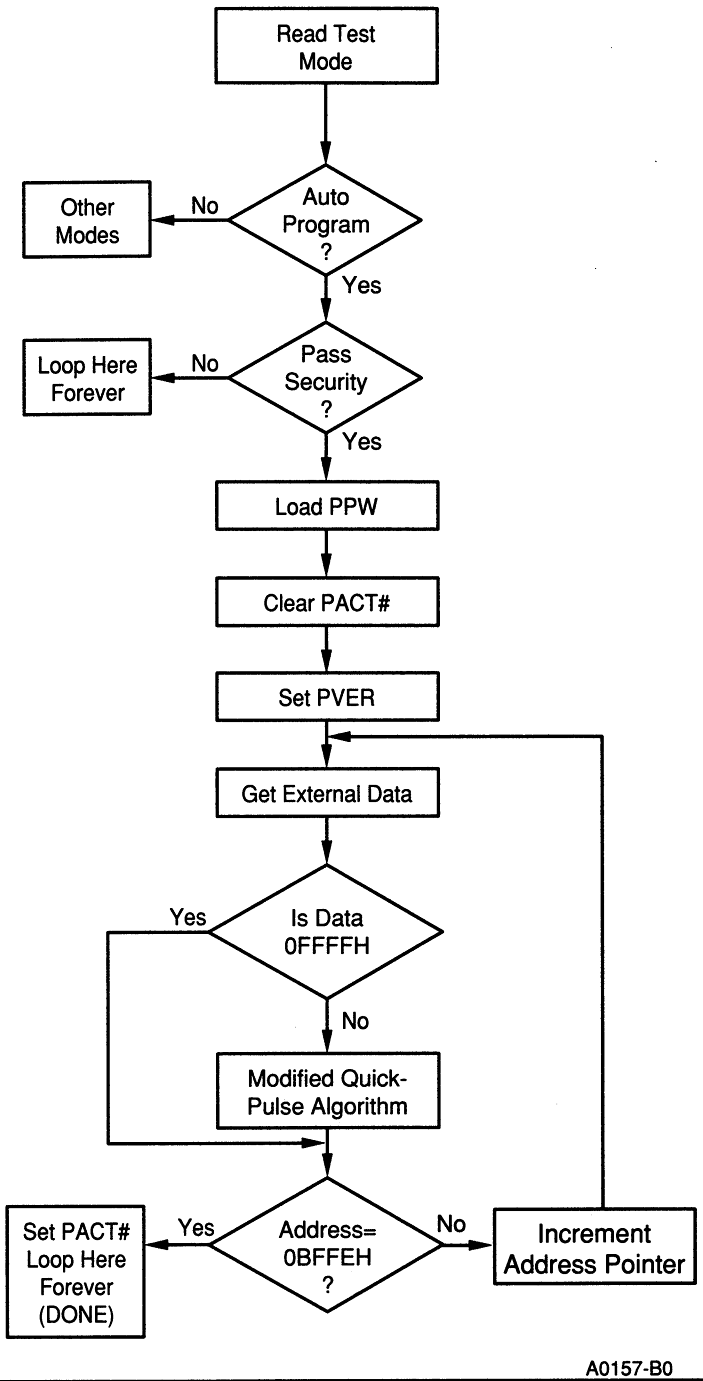
Figure 14-8. Auto Programming Mode Circuit

14.9.2. Verify the Security Key in Auto Programming Mode

If you program the device and clear either CCB.6 or CCB.7 to enable the security feature, the Auto Programming mode algorithm verifies the security key before Auto programming begins. The security-key verification compares external locations E020H–E02FH to internal locations 2020H–202FH. The internal and external locations must match or the device enters an endless loop, preventing programming.

14.9.3. Auto Programming Mode Algorithm

To enter programming mode and initiate the following sequence of events, set PMODE at 0CH and follow the "Power-Up Sequence" described on page 14-6. Figure 14-9 describes the test-ROM program.



1. Determine the programming mode. (Read PMODE.)
2. If Auto Programming mode is selected, continue with this algorithm. (PMODE = 0CH.)
3. If 2020H–202FH is programmed, compare the security key to external locations E020H–E02FH.
4. Load the PPW. (External locations 2014H and 2015H.)
5. Activate PACT# (P2.7 = low).
6. Initialize PVER (P2.0 = high).
7. Read external data, beginning at 4000H.
8. If a word ≠ 0FFFFH, call the Modified Quick-Pulse subroutine.
9. See Figure 14-7.
10. If the address = 0BFFEH, deactivate PACT# (P2.7 = high) and enter an infinite loop. If the address ≠ 0BFFEH, increment address pointer and reiterate programming.

Figure 14-9. Auto Programming Mode Algorithm

14.9.4. Calculating the Programming Pulse Width

The Programming Pulse Width (PPW) register is accessible only during Auto Programming mode; load it from external address 2014H–2015H.

The PPW must equal at least 100 µs for the programmer to function correctly. Use the following formula to calculate the PPW_VALUE. Round the PPW_VALUE to the next higher integer value.

$$\text{PPW\_VALUE} = (0.6944 \times F_{OSC}) - 1$$

where:

PPW_VALUE	is a 15-bit word
F_{OSC}	is the XTAL1 frequency, in MHz

For example, assume XTAL1 is 8 MHz:

$$\begin{aligned}\text{PPW\_VALUE} &= (0.6944 \times 8) - 1\\ &= 5.5552 - 1\\ &= 4.5552\\ &\equiv 5\end{aligned}$$

External memory location 2014H is loaded with the low byte and location 2015H is loaded with the high byte of PPW. Location 2015H usually equals 80H, although it can hold a different value. In every instance, the most-significant bit must equal 1.

The register in Figure 14-10 illustrates the example: most-significant bit (must) = 1, location 2015H = 80H, and location 2014H = 05H.

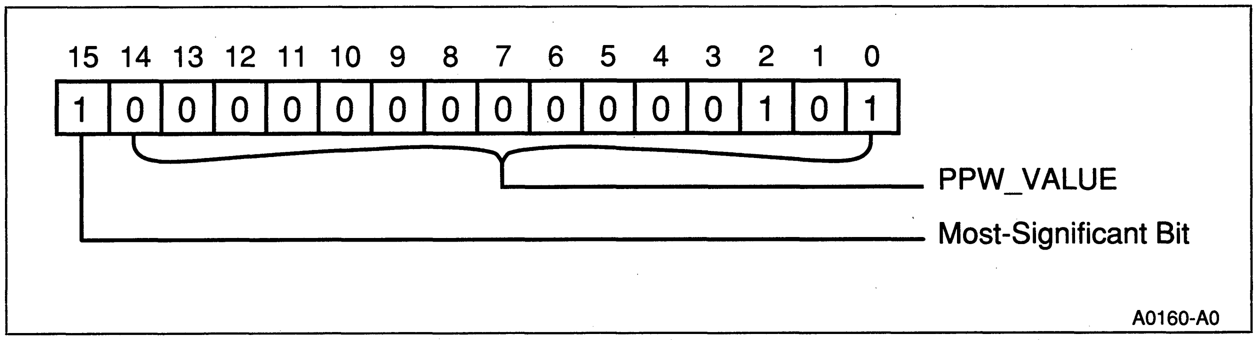


Figure 14-10. Programming Pulse Width Register

Table 14-10. Slave Programming Mode Pin Function

Function Name	Type	Description
AINC# (P2.4)	I	Auto Increment. An active-low signal enables the auto increment mode. Auto increment allows reading from or writing to sequential OTPROM locations without requiring address transactions across the programming bus for each read or write. During programming, AINC# is sampled after each location is programmed. If AINC# is asserted, the address is incremented and the next data word is input. During word dump, AINC# is sampled after each word has been written to Port 3/4. If AINC# is asserted, the address is incremented and the device is ready to output the next data word.
CPVER (P2.6)	O	Cumulative Program Verify. When programming is finished, a high signal indicates that all locations programmed successfully, and a low signal indicates that an error occurred during one of the programming operations.
PALE# (P2.1)	I	Programming Address Latch Enable. Used as the handshake signal for reading address and commands into the device. PALE# is normally held high by the user. When asserted, PALE# causes the device to read an address/command from Ports 3 and 4.
PBUS (Ports 3/4)	I/O	Address/Command/Data Bus. Used to read and write commands, addresses, and data. Ports 3 and 4 are used in their open-drain I/O port modes (not as a system bus). Add external pull-up resistors so you can read data from the device during the Dump Word routine.
PROG# (P2.2)	I	Programming Start. During the Program Word routine, PROG# is used as the handshaking signal for reading programming data into the device. PROG# is normally held high by the user. When asserted, PROG# causes the device to read data from Port 3/4 and program the internal OTPROM. If PROG# remains asserted, that location is programmed with the same data for five 100 µs pulses. For this reason, the data on Port 3/4 must remain stable while PROG# is asserted. When PROG# is deasserted, program flow continues. During Dump Word, PROG# is used as handshaking for outputting data from Port 3/4. PROG# is normally held high by the user. When asserted, PROG# causes the contents of an OTPROM location to output on Port 3/4. When PROG# is deasserted, program flow continues.
PVER (P2.0)	O	Program Verification. Updated after each programming pulse is issued. A high signal indicates successful programming; a low signal indicates a detected error.

The diagram in Figure 14-5 shows the connections we recommend for Slave Programming mode.

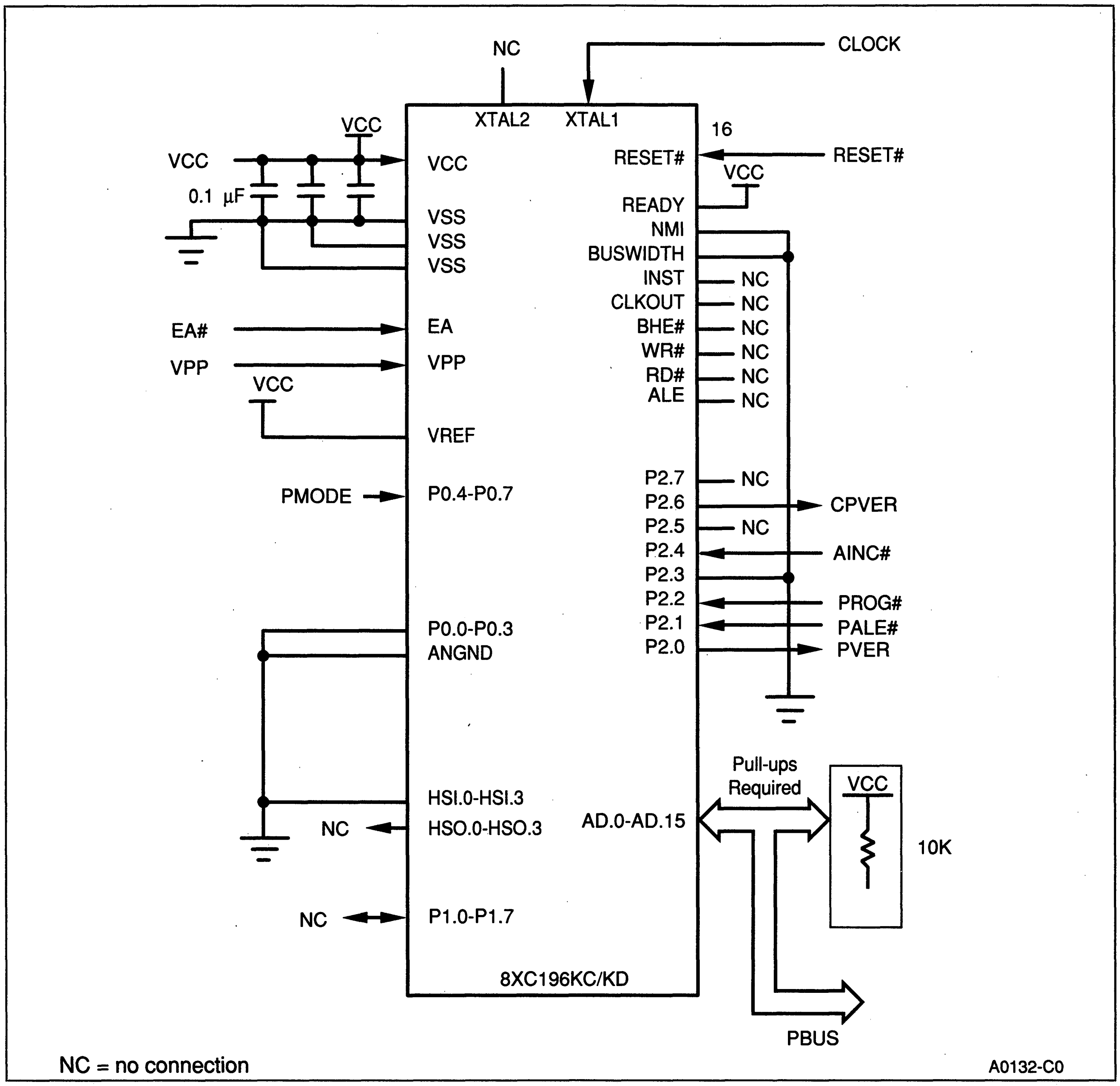
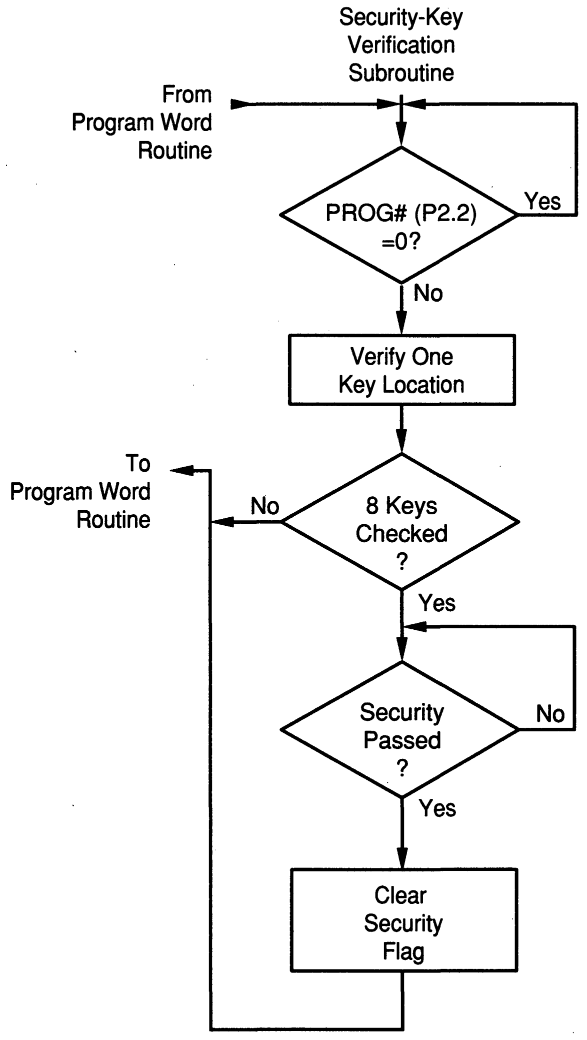


Figure 14-11. Slave Programming Circuit

14.10.2. Verify the Security Key in Slave Programming Mode

The Slave Programming algorithm checks the security-lock bits in the CCB (OTPROM location 2018H) before running the Program Word or Dump Word routine. If the algorithm finds that you cleared either CCB.6 or CCB.7 (write or read protection), programming the security key locations is the next step. Program eight consecutive words, starting at location 2020H and continuing to 202FH. (No programming actually takes place, but the handshaking is identical to that of the Program Word routine.) The algorithm compares the programmed values with internal OTPROM locations 2020H–202FH. If security is passed, access to the device is granted; otherwise, the device hangs up and must be reset. If the verification fails, the device prevents programming or dumping of any location.



1. Wait until PROG# is deasserted.
2. Compare a key location (internal to external).
3. Reiterate until you check eight key locations.
4. If any key location fails verification, the device enters a loop that continues until you reset.
5. If eight key locations pass verification, clear the security flag and return to the Program Word routine.

A0159-B0

Figure 14-12. Security-Key Verification Subroutine

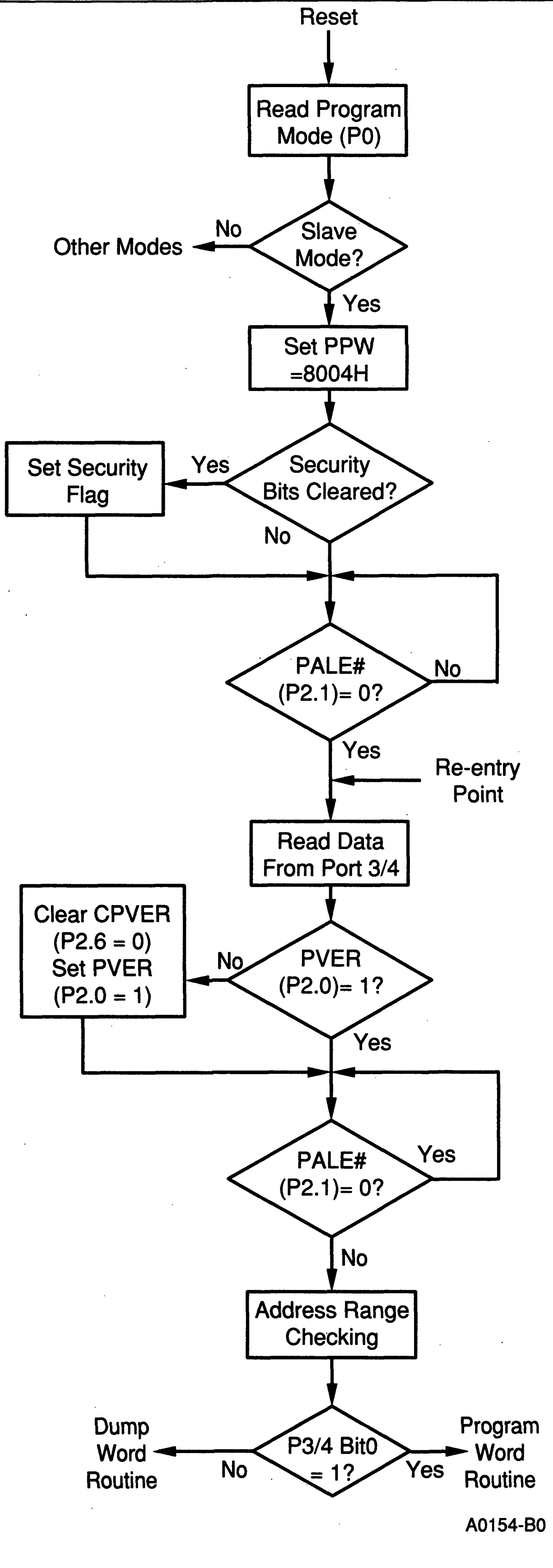
14.10.3. Slave Programming Mode Algorithm

The Power-Up Sequence (page 14-6) and the PMODE Value table (Table 14-1 on page 14-2) provide the information you need to enter Slave Programming mode. The Slave Programming mode algorithm comprises three functional blocks: the Address/Command Decoder, the Program Word routine, and the Dump Word routine.

Figure 14-13 illustrates the Address/Command Decoder routine. In the address/command decoder and initialization block, the algorithm reads PMODE, initializes the PPW register, checks security, and performs handshaking with the PALE# input.

Figure 14-14 illustrates the Program Word routine. This routine programs individual or sequential OTPROM locations. It includes the security-key verification subroutine.

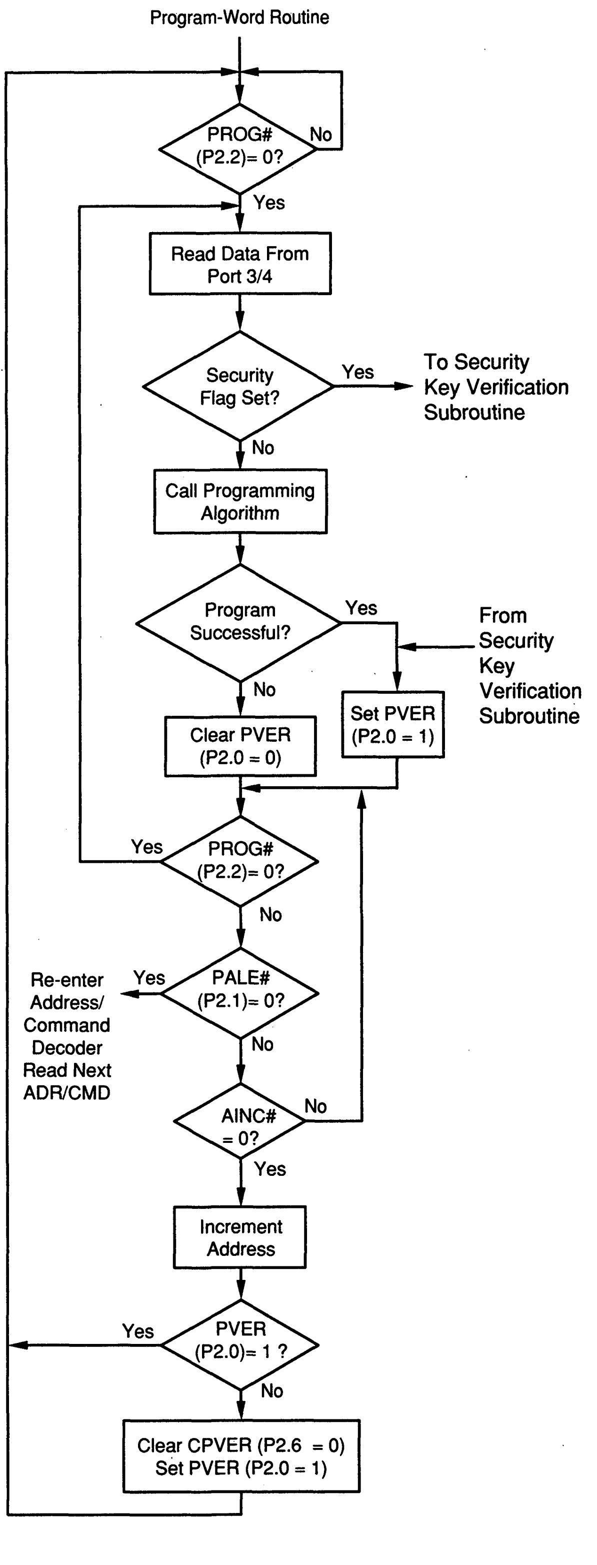
Figure 14-15 illustrates the Dump Word routine. This routine outputs individual or sequential OTPROM locations to Port 3/4.



1. Determine the programming mode. (Read PMODE.)
2. If Slave Programming mode is selected (PMODE = 5H), continue with this algorithm.
3. Initialize the PPW to 8004H.
4. If the CCB.6 or CCB.7 is cleared, set the security flag before going to step 5.
5. If PALE# is asserted, go to the next step; otherwise, reiterate.

   (Re-entry point for the Dump Word or Program Word routine.)
6. Read data from Port 3/4.
7. If PVER is set (no error) go to step 8. If PVER is not set, clear CPVER, set PVER, then go to step 8.
8. Wait until PALE# is deasserted.
9. Check the Port 3/4 address range.
10. If P3.0 = 1, go to the Program Word routine (command mode). If P3.0 = 0, go to the Dump Word routine (address mode). (Use the Port 3/4 word as an address to program to or read from.)

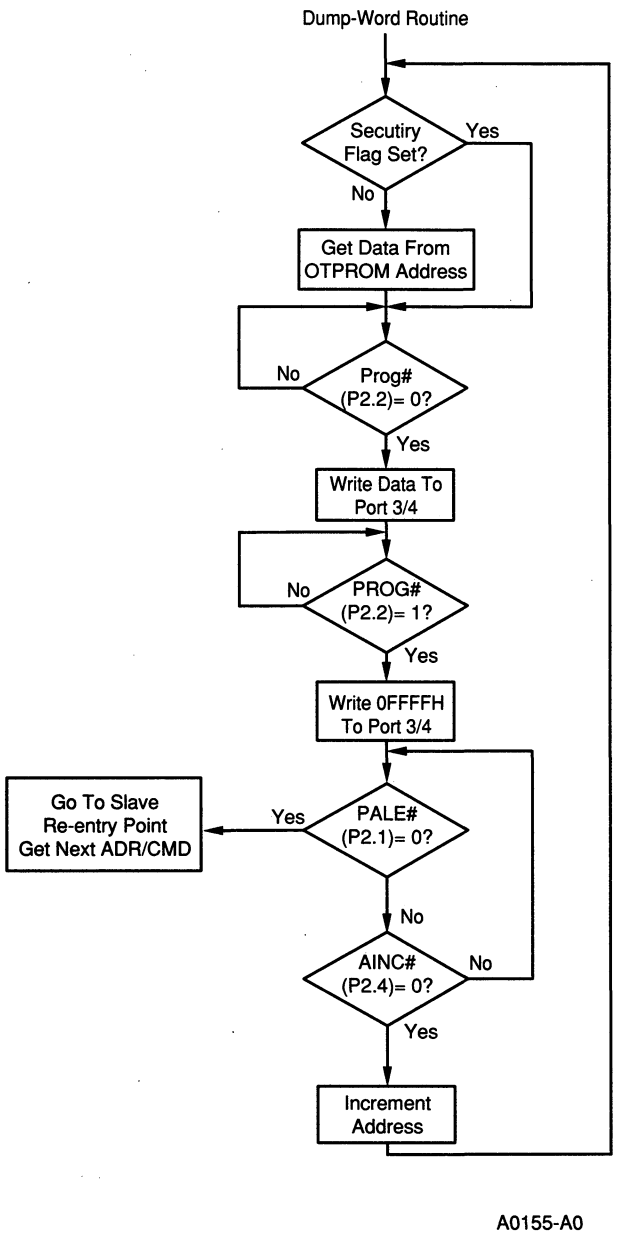
Figure 14-13. Address/Command Decoder Flow



1. Wait until PROG# is asserted.
2. As soon as PROG# is asserted, read the data from Port 3/4.
3. If the security flag is set (see Figure 14-13), go to the Security-Key Verification subroutine. If the security flag is clear, go to step 4.
4. Call the programming algorithm. (See page 14-16.)
5. If programming passes verification, set PVER; if it fails, clear PVER. (The algorithm completes this step for each of the five 100 µs pulses that programs the word.)
6. Hold PROG# asserted until the program loops five times, then deassert PROG#. (An alternative is to pulse PROG# five times while PALE# = 1 and AINC# = 1.)
7. If PALE# is asserted, re-enter the Address/Command Decoder routine and read the next address/command.
8. If PALE# is deasserted, test AINC#.
9. If AINC# is asserted, increment the address by two, then verify.
10. If PVER is asserted (no error occurred) re-enter the Program Word routine.
11. If PVER is not asserted (verification failed), clear CPVER, set PVER, then re-enter the Program Word routine.

A0156-C0

Figure 14-14. Program Word Routine



1. If the security flag is set (see Figure 14-13), skip step 2. If the security flag is not set, go to step 2.

2. Read data from the OTPROM.

3. Wait until PROG# is asserted.

4. As soon as PROG# is asserted, output the data to Port 3/4.

5. Wait until PROG# is deasserted.

6. Write all ones to Port 3/4. (This action allows the open-drain port's pins to float, avoiding conflict with the Program Word and Address/Command Decoder routines, which use Port 3/4 as an input.)

7. If PALE# is asserted, return to the Address/Command-Decoder routine. If PALE# is deasserted, read AINC#.

8. If AINC# is not asserted, return to step 7.

9. If AINC# is asserted, increment the address by two and re-enter this routine.

Figure 14-15. Dump Word Routine

14.10.4. Program Word and Dump Word Commands

Table 14-11 defines the timing mnemonics used in the Program Word and Dump Word waveforms. The 8XC196KC/KD data sheets include timing specifications for these signals.

Table 14-11. Timing Mnemonics

Mnemonic	Description
T_{SHLL}	Reset High to First PALE# Low.
T_{LLLH}	PALE# Pulse Width.
T_{AVLL}	Address Setup Time.
T_{LLAX}	Address Hold Time.
T_{PLDV}	PROG# Low to Word Dump Valid.
T_{PHDX}	Word Dump Data Hold.
T_{DVPL}	Data Setup Time.
T_{PLDX}	Data Hold Time.
T_{PLPH}	PROG# Pulse Width.
T_{PHLL}	PROG# High to Next PALE# Low.
T_{LHPL}	PALE# High to PROG# Low.
T_{PHPL}	PROG# High to Next PROG# Low.
T_{PHIL}	PROG# High to AINC# Low.
T_{ILIH}	AINC# Pulse Width.
T_{ILVH}	PVER Hold After AINC# Low.
T_{ILPL}	AINC# Low to PROG# Low.
T_{PHVL}	PROG# High to PVER Valid.

Figure 14-16 shows the timing of the Program Word command with a repeated programming pulse and auto increment. A one on P3.0 selects the Program Word command. An address of 3501H programs the word location at 3500H. The device receives an input signal, PALE#, to indicate that a valid command is present; asserting PALE# latches the command and address on Ports 3 and 4 (PBUS). Asserting PROG# latches the data on Ports 3 and 4 to start the programming sequence; the device reads in or outputs a word. The PROG# pulse width determines a programming pulse width. (Slave Programming mode does not use the PPW.) After the rising edge of PROG#, the slaves automatically verify the contents of the location just programmed. An asserted PVER indicates successful programming. AINC# is optional and can automatically increment the address for the next location.

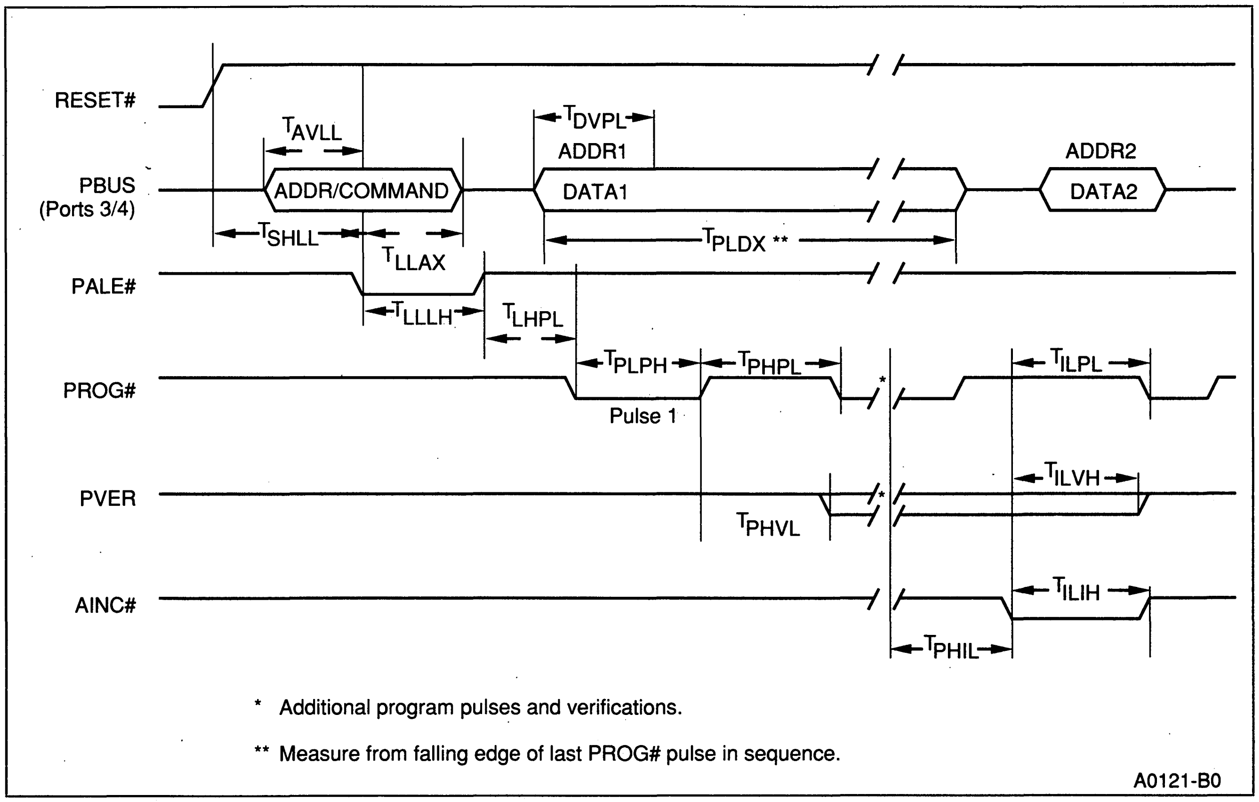


Figure 14-16. Program Word Waveform

Figure 14-17 illustrates the Dump Word command. A zero on P3.0 selects the Program Word command. Sending the command "2100H" results in the slave placing the word at internal address 2100H on Ports 3 and 4. PROG# governs when the device drives the bus. The timings before the Dump Word command are the same as those shown in Figure 14-16. In the Dump Word mode, the AINC# pin can remain active and toggling. The PROG# pin automatically increments the address.

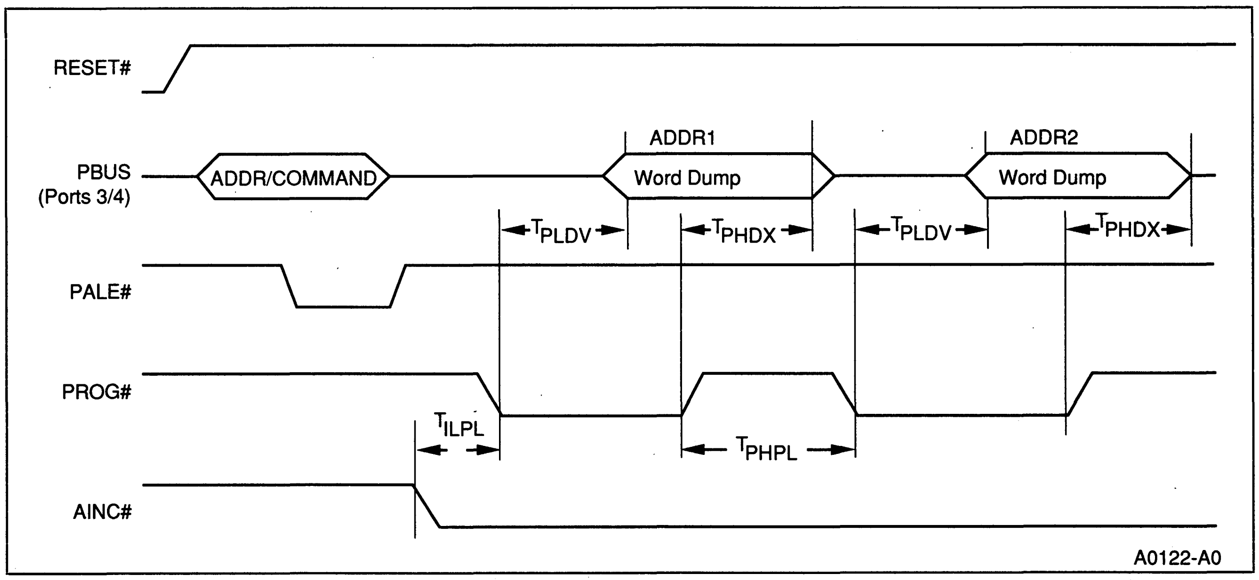


Figure 14-17. Dump Word Waveform

14.11. ROM-DUMP MODE

The ROM-Dump mode is an easy way to verify the contents of the OTPROM array. The Power-Up Sequence (page 14-6) and the PMODE Value table (Table 14-1 on page 14-2) will help you enter ROM-Dump mode.

After verifying the security key, ROM-Dump mode writes the entire OTPROM array to external memory. The security-key verification must find the internal security key duplicated at the corresponding external addresses. If the security key doesn't match, the device enters an endless loop of internal execution, which can be exited only by a chip reset. If USFR.2 (the Disable External Data fetches (DED) bit) has been programmed, ROM-Dump mode is disabled entirely. (The "Memory Protection Options" section provides more information about the security key and UPROM programming.)

Table 14-12. ROM-Dump Mode Memory Map

Device	Internal OTPROM Address	External Memory Address
8XC196KC	2000H–5FFFH	2000H–5FFFH
8XC196KD*	2000H–9FFFH	4000H–BFFFH
8XC196KC*	2000H–5FFFH	4000H–7FFFH
8XC196KC Security Key	2020H–202FH	2020H–202FH
8XC196KD Security Key*	2020H–202FH	E020H–E02FH
8XC196KC Security Key*	2020H–202FH	E020H–E02FH

* Must use bank Switching; P1.0–P1.2 replaces A13–A15.

For devices with 16 Kbytes or less of on-chip OTPROM, the array is dumped to the external memory locations indicated in the table. A bank switching mechanism is provided for devices with more than 16 Kbytes of on-chip OTPROM. The bank switching mechanism must be implemented externally to allow dumps of larger OTPROM arrays, such as an 8XC196KD (32 Kbytes OTPROM). The bank pins (P1.0–P1.2) replace the four high address pins. For instance, if 32 Kbytes are dumped (8XC196KD), they are dumped to external addresses generated by using the bank address pins. However, the real address pins will indicate two consecutive 16 Kbyte dumps. Table 14-13 shows the substitutions for the address bus that must take place before bank switching can operate.

Table 14-13. Bank Switching Mechanism

Bank Pin		High Address Pin
P1.2	replaces	A15
P1.1	replaces	A14
P1.0	replaces	A13

14.12. RUN-TIME PROGRAMMING MODE

You can program an OTPROM location during normal code execution. To make the OTPROM array accessible, hold EA# high while you reset the device. Then simply write to the location to be programmed. Apply V_{PP} voltage to the V_{PP} pin during the entire programming process.

Immediately after writing to the OTPROM, the device must either enter the Idle mode or execute code from external memory; an access to OTPROM would abort the current programming cycle. Each programming cycle begins when the word is written to the OTPROM and ends when the next OTPROM access occurs. Each word requires a total of five programming cycles. The length of each programming cycle must be approximately 100 µs.

Figure 14-18 is a Run-Time Programming code example.

```
Program:     POP ADDRESS_TEMP                          ;load program data
             POP DATA_TEMP                             ;and address
             PUSHF
             LD COUNT, #5t                             ;program using
                                                       ;Modified Quick-
                                                       ;Pulse
LOOP:        LDB INT_MASK, #ENABLE_SWT                 ;program SWT for
             LDB HSO_COMMAND, #SWT0_OVF                ;program pulse width
             ADD HSO_TIME, TIMER1, #PROGRAM_PULSE
             EI
             ST DATA_TEMP, [ADDR_TEMP]                 ;enter idle mode until
             IDLPD 1                                   ;SWT expires
             DJNZ COUNT, LOOP                          ;loop 5 times
             POPF
             RET

SWT_EXPIRED:                                           ;service SWT
             RET                                       ;and return
```

Figure 14-18. Run-Time Programming Code Example

8XC196KC/KD Instruction Set Reference

A

APPENDIX A
8XC196KC/KD INSTRUCTION SET REFERENCE

This appendix provides reference information for the 8XC196KC/KD instruction set. It describes each instruction, shows the relationships between instructions and PSW flags, and shows hexadecimal opcodes, instruction lengths, and execution times.

Table A-1 defines the variables used in Table A-2 to represent instruction operands. Table A-2 lists the instructions alphabetically and describes each of them.

Tables A-3 and A-4 define the abbreviations and symbols used in Tables A-5 and A-6. Table A-5 shows the effect of each instruction on the Program Status Word flags, and Table A-6 shows the effect of the PSW flags or a specified register bit on conditional jump instructions.

Table A-7 lists the instruction opcodes, in hexadecimal order, along with the corresponding instruction mnemonics. Table A-8 is a map of the 8XC196KC/KD opcodes.

Table A-9 lists instruction lengths and opcodes for each applicable addressing mode. Table A-10 lists instruction execution times, expressed in state times. Table A-11 lists execution times, expressed in state times, for PTS cycles.

Table A-1. Operand Variables

Variable	Description
aa	A two-bit field within an opcode that selects the basic addressing mode used. This field is present only in those opcodes that allow address mode options. The field is encoded as follows: 0 0 Register direct 0 1 Immediate 1 0 Indirect 1 1 Indexed
baop	A byte operand that is addressed by any address mode.
bbb	A three-bit field within an opcode that selects a specific bit within a register.
bitno	A three-bit field within an opcode that selects one of the eight bits in a byte.
breg	A byte register in the internal Register File. When it could be unclear whether this variable refers to a source or a destination register, it is prefixed with an *S* or a *D*.
cadd	An address in the program code.
Dbreg *	A byte register in the internal Register File that serves as the destination of the instruction operation.
disp	Displacement. The distance between the end of an instruction and the target label.
Dwreg *	A word register in the internal Register File that serves as the destination of the instruction operation. Must be aligned on an address that is evenly divisible by 2.
lreg	A 32-bit register in the internal Register File. Must be aligned on an address that is evenly divisible by 4.
Sbreg *	A byte register in the internal Register File that serves as the source of the instruction operation.
Swreg *	A byte register in the internal Register File that serves as the source of the instruction operation. Must be aligned on an address that is evenly divisible by 2.
waop	A word operand that is addressed by any address mode.
wreg	A word register in the internal Register File. When it could be unclear whether this variable refers to a source or a destination register, it is prefixed with an *S* or a *D*. Must be aligned on an address that is evenly divisible by 2.
xxx	The three high-order bits of displacement.

* The *D* or *S* prefix is used only when it could be unclear whether a variable refers to a destination or a source register.

Table A-2. Instruction Set

Mnemonic	Operation	Instruction Format
ADD (2 operands)	ADD WORDS. Adds the source and destination word operands and stores the sum into the destination operand. (DEST) ← (DEST) + (SRC)	DEST, SRC ADD wreg, waop (011001aa) (waop) (wreg)
ADD (3 operands)	ADD WORDS. Adds the two source word operands and stores the sum into the destination operand. (DEST) ← (SRC1) + (SRC2)	DEST, SRC1, SRC2 ADD Dwreg, Swreg, waop (010001aa) (waop) (Swreg) (Dwreg)
ADDB (2 operands)	ADD BYTES. Adds the source and destination byte operands and stores the sum into the destination (leftmost) operand. (DEST) ← (DEST) + (SRC)	DEST, SRC ADDB breg, baop (011101aa) (baop) (breg)
ADDB (3 operands)	ADD BYTES. Adds the two source byte operands and stores the sum into the destination operand. (DEST) ← (SRC1) + (SRC2)	DEST, SRC1, SRC2 ADDB Dbreg, Sbreg, baop (010101aa) (baop) (breg)
ADDC	ADD WORDS WITH CARRY. Adds the source and destination word operands and stores the sum and the carry flag (0 or 1) into the destination operand. (DEST) ← (DEST) + (SRC) + C	DEST, SRC ADDC wreg, waop (101001aa) (waop) (wreg)
ADDCB	ADD BYTES WITH CARRY. Adds the source and destination byte operands and stores the sum and the carry flag (0 or 1) into the destination operand. (DEST) ← (DEST) + (SRC) + C	DEST, SRC ADDCB breg, baop (101101aa) (baop) (breg)
AND (2 operands)	LOGICAL AND WORDS. ANDs the source and destination word operands and stores the result into the destination operand. The result has ones in the bit positions in which both operands had a "1" and zeros in all other bit positions. (DEST) ← (DEST) AND (SRC)	DEST, SRC AND wreg, waop (011000aa) (waop) (wreg)
AND (3 operands)	LOGICAL AND WORDS. ANDs the two source word operands and stores the result into the destination operand. The result has ones in only the bit positions in which both operands had a "1" and zeros in all other bit positions. (DEST) ← (SRC1) AND (SRC2)	DEST, SRC1, SRC2 AND Dwreg, Swreg, waop (010000aa) (waop) (Swreg) (Dwreg)
ANDB (2 operands)	LOGICAL AND BYTES. ANDs the source and destination byte operands and stores the result into the destination operand. The result has ones in only the bit positions in which both operands had a "1" and zeros in all other bit positions. (DEST) ← (DEST) AND (SRC)	DEST, SRC ANDB breg, baop (011100aa) (baop) (breg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
ANDB (3 operands)	LOGICAL AND BYTES. ANDs the two source byte operands and stores the result into the destination operand. The result has ones in only the bit positions in which both operands had a "1" and zeros in all other bit positions. (DEST) ← (SRC1) AND (SRC2)	DEST, SRC1, SRC2 ANDB Dbreg, Sbreg, baop (010100aa) (baop) (Sbreg) (Dbreg)
BMOV	BLOCK MOVE. Moves a block of word data from one location in memory to another. The source and destination addresses are calculated using the indirect with auto-increment addressing mode. A long register addresses the source and destination pointers, which are stored in adjacent word registers. A word register (CNTREG) specifies the number of transfers. The blocks of data can reside anywhere in memory, but should not overlap.	DEST, SRC BMOV lreg, wreg (11000001) (wreg) (lreg) **NOTE:** CNTREG is not decremented during this instruction. It is easy to unintentionally create a long, uninterruptable operation with the BMOV instruction. Use the BMOVI instruction for an interruptable operation.
BMOVI	INTERRUPTABLE BLOCK MOVE. Moves a block of word data from one location in memory to another. The instruction is identical to BMOV, except that BMOVI is interruptable. The source and destination addresses are calculated using the indirect with auto-increment addressing mode. A long register addresses the source and destination pointers, which are stored in adjacent word registers. A word register (CNTREG) specifies the number of transfers. The blocks of data can reside anywhere in memory, but should not overlap. COUNT ← (CNTREG) LOOP: SRCPTR ← (PTRS) DSTPTR ← (PTRS + 2) (DSTPTR) ← (SRCPTR) (PTRS) ← SRCPTR + 2 (PTRS + 2) ← DSTPTR + 2 COUNT ← COUNT 1 if COUNT ≠ 0 then go to LOOP	DEST, SRC BMOVI lreg, wreg (11001101) (wreg) (lreg) **NOTE:** CNTREG is not decremented unless the instruction is interrupted. When BMOVI is interrupted, CNTREG is updated to store the interim word count at the time of the interrupt. For this reason, you should always reload CNTREG before starting a BMOVI.
BR	BRANCH INDIRECT. Continues execution at the address specified in the operand word register. PC ← (DEST)	DEST BR [wreg] (11100011) (wreg)
CLR	CLEAR WORD. Clears the value of the operand. (DEST) ← 0	DEST CLR wreg (00000001) (wreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
CLRB	CLEAR BYTE. Clears the value of the operand. (DEST) ← 0	DEST CLRB breg (00010001) (breg)
CLRC	CLEAR CARRY FLAG. Clears the carry flag. C ← 0	CLRC (11111000)
CLRVT	CLEAR OVERFLOW-TRAP FLAG. Clears the overflow-trap flag. VT ← 0	CLRVT (11111100)
CMP	COMPARE WORDS. Subtracts the source word operand from the destination word operand. The flags are altered, but the operands remain unaffected. If a borrow occurs, the carry flag is cleared; otherwise it is set. (DEST) – (SRC)	DEST, SRC CMP wreg, waop (100010aa) (waop) (wreg)
CMPB	COMPARE BYTES. Subtracts the source byte operand from the destination byte operand. The flags are altered, but the operands remain unaffected. If a borrow occurs, the carry flag is cleared; otherwise it is set. (DEST) – (SRC)	DEST, SRC CMPB breg, baop (100110aa) (baop) (breg)
CMPL	COMPARE LONG. Compares the magnitudes of two double-word (long) operands. The operands are specified using the direct addressing mode. The flags are altered, but the operands remain unaffected. If a borrow occurs, the carry flag is cleared; otherwise, it is set. (DEST) – (SRC)	DEST, SRC CMPL lreg, lreg (11000101) (src lreg) (dest#lreg)
DEC	DECREMENT WORD. Decrements the value of the operand by one. (DEST) ← (DEST)–1	DEST DEC wreg (00000101) (wreg)
DECB	DECREMENT BYTE. Decrements the value of the operand by one. (DEST) ← (DEST)–1	DEST DECB breg (00010101) (breg)
DI	DISABLE INTERRUPTS. Disables interrupts. Interrupt-calls cannot occur after this instruction. Interrupt Enable (PSW.1) ← 0	DI (11111010)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
DIV	DIVIDE INTEGERS. Divides the contents of the destination **long-integer** operand by the contents of the source **integer** word operand, using signed arithmetic. It stores the quotient into the low-order word of the destination (i.e., the word with the lower address) and the remainder into the high-order word. (low word DEST) ← (DEST) / (SRC) (high word DEST) ← (DEST) MOD(SRC)	DEST, SRC DIV lreg, waop (11111110) (100011aa) (waop) (lreg)
DIVB	DIVIDE SHORT-INTEGERS. Divides the contents of the destination **integer** operand by the contents of the source **short-integer** operand, using signed arithmetic. It stores the quotient into the low-order word of the destination (i.e., the word with the lower address) and the remainder into the high-order word. (low byte DEST) ← (DEST) / (SRC) (high byte DEST) ← (DEST) MOD (SRC)	DEST, SRC DIVB wreg, baop (11111110) (100111aa) (baop) (wreg)
DIVU	DIVIDE WORDS, UNSIGNED. Divides the content of the destination **double-word** operand by the contents of the source **word** operand, using unsigned arithmetic. It stores the quotient into the low-order word (i.e., the word with the lower address) of the destination operand and the remainder into the high-order word. The following two statements are performed concurrently. (low word DEST) ← (DEST) / (SRC) (high word DEST) ← (DEST) MOD (SRC)	DEST, SRC DIVU lreg, waop (100011aa) (waop) (lreg)
DIVUB	DIVIDE BYTES, UNSIGNED. This instruction divides the contents of the destination **word** operand by the contents of the source **byte** operand, using unsigned arithmetic. It stores the quotient into the low-order word (i.e., the word with the lower address) of the destination operand and the remainder into the high-order word. The following two statements are performed concurrently. (low byte DEST) ← (DEST) / (SRC) (high byte DEST)← (DEST) MOD (SRC)	DEST, SRC DIVUB wreg, baop (100111aa) (baop) (wreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

DJNZ	DECREMENT AND JUMP IF NOT ZERO. Decrements the value of the byte operand by 1. If the result is 0, control passes to the next sequential instruction. If the result is not equal to 0, the instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	(COUNT) ← (COUNT)–1
	if (COUNT) ≠ 0 then
	   PC ← PC + disp (Note 1)
	end_if
		DJNZ breg,cadd
		(11100000) (breg) (disp)

DJNZW	DECREMENT AND JUMP IF NOT ZERO WORD. Decrements the value of the word operand by 1. If the result is 0, control passes to the next sequential instruction. If the result is not equal to 0, the instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127
	
	(COUNT) ← (COUNT)–1
	if (COUNT) ≠ 0 then
	   PC ← PC + disp (Note 1)
	end_if
		DJNZW wreg,cadd
		(11100001) (wreg) (disp)

DPTS	DISABLE PERIPHERAL TRANSACTION SERVER (PTS). Disables the Peripheral Transaction Server (PTS). PTS Disable (PSW.2) ← 0	DPTS (11101100)
EI	ENABLE INTERRUPTS. Enables interrupts following the execution of the next statement. Interrupt-calls cannot occur immediately following this instruction. Interrupt Enable (PSW.1) ← 1	EI (11111011)
EPTS	ENABLE PERIPHERAL TRANSACTION SERVER (PTS). Enables the Peripheral Transaction Server (PTS). PTS Enable (PSW.2) ← 1	EPTS (11101101)

EXT	SIGN-EXTEND INTEGER INTO LONG-INTEGER. Sign-extends the low-order word of the operand throughout the high-order word of the operand.
	
	if (low word DEST) < 8000H then
	   (high word DEST) ← 0
	else
	   (high word DEST) ← 0FFFFH
	end_if
		EXT lreg
		(00000110) (lreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

EXTB	SIGN-EXTEND SHORT-INTEGER INTO INTEGER. Sign-extends the low-order byte of the operand throughout the high-order byte of the operand.
	
	if (low byte DEST) < 80H then
	   (high byte DEST) ← 0
	else
	   (high byte DEST) ← 0FFH
	end_if
		EXTB wreg
		
		(00010110) (wreg)

IDLPD	IDLE/POWERDOWN. Depending on the 8-bit value of the KEY operand, this instruction causes the part
	
	• to enter Idle mode (KEY=1),
	• to enter Powerdown mode (KEY=2),
	• to execute a reset sequence (KEY = any value other than 1 or 2).
	
	The bus controller completes any prefetch cycle in progress before the CPU stops or resets.
	
	if KEY = 1 then
	   enter Idle
	else
	   if KEY = 2 then
	      enter Powerdown
	   else
	      execute reset
		IDLPD #key
		
		(11110110) (key)

INC	INCREMENT WORD. Increments the value of the word operand by 1. (DEST) ← (DEST) + 1	INC wreg (00000111) (wreg)
INCB	INCREMENT BYTE. Increments the value of the byte operand by 1. (DEST) ← (DEST) + 1	INCB breg (00010111) (breg)

JBC	JUMP IF BIT IS CLEAR. Tests the specified bit. If the bit is set, control passes to the next sequential instruction. If the bit is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if (specified bit) = 0 then
	   PC ← PC + disp (Note 1)
		JBC breg,bitno,cadd
		
		(00110bbb) (breg) (disp)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

JBS	JUMP IF BIT IS SET. Tests the specified bit. If the bit is clear, control passes to the next sequential instruction. If the bit is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if (specified bit) = 1 then
	  PC ← PC + disp (Note 1)
		JBS breg,bitno,cadd
		(00111bbb) (breg) (disp)

JC	JUMP IF CARRY FLAG IS SET. Tests the carry flag. If the carry flag is clear, control passes to the next sequential instruction. If the carry flag is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if C = 1 then
	  PC ← PC + disp (Note 1)
		JC cadd
		(11011011) (disp)

JE	JUMP IF EQUAL. Tests the zero flag. If the flag is clear, control passes to the next sequential instruction. If the zero flag is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if Z = 1 then
	  PC ← PC + disp (Note 1)
		JE cadd
		(11011111) (disp)

JGE	JUMP IF SIGNED GREATER THAN OR EQUAL. Tests the negative flag. If the negative flag is set, control passes to the next sequential instruction. If the negative flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if N = 0 then
	  PC ← PC + disp (Note 1)
		JGE cadd
		(11010110) (disp)

JGT	JUMP IF SIGNED GREATER THAN. Tests both the zero flag and the negative flag. If either flag is set, control passes to the next sequential instruction. If both flags are clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if N = 0 AND Z = 0 then
	  PC ← PC + disp (Note 1)
		JGT cadd
		(11010010) (disp)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

JH	JUMP IF HIGHER (UNSIGNED). Tests both the zero flag and the carry flag. If either the carry flag is clear or the zero flag is set, control passes to the next sequential instruction. If the carry flag is set and the zero flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if C = 1 AND Z = 0 then
	  PC ← PC + disp (Note 1)
		JH cadd
		(11011001) (disp)

JLE	JUMP IF SIGNED LESS THAN OR EQUAL. Tests both the negative flag and the zero flag. If both flags are clear, control passes to the next sequential instruction. If either flag is set, this instruction adds to the program counter the offset between the end of this instruction nad the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if N = 1 OR Z = 1 then
	  PC ← PC + disp (Note 1)
		JLE cadd
		(11011010) (disp)

JLT	JUMP IF SIGNED LESS THAN. Tests the negative flag. If the flag is clear, control passes to the next sequential instruction. If the flag is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –128 to +127.
	
	if N = 1 then
	  PC ← PC + disp (Note 1)
		JLT cadd
		(11011110) (disp)

JNC	JUMP IF CARRY FLAG IS CLEAR. Tests the carry flag. If the flag is set, control passes to the next sequential instruction. If the carry flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label. The offset must be in the range of –128 to +127.
	
	if C = 0 then
	  PC ← PC + disp (Note 1)
		JNC cadd
		(11010011) (disp)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

JNE	JUMP IF NOT EQUAL. Tests the zero flag. If the flag is set, control passes to the next sequential instruction. If the zero flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label. The offset must be in the range of –128 to +127.
	
	if Z = 0 then
	PC ← PC + disp (Note 1)
		JNE cadd
		(11010111) (disp)

JNH	JUMP IF NOT HIGHER (UNSIGNED). Tests both the zero flag and the carry flag. If the carry flag is set and the zero flag is clear, control passes to the next sequential instruction. If either the carry flag is set or the zero flag is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if C = 0 OR Z = 1 then
	PC ← PC + disp (Note 1)
		JNH cadd
		(11010001) (disp)

JNST	JUMP IF STICKY BIT FLAG IS CLEAR. Tests the sticky bit flag. If the flag is set, control passes to the next sequential instruction. If the sticky bit flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if ST = 0 then
	PC ← PC + disp (Note 1)
		JNST cadd
		(11010000) (disp)

JNV	JUMP IF OVERFLOW FLAG IS CLEAR. Tests the overflow flag. If the flag is set, control passes to the next sequential instruction. If the overflow flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if V = 0 then
	PC ← PC + disp (Note 1)
		JNV cadd
		(11010101) (disp)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

JNVT	JUMP IF OVERFLOW-TRAP FLAG IS CLEAR. Tests the overflow-trap flag. If the flag is set, this instruction clears the flag and passes control to the next sequential instruction. If the overflow-trap flag is clear, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if VT = 0 then
	  PC ← PC + disp (Note 1)
		JNVT cadd
		
		11010100) (disp)

JST	JUMP IF STICKY BIT FLAG IS SET. Tests the sticky bit flag. If the flag is clear, control passes to the next sequential instruction. If the sticky bit flag is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127..
	
	if ST = 1 then
	  PC ← PC + disp (Note 1)
		JST cadd
		
		(11011000) (disp)

JV	JUMP IF OVERFLOW FLAG IS SET. Tests the overflow flag. If the flag is clear, control passes to the next sequential instruction. If the overflow flag is set, this instruction adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if V = 1 then
	  PC ← PC + disp (Note 1)
		JV cadd
		
		(11011101) (disp)

JVT	JUMP IF OVERFLOW-TRAP FLAG IS SET. Tests the overflow-trap flag. If the flag is clear, control passes to the next sequential instruction. If the overflow-trap flag is set, this instruction clears the flag and adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in range of –128 to +127.
	
	if VT = 1 then
	  PC ← PC + disp (Note 1)
		JVT cadd
		
		(11011100) (disp)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

LCALL	LONG CALL. PUSHes the contents of the program counter (the return address) onto the stack, then adds to the program counter the offset between the end of this instruction and the target label, effecting the call. The operand may be any address in the entire address space.
	
	SP ← SP – 2
	(SP) ← PC
	PC ← PC + disp
		LCALL cadd
		
		(11101111) (disp-low) (disp-high)

LD	LOAD WORD. Loads the value of the source word operand into the destination operand. (DEST) ← (SRC)	DEST, SRC LD wreg, waop (101000aa) (waop) (wreg)
LDB	LOAD BYTE. Loads the value of the source byte operand into the destination operand. (DEST) ← (SRC)	DEST, SRC LDB breg, baop (101100aa) (baop) (breg)

LDBSE	LOAD BYTE SIGN-EXTENDED. Sign-extends the value of the source **short-integer** operand and loads it into the destination **integer** operand.
	
	(low byte DEST) ← (SRC)
	if (SRC) < 80h then
	   (high byte DEST) ← 0
	else
	   (high byte DEST) ← 0FFH
	end_if
		DEST, SRC
		LDBSE wreg, baop
		
		(101111aa) (baop) (wreg)

LDBZE	LOAD BYTE ZERO-EXTENDED. Zero-extends the value of the source **byte** operand and loads it into the destination **word** operand. (low byte DEST) ← (SRC) (high byte DEST) ← 0	DEST, SRC LDBZE wreg, baop (101011aa) (baop) (wreg)
LJMP	LONG JUMP. Adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The operand may be any address in the entire address space. PC ← PC + disp	LJMP cadd (11100111) (disp-low) (disp-high)
MUL (2 operands)	MULTIPLY INTEGERS. Multiplies the source and destination **integer** operands, using signed arithmetic, and stores the 32-bit result into the destination **long-integer** operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (DEST) × (SRC)	DEST, SRC MUL lreg, waop (11111110) (010111aa) (waop) (lreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
MUL (3 operands)	MULTIPLY INTEGERS. Multiplies the two source **integer** operands, using signed arithmetic, and stores the 32-bit result into the destination **long-integer** operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (SRC1) × (SRC2)	DEST, SRC1, SRC2 MUL lreg, wreg, waop (11111110) (010011aa) (waop) (wreg) (lreg)
MULB (2 operands)	MULTIPLY SHORT-INTEGERS. Multiplies the source and destination **short-integer** operands, using signed arithmetic, and stores the 16-bit result into the destination **integer** operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (DEST) × (SRC)	DEST, SRC MULB wreg, baop (11111110) (011111aa) (baop) (wreg)
MULB (3 operands)	MULTIPLY SHORT-INTEGERS. Multiplies the two source **short-integer** operands, using signed arithmetic, and stores the 16-bit result into the destination **integer** operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (SRC1) × (SRC2)	DEST, SRC1, SRC2 MULB wreg, breg, baop (11111110) (010111aa) (baop) (breg) (wreg)
MULU (2 operands)	MULTIPLY WORDS, UNSIGNED. Multiplies the source and destination **word** operands, using unsigned arithmetic, and stores the 32-bit result into the destination **double-word** operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (DEST) × (SRC)	DEST, SRC MULU lreg, waop (011011aa) (waop) (lreg)
MULU (3 operands)	MULTIPLY WORDS, UNSIGNED. Multiplies the two source **word** operands, using unsigned arithmetic, and stores the 32-bit result into the destination **double-word** operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (SRC1) × (SRC2)	DEST, SRC1, SRC2 MULU lreg, wreg, waop (010011aa) (waop) (wreg) (lreg)
MULUB (2 operands)	MULTIPLY BYTES, UNSIGNED. Multiplies the source and destination **byte** operands, using unsigned arithmetic, and stores the **word** result into the destination operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (DEST) × (SRC)	DEST, SRC MULUB wreg, baop (011111aa) (baop) (wreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
MULUB (3 operands)	MULTIPLY BYTES, UNSIGNED. Multiplies the two source **byte** operands, using unsigned arithmetic, and stores the **word** result into the destination operand. The sticky bit flag is undefined after the instruction is executed. (DEST) ← (SRC1) × (SRC2)	DEST, SRC1, SRC2 MULUB wreg, breg, baop (010111aa) (baop) (breg) (wreg)
NEG	NEGATE INTEGER. Negates the value of the **integer** operand. (DEST) ← –(DEST)	NEG wreg (00000011) (wreg)
NEGB	NEGATE SHORT-INTEGER. Negates the value of the **short-integer** operand. (DEST) ← –(DEST)	NEGB breg (00010011) (breg)
NOP	NO OPERATION. Does nothing. Control passes to the next sequential instruction.	NOP (11111101)

NORML	NORMALIZE LONG-INTEGER. Normalizes the source **(leftmost) long-integer** operand. (That is, it shifts the operand to the left until its most significant bit is "1" or until it has performed 31 shifts). If the most significant bit is still "0" after 31 shifts, the instruction stops the process and sets the zero flag. The instruction stores the actual number of shifts performed in the destination **(rightmost)** operand.
	
	(COUNT) ← 0
	do while (MSB(DEST) ← 0) AND (COUNT) < 31)
	   (DEST) ← (DEST) × 2
	   (COUNT) ← (COUNT) + 1
	end_while
		SRC, DEST
		
		NORML lreg, breg
		
		(00001111) (breg) (lreg)

NOT	COMPLEMENT WORD. Complements the value of the word operand (replaces each "1" with a "0" and each "0" with a "1"). (DEST) ← NOT (DEST)	NOT wreg (00000010) (wreg)
NOTB	COMPLEMENT BYTE. Complements the value of the byte operand (replaces each "1" with a "0" and each "0" with a "1"). (DEST) ← NOT (DEST)	NOTB breg (00010010) (breg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
OR	LOGICAL OR WORDS. ORs the source word operand with the destination word operand and replaces the original destination operand with the result. The result has a "1" in each bit position in which either the source or destination operand had a "1". (DEST) ← (DEST) OR (SRC)	DEST, SRC OR wreg, waop (100000aa) (waop) (wreg)
ORB	LOGICAL OR BYTES. ORs the source byte operand with the destination byte operand and replaces the original destination operand with the result. The result has a "1" in each bit position in which either the source or destination operand had a "1". (DEST) ← (DEST) OR (SRC)	DEST, SRC ORB breg, baop (100100aa) (baop) (breg)
POP	POP WORD. Pops the word on top of the stack and places it at the destination operand. (DEST) ← (SP) SP ← SP + 2	POP waop (110011aa) (waop)
POPA	POP ALL. This instruction is used instead of POPF, to support the eight additional interrupts. It pops two words off the stack and places the first word into the INT_MASK1/WSR register and the second word into the PSW/INT_MASK register-pair. This instruction increments the SP by 4. Interrupt-calls cannot occur immediately following this instruction. INT_MASK1/WSR ← (SP) SP ← SP + 2 PSW/INT_MASK ← (SP) SP ← SP + 2	POPA (11110101)
POPF	POP FLAGS. Pops the word on top of the stack and places it into the PSW. Interrupt-calls cannot occur immediately following this instruction. (PSW) ← (SP) SP ← SP + 2	POPF (11110011)
PUSH	PUSH WORD. Pushes the word operand onto the stack. SP ← SP − 2 (SP) ← (DEST)	PUSH waop (110010aa) (waop)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
PUSHA	PUSH ALL. This instruction is used instead of PUSHF, to support the eight additional interrupts. It pushes two words onto the stack — PSW/INT_MASK and the word formed by the INT_MASK1/WSR register-pair. This instruction clears the PSW, INT_MASK, and INT_MASK1 registers and decrements the SP by 4. Interrupt-calls cannot occur immediately following this instruction. SP ← SP 2 (SP) ← PSW/INT_MASK PSW/INT_MASK ← 0 SP ← SP 2 (SP) ← INT_MASK1/WSR INT_MASK1 ← 0	PUSHA (11110100)
PUSHF	PUSH FLAGS. Pushes the PSW onto the top of the stack, then sets it to zeros. This implies that all interrupts are disabled. Interrupt-calls cannot occur immediately following this instruction. SP ← SP – 2 (SP) ← PSW/INT_MASK PSW/INT_MASK ← 0	PUSHF (11110010)

RET	RETURN FROM SUBROUTINE. Pops the PC off the top of the stack.
	
	PC ← (SP)
	SP ← SP + 2
		RET
		(11110000)

RST	RESET SYSTEM. Initializes the PSW to zero, the PC to 2080H, and the SFRs to their initial values. Executing this instruction causes the RESET# pin to be pulled low for 16 state times. SFR Reset Status Pin Reset Status PSW ← 0 PC ← 2080H	RST (11111111)

SCALL	SHORT CALL. Pushes the contents of the program counter (the return address) onto the stack, then adds to the program counter the offset between the end of this instruction and the target label. The offset must be in the range of –1024 to +1023, inclusive.
	
	SP ← SP–2
	(SP) ← PC
	PC ← PC + disp (Note 1)
		SCALL cadd
		(00101xxx) (disp-low)

SETC	SET CARRY FLAG. Sets the carry flag. C ← 1	SETC (11111001)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

SHL	SHIFT WORD LEFT. Shifts the destination word operand to the left as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. The right bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	  C ← High order bit of (DEST)
	  (DEST) ← (DEST) × 2
	  Temp ← Temp – 1
	end_while
		SHL wreg,#count
		(00001001) (count) (wreg)
		or
		SHL wreg,breg
		(00001001) (breg) (wreg)

SHLB	SHIFT BYTE LEFT. Shifts the destination byte operand to the left as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. The right bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	  C ← High order bit of (DEST)
	  (DEST) ← (DEST) × 2
	  Temp ← Temp – 1
	end_while
		SHLB breg,#count
		(00011001) (count) (breg)
		or
		SHLB breg,breg
		(00011001) (breg) (breg)

SHLL	SHIFT DOUBLE-WORD LEFT. Shifts the destination double-word operand to the left as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. The right bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	  C ← High order bit of (DEST)
	  (DEST) ← (DEST) × 2
	  Temp ← Temp – 1
	end_while
		SHLL lreg,#count
		(00001101) (count) (breg)
		or
		SHLL lreg,breg
		(00001101) (breg) (lreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

SHR	LOGICAL RIGHT SHIFT WORD. Shifts the destination word operand to the right as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. The left bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.
	
	This instruction clears the sticky bit flag at the beginning of the instruction. If at any time during the shift a "1" is shifted into the carry flag and another shift cycle occurs, the instruction sets the sticky bit flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	   C ← Low order bit of (DEST)
	   (DEST) ← (DEST)/2 (Note 2)
	   Temp ← Temp – 1
	end_while
		SHR wreg,#count
		(00001000) (count) (wreg)
		or
		SHR wreg,breg
		(00001000) (breg) (wreg)

SHRA	ARITHMETIC RIGHT SHIFT WORD. Shifts the destination word operand to the right as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. If the original high order bit value was "0," zeroes are shifted in. If the value was "1," ones are shifted in. The last bit shifted out is saved in the carry flag.
	
	This instruction clears the sticky bit flag at the beginning of the instruction. If at any time during the shift a "1" is shifted into the carry flag and another shift cycle occurs, the instruction sets the sticky bit flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	   C ← Low order bit of (DEST)
	   (DEST) ← (DEST)/2 (Note 3)
	   Temp ← Temp – 1
	end_while
		SHRA wreg,#count
		(00001010) (count) (wreg)
		or
		SHRA wreg,breg
		(00001010) (breg) (wreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

SHRAB	ARITHMETIC RIGHT SHIFT BYTE. Shifts the destination byte operand to the right as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. If the original high order bit value was "0," zeroes are shifted in. If the value was "1," ones are shifted in. The last bit shifted out is saved in the carry flag.
	
	This instruction clears the sticky bit flag at the beginning of the instruction. If at any time during the shift a "1" is shifted into the carry flag and another shift cycle occurs, the instruction sets the sticky bit flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	   C = Low order bit of (DEST)
	   (DEST) ← (DEST)/2 (Note 3)
	   Temp ← Temp – 1
	end_while
		SHRAB breg,#count
		(00011010) (count) (breg)
		or
		SHRAB breg,breg
		(00011010) (breg) (breg)

SHRAL	ARITHMETIC RIGHT SHIFT DOUBLE-WORD. Shifts the destination double-word operand to the right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. If the original high order bit value was "0," zeroes are shifted in. If the value was "1," ones are shifted in.
	
	This instruction clears the sticky bit flag at the beginning of the instruction. If at any time during the shift a "1" is shifted into the carry flag and another shift cycle occurs, the instruction sets the sticky bit flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	   C ← Low order bit of (DEST)
	   (DEST) ← (DEST)/2 (Note 3)
	   Temp ← Temp – 1
	end_while
		SHRAL lreg,#count
		(00001110) (count) (lreg)
		or
		SHRAL lreg,breg
		(00001110) (breg) (lreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format

SHRB	LOGICAL RIGHT SHIFT BYTE. Shifts the destination byte operand to the right as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. The left bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.
	
	This instruction clears the sticky bit flag at the beginning of the instruction. If at any time during the shift a "1" is shifted into the carry flag and another shift cycle occurs, the instruction sets the sticky bit flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	   C ← Low order bit of (DEST)
	   (DEST) ← (DEST)/2 (Note 2)
	   Temp ← Temp–1
	end_while
		SHRB breg,#count
		(00011000) (count) (breg)
		or
		SHRB breg,breg
		(00011000) (breg) (breg)

SHRL	LOGICAL RIGHT SHIFT DOUBLE-WORD. Shifts the destination double-word operand to the right as many times as specified by the count operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH), inclusive, or as the content of any register with a value in the range of 0 to 31 (1FH), inclusive. The left bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.
	
	This instruction clears the sticky bit flag at the beginning of the instruction. If at any time during the shift a "1" is shifted into the carry flag and another shift cycle occurs, the instruction sets the sticky bit flag.
	
	Temp ← (COUNT)
	do while Temp ≠ 0
	   C ← Low order bit of (DEST)
	   (DEST) ← (DEST)/2 (Note 2)
	   Temp ← Temp – 1
	end_while
		SHRL lreg,#count
		(00001100) (count) (lreg)
		or
		SHRL lreg,breg
		(00001100) (breg) (lreg)

SJMP	SHORT JUMP. Adds to the program counter the offset between the end of this instruction and the target label, effecting the jump. The offset must be in the range of –1024 to +1023, inclusive. PC ← PC + disp (Note 1)	SJMP cadd (00100xxx) (disp-low)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
SKIP	TWO BYTE NO-OPERATION. Does nothing. Control passes to the next sequential instruction. This is actually a two-byte NOP in which the second byte can be any value and is simply ignored.	SKIP breg (00000000) (breg)
ST	STORE WORD. Stores the value of the source **(leftmost)** word operand into the destination **(rightmost)** operand. (DEST) ← (SRC)	**SRC, DEST** ST wreg, waop (110000aa) (waop) (wreg)
STB	STORE BYTE. Stores the value of the source **(leftmost)** byte operand into the destination **(rightmost)** operand. (DEST) ← (SRC)	**SRC, DEST** STB breg, baop (110001aa) (baop) (breg)
SUB (2 operands)	SUBTRACT WORDS. Subtracts the source word operand from the destination word operand, stores the result in the destination operand, and sets the carry flag as the complement of borrow. (DEST) ← (DEST) – (SRC)	DEST, SRC SUB wreg, waop (011010aa) (waop) (wreg)
SUB (3 operands)	SUBTRACT WORDS. Subtracts the first source word operand from the second, stores the result in the destination operand, and sets the carry flag as the complement of borrow. (DEST) ← (SRC1) – (SRC2)	DEST, SRC1, SRC2 SUB Dwreg, Swreg, waop (010010aa) (waop) (Swreg) (Dwreg)
SUBB (2 operands)	SUBTRACT BYTES. Subtracts the source byte operand from the destination byte operand, stores the result in the destination operand, and sets the carry flag as the complement of borrow. (DEST) ← (DEST) – (SRC)	DEST, SRC SUBB breg, baop (010110aa) (baop) (breg)
SUBB (3 operands)	SUBTRACT BYTES. Subtracts the first source byte operand from the second, stores the result in the destination operand, and sets the carry flag as the complement of borrow. (DEST) ← (SRC1) – (SRC2)	DEST, SRC1, SRC2 SUBB Dbreg, Sbreg, baop (010110aa) (baop) (Sbreg) (Dbreg)
SUBC	SUBTRACT WORDS WITH BORROW. Subtracts the source word operand from the destination word operand. If the carry flag was clear, SUBC subtracts 1 from the result. It stores the result in the destination operand and sets the carry flag as the complement of borrow. (DEST) ← (DEST) – (SRC) – (1–C)	DEST, SRC SUBC wreg, waop (101010aa) (waop) (wreg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
SUBCB	SUBTRACT BYTES WITH BORROW. Subtracts the source byte operand from the destination byte operand. If the carry flag was clear, SUBCB subtracts 1 from the result. It stores the result in the destination operand and sets the carry flag as the complement of borrow. (DEST) ← (DEST) – (SRC) – (1–C)	DEST, SRC SUBCB breg, baop (101110aa) (baop) (breg)
TIJMP	TABLE INDIRECT JUMP. Causes execution to continue at an address selected from a table of addresses. The first word register, TBASE, contains the 16-bit address of the beginning of the table. The second word register, INDEX, contains the 16-bit address of a byte that contains the index into the table. The INDEX value must be between 0 and 128. The #byte operand, INDEX_MASK, is 8-bit immediate data to mask INDEX. INDEX_MASK is ANDed with INDEX to determine the offset (OFFSET). OFFSET is multiplied by two, then added to the base address (TBASE) to determine the destination address (DEST X). [INDEX] AND INDEX_MASK = OFFSET (2 × OFFSET) + TBASE = DEST X PC ← (DEST X)	TBASE, INDEX, INDEX_MASK TIJMP wreg, [wreg], #byte (11100010) (wreg) (#byte) [wreg]

TRAP	SOFTWARE TRAP. This instruction causes an interrupt-call that is vectored through location 2010H. The operation of this instruction is not affected by the state of the interrupt enable flag in the PSW (I). Interrupt-calls cannot occur immediately following this instruction.
	
	SP ← SP–2
	(SP) ← PC
	PC ← (2010H)
		TRAP
		
		(11110111)
		
		NOTE: This instruction is not supported by revision 1.2 of the 8096 assembly language. The TRAP instruction is intended for use by Intel-provided tools. These tools will not support user-application of this instruction.

XCH	EXCHANGE WORD. Exchanges the value of the source word operand with that of the destination word operand. (DEST) ← (SRC)	DEST, SRC XCH wreg, waop (00000100) (waop) (wreg) (00001011) (waop) (wreg)
XCHB	EXCHANGE BYTE. Exchanges the value of the source byte operand with that of the destination byte operand. (DEST) ← (SRC)	DEST, SRC XCHB breg, baop (00010100) (baop) (breg) (00011011) (baop) (breg)

Table A-2. Instruction Set (Continued)

Mnemonic	Operation	Instruction Format
XOR	LOGICAL EXCLUSIVE-OR WORDS. XORs the source word operand with the destination word operand and stores the result in the destination operand. The result has ones in the bit positions in which either operand (but not both) had a "1" and zeros in all other bit positions. (DEST) ← (DEST) XOR (SRC)	DEST, SRC XOR wreg, waop (100001aa) (waop) (wreg)
XORB	LOGICAL EXCLUSIVE-OR BYTES. XORs the source byte operand with the destination byte operand and stores the result in the destination operand. The result has ones in the bit positions in which either operand (but not both) had a "1" and zeros in all other bit positions. (DEST) ← (DEST) XOR (SRC)	DEST, SRC XORB breg, baop (100101aa) (baop) (breg)

NOTES:

1. The displacement (disp) is sign-extended to 16 bits.
2. In this operation, DEST/2 represents unsigned division.
3. In this operation, DEST/2 represents signed division.

Tables A-3 and A-4 define the abbreviations and symbols used in Tables A-5 and A-6. Table A-5 shows the effect of each instruction on the Program Status Word flags, and Table A-6 shows the effect of the PSW flags or a specified register bit on conditional jump instructions. (For additional information about the PSW flags, refer to the Program Status Word description in Appendix C.)

Table A-3. PSW Flag Abbreviations

Abbreviation	PSW Flag Name
C	Carry Flag
N	Negative Flag
ST	Sticky Bit Flag
V	Overflow Flag
VT	Overflow-Trap Flag
Z	Zero Flag

Table A-4. PSW Flag Setting Symbols

Symbol	Description
√	The instruction sets or clears the flag, as appropriate.
—	The instruction does not modify the flag.
↓	The instruction may clear the flag, if it is appropriate, but cannot set it.
↑	The instruction may set the flag, if it is appropriate, but cannot clear it.
1	The instruction sets the flag.
0	The instruction clears the flag.
?	The instruction leaves the flag in an indeterminate state.

Table A-5. Effects of Instructions on PSW Flag Settings

Mnemonic	PSW Flag Settings					
	Z	N	C	V	VT	ST
ADD, ADDB	√	√	√	√	↑	—
ADDC, ADDCB	↓	√	√	√	↑	—
AND, ANDB	√	√	0	0	—	—
BMOV, BMOVI	—	—	—	—	—	—
BR (Indirect)	—	—	—	—	—	—
CLR, CLRB	1	0	0	0	—	—
CLRC	—	—	0	—	—	—
CLRVT	—	—	—	—	0	—
CMP, CMPB	√	√	√	√	↑	—
CMPL	√	√	√	√	√	—
DEC, DECB	√	√	√	√	↑	—
DI	—	—	—	—	—	—
DIV, DIVB, DIVU, DIVUB	—	—	—	√	↑	—
DJNZ, DJNZW	—	—	—	—	—	—
DPTS	—	—	—	—	—	—
EI	—	—	—	—	—	—
EPTS	—	—	—	—	—	—
EXT, EXTB	√	√	0	0	—	—
IDLPD						
Legal Key	—	—	—	—	—	—
Illegal Key	0	0	0	0	0	0
INC	√	√	√	√	↑	0
INCB	√	√	√	√	↑	—
JBC, JBS, JC, JE, JGE, JGT, JH, JLE, JLT, JNC, JNE, JNH, JNST	—	—	—	—	—	—
JNV	—	—	—	—	—	—
JNVT	—	—	—	—	0	—
JST, JV	—	—	—	—	—	—
JVT	—	—	—	—	0	—
LCALL, LD, LDB, LDBSE, LDBZE	—	—	—	—	—	—
LJMP	—	—	—	—	—	?
MUL, MULB, MULU, MULUB	—	—	—	—	—	?
NEG, NEGB	√	√	√	√	↑	—
NOP	—	—	—	—	—	—
NORML	√	?	0	—	—	—
NOT, NOTB	√	√	0	0	—	—
OR, ORB	√	√	0	0	—	—
POP	—	—	—	—	—	—
POPA, POPF	√	√	√	√	√	√
PUSH	—	—	—	—	—	—
PUSHA, PUSHF	0	0	0	0	0	0
RET	—	—	—	—	—	—
RST	0	0	0	0	0	0
SCALL	—	—	—	—	—	—
SETC	—	—	1	—	—	—
SHL, SHLB, SHLL	√	?	√	√	↑	—
SHR	√	0	√	0	—	√
SHRA, SHRAB, SHRAL	√	√	√	0	—	√
SHRB, SHRL	√	0	√	0	—	√
SJMP	—	—	—	—	—	—
SKIP	—	—	—	—	—	—
ST, STB	—	—	—	—	—	—
SUB, SUBB	√	√	√	√	↑	—
SUBC, SUBCB	↓	√	√	√	↑	—
TIJMP	—	—	—	—	—	—
TRAP	—	—	—	—	—	—
XCH, XCHB	—	—	—	—	—	—
XOR, XORB	√	√	0	0	—	—

Table A-6. Effect of PSW Flags or Specified Bits on Conditional Jump Instructions

Instruction	Jumps to Destination if	Continues if
DJNZ	decremented byte ≠ 0	decremented byte = 0
DJNZW	decremented word ≠ 0	decremented word = 0
JBC	specified register bit = 0	specified register bit = 1
JBS	specified register bit = 1	specified register bit = 0
JNC	C = 0	C = 1
JNH	C = 0 OR Z = 1	C = 1 AND Z = 0
JC	C = 1	C = 0
JH	C = 1 AND Z = 0	C = 0 OR Z = 1
JGE	N = 0	N = 1
JGT	N = 0 AND Z = 0	N = 1 OR Z = 1
JLT	N = 1	N = 0
JLE	N = 1 OR Z = 1	N = 0 AND Z = 0
JNST	ST = 0	ST = 1
JST	ST = 1	ST = 0
JNV	V = 0	V = 1
JV	V = 1	V = 0
JNVT	VT = 0	VT = 1 (clears VT)
JVT	VT = 1 (clears VT)	VT = 0
JNE	Z = 0	Z = 1
JE	Z = 1	Z = 0

Table A-7 lists the instruction opcodes, in hexadecimal order, along with the corresponding instruction mnemonics.

Table A-7. 8XC196KC/KD Instruction Opcodes

Hex Code	Instruction Mnemonic	Hex Code	Instruction Mnemonic
00	SKIP	28–2F	SCALL
01	CLR	30–37	JBC
02	NOT	38–3F	JBS
03	NEG	40	AND Direct (3 ops)
04	XCH	41	AND Immediate (3 ops)
05	DEC	42	AND Indirect (3 ops)
06	EXT	43	AND Indexed (3 ops)
07	INC	44	ADD Direct (3 ops)
08	SHR	45	ADD Immediate (3 ops)
09	SHL	46	ADD Indirect (3 ops)
0A	SHRA	47	ADD Indexed (3 ops)
0B	XCH	48	SUB Direct (3 ops)
0C	SHRL	49	SUB Immediate (3 ops)
0D	SHLL	4A	SUB Indirect (3 ops)
0E	SHRAL	4B	SUB Indexed (3 ops)
0F	NORML	4C	MULU Direct (3 ops)
10	Reserved	4D	MULU Immediate (3 ops)
11	CLRB	4E	MULU Indirect (3 ops)
12	NOTB	4F	MULU Indexed (3 ops)
13	NEGB	50	ANDB Direct (3 ops)
14	XCHB	51	ANDB Immediate (3 ops)
15	DECB	52	ANDB Indirect (3 ops)
16	EXTB	53	ANDB Indexed (3 ops)
17	INCB	54	ADDB Direct (3 ops)
18	SHRB	55	ADDB Immediate (3 ops)
19	SHLB	56	ADDB Indirect (3 ops)
1A	SHRAB	57	ADDB Indexed (3 ops)
1B	XCHB	58	SUBB Direct (3 ops)
1C–1F	Reserved	59	SUBB Immediate (3 ops)
20–27	SJMP	5A	SUBB Indirect (3 ops)

Table A-7. 8XC196KC/KD Instruction Opcodes (Continued)

Hex Code	Instruction Mnemonic
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)
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)

Hex Code	Instruction Mnemonic
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
9C	DIVUB Direct

Table A-7. 8XC196KC/KD Instruction Opcodes (Continued)

Hex Code	Instruction Mnemonic
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

Hex Code	Instruction Mnemonic
BE	LDBSE Indirect
BF	LDBSE Indexed
C0	ST Direct
C1	BMOV
C2	ST Indirect
C3	ST Indexed
C4	STB Direct
C5	CMPL
C6	STB Indirect
C7	STB Indexed
C8	PUSH Direct
C9	PUSH Immediate
CA	PUSH Indirect
CB	PUSH Indexed
CC	POP Direct
CD	BMOVI
CE	POP Indirect
CF	POP Indexed
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

Table A-7. 8XC196KC/KD Instruction Opcodes (Continued)

Hex Code	Instruction Mnemonic
DF	JE
E0	DJNZ
E1	DJNZW
E2	TIJMP
E3	BR (Indirect)
E4–E6	Reserved
E7	LJMP
E8–EB	Reserved
EC	DPTS
ED	EPTS
EE	Reserved *
EF	LCALL
F0	RET
F1	Reserved

* Opcode EE is reserved; however, it does not generate an unimplemented opcode interrupt.

Hex Code	Instruction Mnemonic
F2	PUSHF
F3	POPF
F4	PUSHA
F5	POPA
F6	IDLPD
F7	TRAP
F8	CLRC
F9	SETC
FA	DI
FB	EI
FC	CLRVT
FD	NOP
FE	*DIV/DIVB/MUL/MULB
FF	RST

* Signed multiplication and division are two-byte instructions. For each signed instruction, the first byte is "FE" and the second is the opcode of the corresponding unsigned instruction. For example, the opcode for MULU (3 operands) direct is "4C," so the opcode for MUL (3 operands) direct is "FE 4C."

Table A-8 is a map of the 8XC196KC/KD opcodes. The first digit of the opcode is listed vertically, and the second digit is listed horizontally. The related instruction mnemonic is shown at the intersection of the two digits. Shading indicates reserved opcodes.

Table A-8. 8XC196KC/KD Opcode Map (Left Half)

Op-code	x0	x1	x2	x3	x4	x5	x6	x7
0x	SKIP	CLR	NOT	NEG	XCH di	DEC	EXT	INC
1x	▨	CLRB	NOTB	NEGB	XCHB di	DECB	EXTB	INCB
2x	SJMP							
3x	JBC							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	AND 3op				ADD 3op			
	di	im	in	ix	di	im	in	ix
5x	ANDB 3op				ADDB 3op			
	di	im	in	ix	di	im	in	ix
6x	AND 2op				ADD 2op			
	di	im	in	ix	di	im	in	ix
7x	ANDB 2op				ADDB 2op			
	di	im	in	ix	di	im	in	ix
8x	OR				XOR			
	di	im	in	ix	di	im	in	ix
9x	ORB				XORB			
	di	im	in	ix	di	im	in	ix
Ax	LD				ADDC			
	di	im	in	ix	di	im	in	ix
Bx	LDB				ADDCB			
	di	im	in	ix	di	im	in	ix
Cx	ST	BMOV	ST		STB	CMPL	STB	
	di		in	ix	di		in	ix
Dx	JNST	JNH	JGT	JNC	JNVT	JNV	JGE	JNE
Ex	DJNZ	DJNZW	TIJMP	BR in	▨	▨	▨	LJMP
Fx	RET	▨	PUSHF	POPF	PUSHA	POPA	IDLPD	TRAP

Shading indicates reserved opcodes.

Table A-8. 8XC196KC/KD Opcode Map (Right Half)

Op-code	x8	x9	xA	xB	xC	xD	xE	xF
0x	SHR	SHL	SHRA	XCH ix	SHRL	SHLL	SHRAL	NORML
1x	SHRB	SHLB	SHRAB	XCHB ix	▨	▨	▨	▨
2x	SCALL							
3x	JBS							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	SUB 3op				MULU 3op *			
	di	im	in	ix	di	im	in	ix
5x	SUBB 3op				MULUB 3op *			
	di	im	in	ix	di	im	in	ix
6x	SUB 2op				MULU 2op *			
	di	im	in	ix	di	im	in	ix
7x	SUBB 2op				MULUB 2op *			
	di	im	in	ix	di	im	in	ix
8x	CMP				DIVU *			
	di	im	in	ix	di	im	in	ix
9x	CMPB				DIVUB *			
	di	im	in	ix	di	im	in	ix
Ax	SUBC				LDBZE			
	di	im	in	ix	di	im	in	ix
Bx	SUBCB				LDBSE			
	di	im	in	ix	di	im	in	ix
Cx	PUSH				POP	BMOVI	POP	
	di	im	in	ix	di		in	ix
Dx	JST	JH	JLE	JC	JVT	JV	JLT	JE
Ex	▨	▨	▨	▨	DPTS	EPTS	▨ **	LCALL
Fx	CLRC	SETC	DI	EI	CLRVT	NOP	signed multiply/ divide*	RST

* Signed multiplication and division are two-byte instructions. The first byte is "FE" and the second is the opcode of the corresponding unsigned instruction.
** Opcode EE is reserved, but it does not generate an unimplemented opcode interrupt.

Table A-9 lists instructions along with their lengths and opcodes for each applicable addressing mode. There are two columns for each addressing mode. The first column lists the instruction length, and the second column lists the hexadecimal opcodes. For indexed instructions, the first column lists instruction lengths as *S/L*, where *S* is the short-indexed instruction length and *L* is the long-indexed instruction length. A dash in any column (—) indicates "not applicable."

Table A-9. Instruction Lengths and Hexadecimal Opcodes

Arithmetic (Group I)								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
ADD (2 ops)	3	64	4	65	3	66	4/5	67
ADD (3 ops)	4	44	5	45	4	46	5/6	47
ADDB (2 ops)	3	74	3	75	3	76	4/5	77
ADDB (3 ops)	4	54	4	55	4	56	5/6	57
ADDC	3	A4	4	A5	3	A6	4/5	A7
ADDCB	3	B4	3	B5	3	B6	4/5	B7
CMP	3	88	4	89	3	8A	4/5	8B
CMPB	3	98	3	99	3	9A	4/5	9B
SUB (2 ops)	3	68	4	69	3	6A	4/5	6B
SUB (3 ops)	4	48	5	49	4	4A	5/6	4B
SUBB (2 ops)	3	78	3	79	3	7A	4/5	7B
SUBB (3 ops)	4	58	4	59	4	5A	5/6	5B
SUBC	3	A8	4	A9	3	AA	4/5	AB
SUBCB	3	B8	3	B9	3	BA	4/5	BB

Table A-9. Instruction Lengths and Hexadecimal Opcodes (Continued)

Arithmetic (Group II)								
Mnemonic	**Direct**		**Immediate**		**Indirect** (1)		**Indexed** S/L (1)	
DIV	4	FE 8C	5	FE 8D	4	FE 8E	5/6	FE 8F
DIVB	4	FE 9C	4	FE 9D	4	FE 9E	5/6	FE 9F
DIVU	3	8C	4	8D	3	8E	4/5	8F
DIVUB	3	9C	3	9D	3	9E	4/5	9F
MUL (2 ops)	4	FE 6C	5	FE 6D	4	F3 6E	5/6	FE 6F
MUL (3 ops)	5	FE 4C	6	FE 4D	5	FE 4E	6/7	FE 4F
MULB (2 ops)	4	FE 7C	4	FE 7D	4	FE 7E	5/6	FE 7F
MULB (3 ops)	5	FE 5C	5	FE 5D	5	FE 5E	6/7	FE 5F
MULU (2 ops)	3	6C	4	6D	3	6E	4/5	6F
MULU (3 ops)	4	4C	5	4D	4	4E	5/6	4F
MULUB (2 ops)	3	7C	3	7D	3	7E	4/5	7F
MULUB (3 ops)	4	5C	4	5D	4	5E	5/6	5F
Logical								
Mnemonic	**Direct**		**Immediate**		**Indirect** (1)		**Indexed** S/L (1)	
AND (2 ops)	3	60	4	61	3	62	4/5	63
AND (3 ops)	4	40	5	41	4	42	5/6	43
ANDB (2 ops)	3	70	3	71	3	72	4/4	73
ANDB (3 ops)	4	50	4	51	4	52	5/5	53
OR	3	80	4	81	3	82	4/5	83
ORB	3	90	3	91	3	92	4/5	93
XOR	3	84	4	85	3	86	4/5	87
XORB	3	94	3	95	3	96	4/5	97

Table A-9. Instruction Lengths and Hexadecimal Opcodes (Continued)

Stack								
Mnemonic	**Direct**		**Immediate**		**Indirect** (1)		**Indexed S/L** (1)	
POP	2	CC	—	—	2	CE	3/4	CF
POPA	—	—	—	—	1	F5	—	—
POPF	—	—	—	—	1	F3	—	—
PUSH	2	C8	3	C9	2	CA	3/4	CB
PUSHA	—	—	—	—	1	F4	—	—
PUSHF	—	—	—	—	1	F2	—	—
Data								
Mnemonic	**Direct**		**Immediate**		**Indirect** (1)		**Indexed S/L** (1)	
LD	3	A0	4	A1	3	A2	4/5	A3
LDB	3	B0	3	B1	3	B2	4/5	B3
LDBSE	3	BC	3	BD	3	BE	4/5	BF
LDBZE	3	AC	3	AD	3	AE	4/5	AF
ST	3	C0	—	—	3	C2	4/5	C3
STB	3	C4	—	—	3	C6	4/5	C7
XCH	3	04	—	—	—	—	4/5	0B
XCHB	3	14	—	—	—	—	4/5	1B
Jump								
Mnemonic	**Direct**		**Immediate**		**Indirect** (1)		**Indexed S/L** (1)	
BR (Indirect)	—	—	—	—	2	E3	2	E3
LJMP	—	—	—	—	—	—	—/2	E7 (2)
SJMP	—	—	—	—	—	—	2/—	20–27 (2)
TIJMP	4	E2	4	E2	—	—	—/4	E2

Table A-9. Instruction Lengths and Hexadecimal Opcodes (Continued)

Call								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
LCALL	—	—	—	—	—	—	—/3	FF (2)
SCALL	—	—	—	—	—	—	2/—	28–2F (2)
RET	—	—	—	—	1	F0	—	—
TRAP	1	F7	—	—	—	—	—	—

Conditional Jump								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
DJNZ	—	—	—	—	—	—	3/—	E0
DJNZW	—	—	—	—	—	—	3/—	E1
JBC	—	—	—	—	—	—	3/—	30–37
JBS	—	—	—	—	—	—	3/—	38–3F
JC	—	—	—	—	—	—	1/—	DB
JE	—	—	—	—	—	—	1/—	DF
JGE	—	—	—	—	—	—	1/—	D6
JGT	—	—	—	—	—	—	1/—	D2
JH	—	—	—	—	—	—	1/—	D9
JLE	—	—	—	—	—	—	1/—	DA
JLT	—	—	—	—	—	—	1/—	DE
JNC	—	—	—	—	—	—	1/—	D3
JNE	—	—	—	—	—	—	1/—	D7
JNH	—	—	—	—	—	—	1/—	D1
JNST	—	—	—	—	—	—	1/—	D0
JNV	—	—	—	—	—	—	1/—	D5
JNVT	—	—	—	—	—	—	1/—	D4
JST	—	—	—	—	—	—	1/—	D8
JV	—	—	—	—	—	—	1/—	DD
JVT	—	—	—	—	—	—	1/—	DC

Table A-9. Instruction Lengths and Hexadecimal Opcodes (Continued)

Shift								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
NORML	3	0F	—	—	—	—	—	—
SHL	3	09	—	—	—	—	—	—
SHLB	3	19	—	—	—	—	—	—
SHLL	3	0D	—	—	—	—	—	—
SHR	3	08	—	—	—	—	—	—
SHRA	3	0A	—	—	—	—	—	—
SHRAB	3	1A	—	—	—	—	—	—
SHRAL	3	0E	—	—	—	—	—	—
SHRB	3	18	—	—	—	—	—	—
SHRL	3	0C	—	—	—	—	—	—
Block								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
BMOV	—	—	—	—	—	—	—/3	C1
BMOVI	—	—	—	—	—	—	—/3	CD
Special								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
CLRC	1	F8	—	—	—	—	—	—
CLRVT	1	FC	—	—	—	—	—	—
DI	1	FA	—	—	—	—	—	—
EI	1	FB	—	—	—	—	—	—
IDLPD	—	—	1	F6	—	—	—	—
NOP	1	FD	—	—	—	—	—	—
RST	1	FE	—	—	—	—	—	—
SETC	1	F9	—	—	—	—	—	—
SKIP	2	00	—	—	—	—	—	—
PTS								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
DPTS	1	EC	—	—	—	—	—	—
EPTS	1	ED	—	—	—	—	—	—

Table A-9. Instruction Lengths and Hexadecimal Opcodes (Continued)

Single								
Mnemonic	Direct		Immediate		Indirect (1)		Indexed S/L (1)	
CLR	—	01	—	—	—	—	—	—
CLRB	—	11	—	—	—	—	—	—
CMPL	—	C5	—	—	—	—	—	—
DEC	—	05	—	—	—	—	—	—
DECB	—	15	—	—	—	—	—	—
EXT	—	06	—	—	—	—	—	—
EXTB	—	16	—	—	—	—	—	—
INC	—	07	—	—	—	—	—	—
INCB	—	17	—	—	—	—	—	—
NEG	—	03	—	—	—	—	—	—
NEGB	—	13	—	—	—	—	—	—
NOT	—	02	—	—	—	—	—	—
NOTB	—	12	—	—	—	—	—	—

NOTES:

1. Indirect normal and indirect auto-increment share the same opcodes, as do short- and long-indexed modes. To use indirect normal or short-indexed mode, the second byte of the instruction must be even. To use indirect auto-increment or long-indexed mode, the second byte of the instruction must be odd.
2. For these instructions (SCALL, SJMP, LCALL, LJMP), the 3 least-significant bits of the opcode are concatenated with the 8 bits to form an 11-bit, 2's complement offset.

Table A-10 lists instructions alphabetically within groups, along with their execution times, expressed in state times. The table denotes execution times for indirect and indexed addressing modes as *R*/*M*, where *R* is the execution time using SFRs and internal RAM (0H–1FFH) and *M* is the execution time using the memory controller (200H–0FFFFH).

Table A-10. 8XC196KC/KD Instruction Execution Times (in State Times)

Arithmetic (Group I)						
			Indirect		**Indexed**	
Mnemonic	**Direct**	**Immed.**	**Normal**	**Auto-Inc.**	**Short**	**Long**
ADD (2 ops)	4	5	6/8	7/9	6/8	7/9
ADD (3 ops)	5	6	7/10	8/11	7/10	8/11
ADDB (2 ops)	4	5	6/8	7/9	6/8	7/9
ADDB (3 ops)	5	6	7/10	8/11	7/10	8/11
ADDC	4	5	6/8	7/9	6/8	7/9
ADDCB	4	5	6/8	7/9	6/8	7/9
CMP	4	5	6/8	7/9	6/8	7/9
CMPB	4	5	6/8	7/9	6/8	7/9
SUB (2 ops)	4	5	6/8	7/9	6/8	7/9
SUB (3 ops)	5	6	7/10	8/11	7/10	8/11
SUBB (2 ops)	4	5	6/8	7/9	6/8	7/9
SUBB (3 ops)	5	6	7/10	8/11	7/10	8/11
SUBC	4	5	6/8	7/9	6/8	7/9
SUBCB	4	5	6/8	7/9	6/8	7/9

Table A-10. 8XC196KC/KD Instruction Execution Times (Continued)

Arithmetic (Group II)						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
DIV	26	27	28/31	29/32	29/32	30/33
DIVB	18	18	20/23	21/24	21/24	22/25
DIVU	24	25	26/29	27/30	27/30	28/31
DIVUB	16	16	18/21	19/22	19/22	20/23
MUL (2 ops)	16	17	18/21	19/22	19/22	20/23
MUL (3 ops)	16	17	18/21	19/22	19/22	20/23
MULB (2 ops)	12	12	14/17	15/18	15/18	16/19
MULB (3 ops)	12	12	14/17	15/18	15/18	16/19
MULU (2 ops)	14	15	16/19	17/19	17/20	18/21
MULU (3 ops)	14	15	16/19	17/19	17/20	18/21
MULUB (2 ops)	10	10	12/15	13/15	12/16	14/17
MULUB (3 ops)	10	10	15/15	12/16	12/16	14/17

Logical						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
AND (2 ops)	4	5	6/8	7/9	6/8	7/9
AND (3 ops)	5	6	7/10	8/11	7/10	8/11
ANDB (2 ops)	4	5	6/8	7/9	6/8	7/9
ANDB (3 ops)	5	6	7/10	8/11	7/10	8/11
OR	4	5	6/8	7/9	6/8	7/9
ORB	4	5	6/8	7/9	6/8	7/9
XOR	4	5	6/8	7/9	6/8	7/9
XORB	4	5	6/8	7/9	6/8	7/9

Table A-10. 8XC196KC/KD Instruction Execution Times (Continued)

Stack (Internal Stack)						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
POP	8	—	10/12	11/13	11/13	12/14
POPA	12	—	—	—	—	—
POPF	7	—	—	—	—	—
PUSH	6	7	9/12	10/13	10/13	11/14
PUSHA	12	—	—	—	—	—
PUSHF	6	—	—	—	—	—
Stack (External Stack)						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
POP	11	—	13/15	14/16	14/16	15/17
POPA	18	—	—	—	—	—
POPF	10	—	—	—	—	—
PUSH	8	9	11/14	12/15	12/15	13/16
PUSHA	18	—	—	—	—	—
PUSHF	8	—	—	—	—	—
Data						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
LD	4	5	5/8	6/8	6/9	7/10
LDB	4	5	5/8	6/8	6/9	7/10
LDBSE	4	4	5/8	6/8	6/9	7/10
LDBZE	4	4	5/8	6/8	6/9	7/10
ST	4	—	5/8	6/9	6/9	7/10
STB	4	—	5/8	6/9	6/9	7/10
XCH	5	—	—	—	8/13	9/14
XCHB	5	—	—	—	8/13	9/14

Table A-10. 8XC196KC/KD Instruction Execution Times (Continued)

Jump						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
BR (Indirect)	—	—	7	7	7	7
LJMP	—	—	—	—	—	7
SJMP	—	—	—	—	7	—
TIJMP internal/internal external/internal external/external	 15 18 21	 15 18 21	—	—	—	—
Call (Internal Stack)						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
LCALL	—	—	—	—	—	11
RET	—	—	11	—	—	—
SCALL	—	—	—	—	11	—
TRAP	16	—	—	—	—	—
Call (External Stack)						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
LCALL	—	—	—	—	—	13
RET	—	—	14	—	—	—
SCALL	—	—	—	—	13	—
TRAP	18	—	—	—	—	—

Table A-10. 8XC196KC/KD Instruction Execution Times (Continued)

Conditional Jump	
Mnemonic	**Short-Indexed**
DJNZ	5 (jump not taken), 9 (jump taken)
DJNZW 80C196KC only 8XC196KC/KD	 7 (jump not taken), 11 (jump taken) 6 (jump not taken), 10 (jump taken)
JBC	5 (jump not taken), 9 (jump taken)
JBS	5 (jump not taken), 9 (jump taken)
JC	4 (jump not taken), 8 (jump taken)
JE	4 (jump not taken), 8 (jump taken)
JGE	4 (jump not taken), 8 (jump taken)
JGT	4 (jump not taken), 8 (jump taken)
JH	4 (jump not taken), 8 (jump taken)
JLE	4 (jump not taken), 8 (jump taken)
JLT	4 (jump not taken), 8 (jump taken)
JNC	4 (jump not taken), 8 (jump taken)
JNE	4 (jump not taken), 8 (jump taken)
JNH	4 (jump not taken), 8 (jump taken)
JNST	4 (jump not taken), 8 (jump taken)
JNV	4 (jump not taken), 8 (jump taken)
JNVT	4 (jump not taken), 8 (jump taken)
JST	4 (jump not taken), 8 (jump taken)
JV	4 (jump not taken), 8 (jump taken)
JVT	4 (jump not taken), 8 (jump taken)

Table A-10. 8XC196KC/KD Instruction Execution Times (Continued)

Shift	
Mnemonic	**Direct**
NORML	8 + 1 per shift (9 for 0 shift)
SHL	6 + 1 per shift (7 for 0 shift)
SHLB	6 + 1 per shift (7 for 0 shift)
SHR	6 + 1 per shift (7 for 0 shift)
SHRA	6 + 1 per shift (7 for 0 shift)
SHRAB	6 + 1 per shift (7 for 0 shift)
SHRAL	7 + 1 per shift (8 for 0 shift)
SHRB	6 + 1 per shift (7 for 0 shift)
SHRL	7 + 1 per shift (8 for 0 shift)

Block	
Mnemonic	**Long-Indexed**
BMOV	internal/internal 6 + 8 per word external/internal 6 + 11 per word external/external 6 + 14 per word
BMOVI	internal/internal 7 + 8 per word + 14 per interrupt external/internal 7 + 11 per word + 14 per interrupt external/external 7 + 14 per word + 14 per interrupt

Special

			Indirect		Indexed	
Mnemonic	**Direct**	**Immed.**	**Normal**	**Auto-Inc.**	**Short**	**Long**
CLRC	2	—	—	—	—	—
CLRVT	2	—	—	—	—	—
DI	2	—	—	—	—	—
EI	2	—	—	—	—	—
IDLPD Valid key Invalid key	 — —	 8 25	 — —	 — —	 — —	 — —
NOP	2	—	—	—	—	—
RST (including fetch of configuration byte)	20	—	—	—	—	—
SETC	2	—	—	—	—	—
SKIP	3	—	—	—	—	—

Table A-10. 8XC196KC/KD Instruction Execution Times (Continued)

PTS						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
DPTS	2	—	—	—	—	—
EPTS	2	—	—	—	—	—

Single						
			Indirect		Indexed	
Mnemonic	Direct	Immed.	Normal	Auto-Inc.	Short	Long
CLR	3	—	—	—	—	—
CLRB	3	—	—	—	—	—
CMPL	7	—	—	—	—	—
DEC	3	—	—	—	—	—
DECB	3	—	—	—	—	—
EXT	4	—	—	—	—	—
EXTB	4	—	—	—	—	—
INC	3	—	—	—	—	—
INCB	3	—	—	—	—	—
NEG	3	—	—	—	—	—
NEGB	3	—	—	—	—	—
NOT	3	—	—	—	—	—
NOTB	3	—	—	—	—	—

Table A-11. 8XC196KC/KD PTS Cycle Execution Times

PTS Cycles	
PTS Mode	**Execution Time (in State Times)**
Single Transfer mode	
internal/internal	18
external/internal	21
external/external	24
Block Transfer mode	
internal/internal	13 + 7 per transfer (1 minimum)
external/internal	16 + 7 per transfer (1 minimum)
external/external	19 + 7 per transfer (1 minimum)
A/D Scan mode	
SFRs/Internal RAM	21
Memory Controller	25
HSI mode	
SFRs/Internal RAM	12 + 10 per transfer (1 minimum)
Memory Controller	16 + 10 per transfer (1 minimum)
HSO mode	
SFRs/Internal RAM	11 + 10 per transfer (1 minimum)
Memory Controller	15 + 10 per transfer (1 minimum)

8XC196KC/KD Signal Descriptions

B

APPENDIX B
8XC196KC/KD SIGNAL DESCRIPTIONS

This appendix provides reference information for the pin functions of the 8XC196KC and 8XC196KD. Table B-1 defines the columns used in Table B-2, which describes the pin functions. Table B-3 shows the reset status of I/O and control pins, and Table B-4 defines the pin status terminology. Tables B-5 and B-6 list package pin numbers along with their corresponding pin functions.

Table B-1. Description of Columns of Table B-2

Column Heading	Description
Function Name	Lists the pin functions, arranged alphabetically. Many pins have more than one function, so there are more entries in this column than there are pins. Every pin function is listed in this column; for each pin function, any additional functions that share the pin are listed in the "Additional Functions" column.
Additional Functions	Lists all other functions that the pin provides.
Selected by	Lists the register setting, pin state, or other condition that causes the function listed in the "Function Name" column to become active. A dash (—) indicates "none" or "not applicable."
Type	Identifies the pin function listed in the "Function Name" column as an input (I), output (O), bidirectional (I/O), quasi-bidirectional (QBD), power (PWR), or ground (GND). Note that all inputs, with the exception of RESET#, are *sampled inputs.* RESET# is a level-sensitive input. During Powerdown mode, the powerdown circuitry uses EXTINT as a level-sensitive input.
Description	Briefly describes the pin function listed in the "Function Name" column.

Table B-2. Signal Descriptions

Function Name	Additional Functions	Selected by	Type	Description
ACH0 ACH1 ACH2 ACH3 ACH4 ACH5 ACH6 ACH7	P0.0 P0.1 P0.2 P0.3 P0.4/PMODE.0 P0.5/PMODE.1 P0.6/PMODE.2 P0.7/PMODE.3	—	I	Analog Channels 0 through 7. ACH0–ACH7 are analog inputs to the A/D converter. These pins may individually be used as analog inputs (ACH*x*) or digital inputs (P0.*x*). While it is possible for the pins to function simultaneously as analog and digital inputs, this is not recommended because reading Port 0 while a conversion is in process can produce unreliable conversion results. The ANGND and V_{REF} pins must be connected for the A/D converter and Port 0 to function.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
AD0–AD7 AD8–AD15	P3.0–P3.7 P4.0–P4.7	Bus access to external address	I/O	System Address/Data Bus. These pins provide a multiplexed address and data bus. During the address phase of the bus cycle, address bits 0–15 are presented on the bus and can be latched using ALE or ADV#. During the data phase, 8- or 16-bit data is transferred.
ADV#	ALE	CCR.3=0	O	Address Valid. This active-low output signal is asserted only during external memory accesses. ADV# indicates that valid address information is available on the system address/data bus. The signal remains low while a valid bus cycle is in progress and is returned high as soon as the bus cycle completes. The ADV# signal is used by an external latch to demultiplex the address from the address/data bus. ADV# can also be used with a decoder to generate chip-selects for external memory.
AINC#	P2.4/T2RST	EA# = V_{EA} (programming mode)	I	Auto Increment. In Slave Programming Mode, this active-low input signal enables the auto-increment mode. Auto increment allows reading from or writing to sequential EPROM locations without requiring address transactions across the programming bus for each read or write.
ALE	ADV#	CCR.3=1	O	Address Latch Enable. This active-high output signal is asserted only during external memory accesses. ALE indicates that valid address information is available on the system address/data bus and signals the start of a valid bus cycle. ALE differs from ADV# in that it is not returned high until a new bus cycle is to begin. ALE is used by an external latch to demultiplex the address from the address/data bus.
ANGND	—	—	GND	Reference ground for the A/D converter and the logic used to read Port 0. ANGND must be connected for the A/D converter and Port 0 to function.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
BHE#	WRH#	CCR.2=1	O	Byte High Enable. This active-low output signal is asserted only during **word** writes and **high byte** writes to external memory. BHE# indicates that valid data is being transferred (written) over the upper half of the system address/data bus. BHE#, in conjunction with A0, selects the memory byte to be written: **BHE#** **A0** **Byte Written** 0 0 both bytes 0 1 high byte only 1 0 low byte only
BREQ#	P1.5	WSR.7=1	O	Bus Request. This active-low output signal is asserted during a HOLD cycle when the bus controller has a pending external memory cycle. The earliest time in the HOLD cycle that the device can assert BREQ# is at the same time it asserts HLDA#. Once it is asserted, BREQ# remains asserted until HOLD# is removed. When this function is active, the pin acts as a standard output (not quasi-bidirectional). Once the alternate function is enabled, the pin functions as BREQ# until the device is reset.
BUSWIDTH	—	CCR.1=1	I	Bus Width. This active-high input signal dynamically selects the bus width. When high, BUSWIDTH selects a 16-bit bus width; when low, it selects an 8-bit bus width. BUSWIDTH is active during a CCR fetch. When CCR.1=0 the bus width is always 8 bits; the BUSWIDTH signal is ignored.
CLKOUT	—	IOC3.1=0 **8XC196KD and 8XC196KC (C-Step) only**	O	Clock Output. Output of the internal clock generator. The CLKOUT frequency is 1/2 the oscillator frequency input (XTAL1) and CLKOUT has a 50% duty cycle. On the 8XC196KD and the 8XC196KC (C-Step), clearing IOC3.1 enables CLKOUT; setting IOC3.1 disables the signal. On earlier versions of the 8XC196KC, CLKOUT cannot be disabled.
CPVER	P2.6/T2UP-DN	EA# = V_{EA} (programming mode)	O	Cumulative Program Verification. This active-high output signal indicates whether any verify errors have occurred since the device entered programming mode. CPVER remains high until a verify error occurs, at which time it is driven low. Once an error occurs, CPVER remains low until the device exits programming mode. When high, CPVER indicates that all locations have programmed correctly since the device entered programming mode.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
EA#	Programming mode select (EA# = V_{EA})	—	I	External Access. This active-low input signal directs memory accesses to on-chip or off-chip memory. When low, it selects off-chip memory; when high, it selects on-chip ROM or EPROM. EA# is sampled **only** on the rising edge of RESET#. EA# = V_{EA} on the rising edge of RESET# causes the device to enter the programming mode selected by PMODE.0–PMODE.3. For ROMless devices, EA# must be tied low.
EXTINT				External Interrupt. IOC1.1 assigns this input to either P2.2 or P0.7. EXTINT must be asserted for greater than two state times to guarantee that it is recognized. In Powerdown mode, EXTINT causes the chip to return to normal operating mode. EXTINT is normally a *sampled input*; however, the powerdown circuitry uses it as a level-sensitive input in Powerdown mode.
	P2.2/PROG#	IOC1.1=0	I	P2.2 always generates EXTINT1 (INT13, 203AH). When IOC1.1=0, a rising edge on the P2.2 pin also generates EXTINT (INT07, 200EH). EXTINT and EXTINT1 must be asserted for greater than two state times to guarantee that they are recognized.
	P0.7/PMODE.3 /ACH7	IOC1.1=1	I	When IOC1.1=1, a rising edge on the P0.7 pin generates external interrupt EXTINT (INT07, 200EH). EXTINT must be asserted for greater than two state times to guarantee that it is recognized.
HLDA#	P1.6	WSR.7=1	O	Bus Hold Acknowledge. This active-low output indicates that the 8XC196KC/KD has released the bus as a result of another device asserting HOLD#. When this function is active, the pin acts as a standard output (not quasi-bidirectional). Once the HLDA# function is enabled, the pin functions as HLDA# until the device is reset.
HOLD#	P1.7	WSR.7=1	I	Bus Hold. This active-low input is used to request control of the bus. When this function is active, the pin acts as a standard input (not quasi-bidirectional). Once the alternate function is enabled, the pin functions as HOLD# until the device is reset.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
HSI.0	INT04 interrupt /T2 reset source	IOC0.0=1	I	Input to the High-Speed Input module. The HSI.0 pin can also be used to generate an interrupt. A rising edge on HSI.0 generates the HSI.0 Pin interrupt (INT04, 2008H). HSI.0 must be asserted for greater than two state times to guarantee that it is recognized. In addition, when IOC0.5=1, a rising edge on HSI.0 resets Timer 2.
HSI.1	T2 clock source	IOC0.2=1	I	Input to the High-Speed Input module. When IOC0.7=0 and IOC3.0=0, the HSI.1 pin functions as the T2 clock source.
HSI.2	HSO.4	IOC0.4=1	I	Input to the High-Speed Input module. Note that HSI and HSO functions can be active at the same time, in which case the pin acts as an output that the HSO monitors.
HSI.3	HSO.5	IOC0.6=1	I	Input to the High-Speed Input module. Note that HSI and HSO functions can be active at the same time, in which case the pin acts as an output that the HSO monitors.
HSO.0 HSO.1 HSO.2 HSO.3	— — — —	— — — —	O	Outputs from the High-Speed Output module.
HSO.4	HSI.2	IOC1.4=1	O	Output from the High-Speed Output module. Note that the HSI and HSO functions can be active at the same time, in which case the pin acts as an output that the HSO monitors.
HSO.5	HSI.3	IOC1.6=1	O	Output from the High-Speed Output module. Note that the HSI and HSO functions can be active at the same time, in which case the pin acts as an output that the HSI monitors.
INST	—	—	O	Instruction Fetch. This signal is valid only during external memory read cycles. When high, INST indicates that an instruction is being fetched; when low, it indicates that data is being read. INST can be used in applications that require separate memory banks for instructions and data. (Note that CCB bytes and interrupt vectors are considered **data**.)
NMI	—	—	I	Nonmaskable Interrupt. A positive transition causes a vector through the NMI interrupt at location 203EH. NMI must be asserted for greater than one state time to guarantee that it is recognized. NMI is used by Intel tools that could conflict with user software. When NMI is not used, it should be tied low.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	ACH0 ACH1 ACH2 ACH3 ACH4/PMODE.0 ACH5/PMODE.1 ACH6/PMODE.2 ACH7/PMODE.3 /EXTINT	—	I	Port 0. This port is an 8-bit, high-impedance, input-only port. Port 0 is read (only) at location 0EH in HWindow 0. P0.0–P0.7 are digital inputs. These pins may individually be used as analog inputs (ACH*x*) or digital inputs (P0.*x*). While it is possible for the pins to function simultaneously as analog and digital inputs, this is not recommended because reading Port 0 while a conversion is in process can produce unreliable conversion results. ANGND and V_{REF} must be connected for Port 0 and the A/D converter to function.

P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
	—
	—
	—
	PWM1
	PWM2
	BREQ#
	HLDA#
	HOLD#
		—
		—
		—
		IOC3.2=0
		IOC3.3=0
		WSR.7=0
		WSR.7=0
		WSR.7=0
			QBD	Port 1. This port is an 8-bit, quasi-bidirectional input/output port. Port 1 is read and written at location 0FH in HWindow 0.
				The additional functions act as standard I/O pins (not quasi-bidirectional).

P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	TXD/PVER# RXD/PALE# EXTINT/PROG# T2CLK T2RST/AINC# PWM0 T2UP-DN/CPVER T2CAPTURE /PACT	IOC1.5=0 SPCON.3=0 — — — IOC1.0=0 — —	O I I I I O QBD QBD	Port 2. This port is an 8-bit, multifunctional port. Port 2 is read and written at location 10H in HWindow 0.
P3.0–P3.7 P4.0–P4.7	AD0–AD7 AD8–AD15	EA#=1	I/O	Ports 3 and 4. These are 8-bit, bidirectional, input/output ports with open-drain outputs. The pins are shared with the multiplexed address/data bus, which has strong internal pull-ups. Ports 3 and 4 can be read and written only as a word, at location 1FFEH. During Programming Modes, these ports function as the PBUS.
PACT#	P2.7 /T2CAPTURE	EA# = V_{EA} (programming mode)	O	Programming Active. In Auto Programming Mode, PACT# low indicates that programming activity is occurring.
PALE#	P2.1/RXD	EA# = V_{EA} (programming mode)	I	Programming ALE. When PALE# is asserted, data and commands on Ports 3 and 4 are read into the device.
PMODE.0 PMODE.1 PMODE.2 PMODE.3	P0.4/ACH4 P0.5/ACH5 P0.6/ACH6 P0.7/ACH7	EA# = V_{EA} (programming mode)	I	Programming Mode Select. Determines the EPROM programming algorithm that is performed. PMODE is sampled after a device reset when EA# = V_{EA} and must be stable while the device is operating.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
PROG#	P2.2/EXTINT	EA# = V_{EA} (programming mode)	I	Programming Start. This active-low input is valid only in Slave Programming Mode. When asserted, PROG# causes the data on the programming bus to be programmed into the EPROM. When PROG# is deasserted, the programming pulse is terminated.
PVER	P2.0/TXD	EA# = V_{EA} (programming mode)	O	Program Verification. In Programming Modes, this active-high output signal is asserted to indicate that the word has programmed correctly.
PWM0	P2.5	IOC1.0=1	O	Pulse Width Modulator (PWM) Output 0. If PWM0 is forced high on the rising edge of RESET#, the device enters Test Mode.
PWM1	P1.3	IOC3.2=1	O	PWM Output 1.
PWM2	P1.4	IOC3.3=1	O	PWM Output 2.
RD#	—	—	O	Read signal output to external memory. RD# is asserted only during external memory reads.
READY	—	—	I	Ready input. This signal is used to lengthen external memory cycles by generating “wait states” for interfacing to slow memory. When READY is high, CPU operation continues in a normal manner. If READY is low prior to the falling edge of CLKOUT, the memory controller inserts wait states until the next positive transition in CLKOUT occurs with READY high or until the number of wait states is equal to the number programmed into CCR.4 and CCR.5. READY is ignored for all internal memory accesses. READY is active during a CCR fetch.
RESET#	—	—	I/O	Reset input to and open-drain output from the chip. A falling edge on RESET# initiates the reset process. When RESET# is first asserted, the chip turns on a pull-down transistor connected to the RESET pin for 16 state times. This function can also be activated by a watchdog timer overflow or by execution of the RST instruction. In Powerdown mode, the reset process causes the chip to return to normal operating mode.
RXD	P2.1/PALE#	SPCON.3 = 1	I/O	Receive Serial Data. In modes 1, 2, and 3, RXD is used to receive serial port data. In mode 0, it functions as an input or an open-drain output for data.
T2CAPTURE	P2.7/PACT#	—	QBD	A rising edge on P2.7 captures the value of Timer 2 in the T2CAPTURE register and triggers a Timer 2 Capture interrupt (INT11, 2036H). T2CAPTURE must be asserted for greater than two state times to guarantee that it is recognized.

Table B-2. Signal Descriptions (Continued)

Function Name	Additional Functions	Selected by	Type	Description
T2CLK	P2.3	IOC0.7=0 and IOC3.0=0	I	Timer 2 clock input and
		BAUD_RATE. 15 = 0	I	Serial port baud rate generator input.
T2RST	P2.4/AINC#	IOC0.3=1 and IOC0.5=0	I	Timer 2 Reset. A rising edge on T2RST resets Timer 2.
T2UP-DN	P2.6 /CPVER	IOC2.1=1	I	Timer 2 Up/Down Control. This active-high input controls the direction of the Timer 2 counter. When T2UP-DN is high, Timer 2 counts down; when T2UP-DN is low, Timer 2 counts up.
TXD	P2.0/PVER	IOC1.5=1	O	Transmit Serial Data. In modes 1, 2, and 3, TXD is used to transmit serial port data. In mode 0, it is used as the serial clock output. Holding TXD low on the rising edge of RESET# causes the device to enter ONCE mode.
V_{CC}	—	—	PWR	Digital supply voltage (+5 volts).
V_{PP}	—	—	PWR	Programming voltage. Also the timing pin for the "return from power-down" circuit.
V_{REF}	—	—	PWR	Reference voltage for the A/D converter. V_{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0. V_{REF} must be connected for the A/D and Port 0 to function.
V_{SS}	—	—	GND	Digital circuit ground (0 volts). There are multiple Vss pins, all of which must be connected.
WR#	WRL#	CCR.2=1	O	Write. This active-low output indicates that an external write is occurring. This signal is asserted only during external memory writes.
WRH#	BHE#	CCR.2=0	O	Write High. During 16-bit bus cycles, this active-low output signal is asserted during high byte writes and word writes. During 8-bit bus cycles, WRH# is asserted for all write operations.
WRL#	WR#	CCR.2=0	O	Write Low. During 16-bit bus cycles, this active-low output signal is asserted during low byte writes and word writes. During 8-bit bus cycles, WRL# is asserted for all write operations.
XTAL1	—	—	I	Input of the on-chip oscillator inverter and of the internal clock generator. If an external oscillator is used, XTAL1 also serves as the 8XC196KC/KD clock input (the XTAL1 V_{IH} specification must be met.)
XTAL2	—	—	O	Output of the on-chip oscillator inverter.

Table B-3 lists the default functions of the 8XC196KC/KD I/O and control pins with their values during and after reset. Table B-4 defines the pin status terminology.

Table B-3. Pin Reset Status

Pin Name	Multiplexed Port Pins	Pin Status During Reset	Pin Status After Reset
ACH0–ACH7	P0.0–P0.7	Undefined Inputs (Note 1)	Undefined Inputs (Note 1)
PORT1	P1.0–P1.7	Weak Pull-ups (I_{IL} spec)	Weak Pull-ups (I_{IL} spec)
TXD	P2.0	Strong Pull-up (I_{LI1} spec)	Strongly Driven
RXD	P2.1	Undefined Input (Note 3)	Undefined Input (Note 3)
EXTINT	P2.2	Undefined Input (Note 3)	Undefined Input (Note 3)
T2CLK	P2.3	Undefined Input (Note 3)	Undefined Input (Note 3)
T2RST	P2.4	Undefined Input (Note 3)	Undefined Input (Note 3)
PWM0	P2.5	Medium Pull-down	Strongly Driven
—	P2.6–P2.7	Weak Pull-ups	Weak Pull-ups
AD0–AD15	P3.0–P4.7	Weak Pull-ups	Address/Data Bus or Open-Drain I/O (Note 2)
HSI.0, HSI.1	—	Undefined Input (Note 3)	Undefined Input (Note 3)
HSI.2/HSO.4	—	Undefined Input (Note 3)	Undefined Input (Note 3)
HSI.3/HSO.5	—	Undefined Input (Note 3)	Undefined Input (Note 3)
HSO.0–HSO.3	—	Weak Pull-down	Weak Pull-down
ALE	—	Weak Pull-up	Strongly Driven
BHE#	—	Weak Pull-up	Strongly Driven
BUSWIDTH	—	Undefined Input (Note 3)	Undefined Input (Note 3)
CLKOUT	—	CLKOUT (Strongly Driven)	CLKOUT (Strongly Driven)
EA#	—	Undefined Input (Note 3)	Undefined Input (Note 3)
INST	—	Weak Pull-down	Strongly Driven
NMI	—	Weak Pull-down (I_{IH1} spec)	Weak Pull-down (I_{IH1} spec)
RD#	—	Weak Pull-up	Strongly Driven
READY	—	Undefined Input (Note 3)	Undefined Input (Note 3)
RESET#	—	Medium Pull-up (R_{RST} spec)	Medium Pull-up (R_{RST} spec)
WR#	—	Weak Pull-up	Strongly Driven

NOTES:

1. These pins are allowed to float. However, it is recommended that unused pins be tied high or low.
2. The state of these pins depends on device configuration. If the address/data bus is active, the pins act as a strongly driven bus; otherwise, they act as an open-drain I/O port and are left floating.
3. These pins must be driven and not left floating. Input voltage must **not** exceed V_{CC} during power-up.
4. Consult the 8XC196KC/KD data sheet for specifications.

Table B-4. Pin Status Descriptions

Pin Status	Approximate Value
Weak Pull-up	70 μA
Medium Pull-up	1 mA
Strong Pull-up	12 mA
Weak Pull-down	200 μA
Medium Pull-down	1 mA
Strongly Driven High	see V_{OH} specification
Strongly Driven Low	see V_{OL} specification

NOTE:

These typical maximum values are approximate; they are provided for reference only and are not guaranteed.

Table B-5. 8XC196KC/KD 68-Lead PLCC Package Pin Assignments

Pin No.	Pin Function(s)	Pin No.	Pin Function(s)
1	V_{CC}	35	HSO.3
2	EA#	36	V_{SS}
3	NMI	37	V_{PP}
4	P0.3/ACH3	38	P2.7/T2CAPTURE/PACT#
5	P0.1/ACH1	39	P2.5/PWM0
6	P0.0/ACH0	40	WR#/WRL#
7	P0.2/ACH2	41	BHE#/WRH#
8	P0.6/ACH6/PMODE.2	42	P2.4/T2RST/AINC#
9	P0.7/ACH7/PMODE.3/EXTINT	43	READY
10	P0.5/ACH5/PMODE.1	44	P2.3/T2CLK
11	P0.4/ACH4/PMODE.0	45	P4.7/AD15
12	ANGND	46	P4.6/AD14
13	V_{REF}	47	P4.5/AD13
14	V_{SS}	48	P4.4/AD12
15	P2.2/EXTINT/PROG#	49	P4.3/AD11
16	RESET#	50	P4.2/AD10
17	P2.1/RXD/PALE#	51	P4.1/AD9
18	P2.0/TXD/PVER	52	P4.0/AD8
19	P1.0	53	P3.7/AD7
20	P1.1	54	P3.6/AD6
21	P1.2	55	P3.5/AD5
22	P1.3/PWM1	56	P3.4/AD4
23	P1.4/PWM2	57	P3.3/AD3
24	HSI.0	58	P3.2/AD2
25	HSI.1	59	P3.1/AD1
26	HSI.2/HSO.4	60	P3.0/AD0
27	HSI.3/HSO.5	61	RD#
28	HSO.0	62	ALE/ADV#
29	HSO.1	63	INST
30	P1.5/BREQ#	64	BUSWIDTH
31	P1.6/HLDA#	65	CLKOUT
32	P1.7/HOLD#	66	XTAL2
33	P2.6/T2UP-DN/CPVER	67	XTAL1
34	HSO.2	68	V_{SS}

Table B-6. 8XC196KC/KD 80-Lead QFP Package Pin Assignments

Pin No.	Pin Function(s)	Pin No.	Pin Function(s)
1	P3.1/AD1	41	HSI.2/HSO.4
2	P3.0/AD0	42	V_{SS}
3	RD#	43	HSI.3/HSO.5
4	ALE/ADV#	44	HSO.0
5	INST	45	HSO.1
6	BUSWIDTH	46	P1.5/BREQ#
7	CLKOUT	47	P1.6/HLDA#
8	XTAL2	48	P1.7/HOLD#
9	XTAL1	49	P2.6/T2UP-DN/CPVER
10	V_{SS}	50	HSO.2
11	V_{SS}	51	V_{SS}
12	V_{CC}	52	V_{CC}
13	V_{CC}	53	HSO.3
14	EA#	54	V_{SS}
15	NMI	55	V_{SS}
16	P0.3/ACH3	56	V_{PP}
17	P0.1/ACH1	57	P2.7/T2CAPTURE/PACT#
18	P0.0/ACH0	58	P2.5/PWM0
19	P0.2/ACH2	59	WR#/WRL#
20	P0.6/ACH6/PMODE.2	60	BHE#/WRH#
21	P0.7/ACH7/PMODE.3/EXTINT	61	P2.4/T2RST/AINC#
22	No Connection	62	READY
23	P0.5/ACH5/PMODE.1	63	V_{SS}
24	P0.4/ACH4/PMODE.0	64	P2.3/T2CLK
25	ANGND	65	P4.7/AD15
26	V_{REF}	66	P4.6/AD14
27	V_{SS}	67	P4.5/AD13
28	P2.2/EXTINT/PROG#	68	P4.4/AD12
29	V_{CC}	69	P4.3/AD11
30	RESET#	70	P4.2/AD10
31	P2.1/RXD/PALE#	71	P4.1/AD9
32	P2.0/TXD/PVER	72	P4.0/AD8
33	V_{SS}	73	P3.7/AD7
34	P1.0	74	P3.6/AD6
35	P1.1	75	V_{CC}
36	P1.2	76	P3.5/AD5
37	P1.3/PWM1	77	P3.4/AD4
38	P1.4/PWM2	78	P3.3/AD3
39	HSI.0	79	V_{SS}
40	HSI.1	80	P3.2/AD2

8XC196KC/KD Registers

C

APPENDIX C
8XC196KC/KD REGISTERS

This appendix provides reference information about the registers of the 8XC196KC/KD. Table C-1 lists the modules and major components of the 8XC196KC/KD with their related configuration and status registers. Table C-2 lists the horizontal windows (HWindows) with the special function registers (SFRs) and functions available in each. Table C-3 lists the SFRs, arranged alphabetically by mnemonic, along with register names, addresses, and reset values. Table C-4 lists the interrupts of the 8XC196KC/KD, the vector locations for both normal and PTS interrupt vectors, and interrupt priorities. Table C-5 lists interrupt sources with their related interrupt vectors, the mask bits that enable the interrupts, and the register bits that select the sources. Following the tables, individual descriptions of the registers are arranged alphabetically.

Table C-1. Modules and Related Registers

A/D	HSI	HSO	PTS	PWM
AD_COMMAND	HSI_MODE	HSO_COMMAND	PTSBLOCK	PWM0_CONTROL
AD_RESULT	HSI_STATUS	HSO_TIME	PTSCON	PWM1_CONTROL
AD_TIME	HSI_TIME	IOC1	PTSCOUNT	PWM2_CONTROL
IOC2	IOC0	IOC2	PTSDST	IOC2
	IOS1	IOS0	PTS_REG	IOC3
		IOS1	PTS_S/D	
		IOS2	PTSSEL	
			PTSSRC	
			PTSSRV	
Interrupts	**I/O Ports**	**Serial Port**	**Timer 1**	**Timer 2**
INT_MASK	IOPORT0	BAUD_RATE	TIMER1	TIMER2
INT_MASK1	IOPORT1	SBUF (RX)	IOC1	T2CAPTURE
INT_PEND	IOPORT2	SBUF (TX)	IOS1	IOC0
INT_PEND1	IOPORT34	SP_CON		IOC1
		SP_STAT		IOC2
				IOC3
Chip Config.	**Stack**	**Watchdog**	**Window Select**	**Zero**
CCR	SP	WATCHDOG	WSR	ZERO_REG

Table C-2. Special Function Register (SFR) HWindows

Hex Addr.	Read/ Write	HWindow 0	HWindow 1	HWindow 15
00	Read	ZERO_REG (LO)	ZERO_REG (LO)	ZERO_REG (LO)
	Write	ZERO_REG (LO)	ZERO_REG (LO)	ZERO_REG (LO)
01	Read	ZERO_REG (HI)	ZERO_REG (HI)	ZERO_REG (HI)
	Write	ZERO_REG (HI)	ZERO_REG (HI)	ZERO_REG (HI)
02	Read	AD_RESULT (LO)	Reserved	AD_COMMAND
	Write	AD_COMMAND		AD_RESULT (LO)
03	Read	AD_RESULT (HI)	AD_TIME	HSI_MODE
	Write	HSI_MODE	AD_TIME	AD_RESULT (HI)
04	Read	HSI_TIME (LO)	PTSSEL (LO)	HSO_TIME (LO)
	Write	HSO_TIME (LO)	PTSSEL (LO)	HSI_TIME (LO)
05	Read	HSI_TIME (HI)	PTSSEL (HI)	HSO_TIME (HI)
	Write	HSO_TIME (HI)	PTSSEL (HI)	HSI_TIME (HI)
06	Read	HSI_STATUS	PTSSRV (LO)	HSO_COMMAND
	Write	HSO_COMMAND	PTSSRV (LO)	HSI_STATUS, bits 0, 2, 4, 6
07	Read	SBUF (RX)	PTSSRV (HI)	SBUF (TX)
	Write	SBUF (TX)	PTSSRV (HI)	SBUF(RX)
08	Read	INT_MASK	INT_MASK	INT_MASK
	Write	INT_MASK	INT_MASK	INT_MASK
09	Read	INT_PEND	INT_PEND	INT_PEND
	Write	INT_PEND	INT_PEND	INT_PEND
0A	Read	TIMER1 (LO)	Reserved	WATCHDOG
	Write	WATCHDOG		TIMER1 (LO)
0B	Read	TIMER1 (HI)	Reserved	IOC2, except bit 7 (Note 1)
	Write	IOC2		TIMER1 (HI)
0C	Read	TIMER2 (LO)	IOC3 (Note 2)	T2CAPTURE (LO)
	Write	TIMER2 (LO)	IOC3 (Note 2)	T2CAPTURE (LO)
0D	Read	TIMER2 (HI)	Reserved	T2CAPTURE (HI)
	Write	TIMER2 (HI)		T2CAPTURE (HI)

Table C-2. Special Function Register (SFR) HWindows (Continued)

Hex Addr.	Read/ Write	HWindow 0	HWindow 1	HWindow 15
0E	Read	IOPORT0	Reserved	Reserved
	Write	BAUD_RATE		
0F	Read	IOPORT1	Reserved	Reserved
	Write	IOPORT1		
10	Read	IOPORT2	Reserved	Reserved
	Write	IOPORT2		
11	Read	SP_STAT	Reserved	SP_CON
	Write	SP_CON		SP_STAT (Note 3)
12	Read	INT_PEND1	INT_PEND1	INT_PEND1
	Write	INT_PEND1	INT_PEND1	INT_PEND1
13	Read	INT_MASK1	INT_MASK1	INT_MASK1
	Write	INT_MASK1	INT_MASK1	INT_MASK1
14	Read	WSR	WSR	WSR
	Write	WSR	WSR	WSR
15	Read	IOS0	Reserved	IOC0, except bit 1 (Note 1)
	Write	IOC0		IOS0, except bits 6 and 7
16	Read	IOS1	PWM2_CONTROL	IOC1
	Write	IOC1	PWM2_CONTROL	IOS1, except bits 6 and 7 (Note 3)
17	Read	IOS2	PWM1_CONTROL	PWM0_CONTROL
	Write	PWM0_CONTROL	PWM1_CONTROL	IOS2 (Note 3)
18	Read	SP (LO)	SP (LO)	SP (LO)
	Write	SP (LO)	SP (LO)	SP (LO)
19	Read	SP (HI)	SP (HI)	SP (HI)
	Write	SP (HI)	SP (HI)	SP (HI)

NOTES:

1. IOC2.7 and IOC0.1 are not latched and will read as “1” (precharged bus).
2. The IOC3 register was previously called T2CONTROL or T2CNTC.
3. Writing to the SP_STAT, IOS1, and IOS2 registers in HWindow 15 sets the status bits, but does not cause interrupts.

Table C-3. SFR Name, Address, and Reset Status

Register Mnemonic	Register Name	Hex Addr.	Binary Reset Value
AD_COMMAND	A/D Command Register	02	XXXX XXXX
AD_RESULT	A/D Result Register	03, 02	0111 1111 1100 0000
AD_TIME	A/D Time Register	03	1111 1111
BAUD_RATE	Baud Rate Register	0E	0000 00X0
HSI_MODE	HSI Mode Register	03	1111 1111
HSI_STATUS	HSI Status Register	06	X0X0 X0X0
HSI_TIME	HSI Time Register	05, 04	XXXX XXXX XXXX XXXX
HSO_COMMAND	HSO Command Register	06	XXXX XXXX
HSO_TIME	HSO Time Register	05, 04	XXXX XXXX XXXX XXXX
INT_MASK	Interrupt Mask Register	08	0000 0000
INT_MASK1	Interrupt Mask Register 1	13	0000 0000
INT_PEND	Interrupt Pending Register	09	0000 0000
INT_PEND1	Interrupt Pending Register 1	12	0000 0000
IOC0	I/O Control Register 0	15	0000 00X0
IOC1	I/O Control Register 1	16	0010 0001
IOC2	I/O Control Register 2	0B	X000 0000
IOC3	I/O Control Register 3	0C	**KC-B** 1111 0010 **KC-C** **KD** 1111 0000
IOPORT0	Port 0 Register	0E	XXXX XXXX
IOPORT1	Port 1 Register	0F	1111 1111

Table C-3. SFR Name, Address, and Reset Status (Continued)

Register Mnemonic	Register Name	Hex Addr.	Binary Reset Value			
IOPORT2	Port 2 Register	10			1100	0001
IOPORT34	Port 3 and 4 Register	1FFE	1111	1111	1111	1111
IOS0	I/O Status Register 0	15			0000	0000
IOS1	I/O Status Register 1	16			0000	0000
IOS2	I/O Status Register 2	17			0000	0000
PTSSEL	PTS Select Register	05, 04	0000	0000	0000	0000
PTSSRV	PTS Service Register	07, 06	0000	0000	0000	0000
PWM0_CONTROL	PWM0 Control Register	17			0000	0000
PWM1_CONTROL	PWM1 Control Register	16			0000	0000
PWM2_CONTROL	PWM2 Control Register	17			0000	0000
SBUF (RX/TX)	Serial Port Buffer	07			0000	0000
SP	Stack Pointer	19, 18	0001	1101	0001	1000
SP_CON	Serial Port Control Register	11			0000	1011
SP_STAT	Serial Port Status Register	11			0000	1011
T2CAPTURE	Timer 2 Capture Register	0D, 0C	XXXX	XXXX	XXXX	XXXX
TIMER1	Timer 1 Value Register	0B, 0A	0000	0000	0000	0000
TIMER2	Timer 2 Value Register	0D, 0C	0000	0000	0000	0000
WATCHDOG	Watchdog Timer Register	0A			XXXX	XXXX
WSR	Window Select Register	14			XXXX	0000
ZERO_REG	Zero Register	01, 00	0000	0000	0000	0000

Table C-4. 8XC196KC/KD Interrupt Vector Sources, Locations, and Priorities

Number	Interrupt Vector	Source(s)	Interrupt Vector Location	PTS Vector Location	Priority (1)
Special	Unimplemented Opcode	Unimplemented Opcode	2012H	—	—
Special	Software Trap	TRAP Instruction	2010H	—	—
INT15	NMI **(2)**	NMI	203EH	—	15
Each of the following, maskable interrupts can be assigned to the PTS. Any PTS interrupt has priority over all other maskable interrupts.					
INT14	HSI FIFO Full	HSI FIFO Full	203CH	205CH	14
INT13	EXTINT1 **(2)**	P2.2	203AH	205AH	13
INT12	Timer 2 Overflow	Timer 2 Overflow	2038H	2058H	12
INT11	Timer 2 Capture **(2)**	Timer 2 Capture	2036H	2056H	11
INT10	HSI FIFO 4	HSI FIFO Fourth Entry	2034H	2054H	10
INT09	Receive	RI Flag **(3)**	2032H	2052H	9
INT08	Transmit	TI Flag **(3)**	2030H	2050H	8
INT07	EXTINT **(2)**	P2.2 or P0.7	200EH	204EH	7
INT06	Serial Port	RI Flag and TI Flag **(4)**	200CH	204CH	6
INT05	Software Timer	Software Timer 0–3 Timer 2 Reset A/D Conversion Start	200AH	204AH	5
INT04	HSI.0 Pin **(2)**	HSI.0	2008H	2048H	4
INT03	High Speed Outputs	HSO.0–HSO.5	2006H	2046H	3
INT02	HSI Data Available	HSI FIFO Full or HSI Holding Reg. Loaded	2004H	2044H	2
INT01	A/D Conversion Complete	A/D Conversion Complete	2002H	2042H	1
INT00	Timer Overflow	Timer 1 or Timer 2	2000H	2040H	0

NOTES:

1. The Unimplemented Opcode and Software Trap interrupts are not prioritized. They go directly to the Interrupt Controller for servicing. NMI has the highest priority of all prioritized interrupts. Any PTS interrupt has priority over all other maskable interrupts.
2. These interrupts can be configured to function as independent, external interrupts.
3. If the Serial interrupt is masked and the Receive and Transmit interrupts are enabled, the RI flag and TI flag generate separate Receive and Transmit interrupts. If 8096BH compatibility is not an issue, this configuration is preferred.
4. If the Receive and Transmit interrupts are masked and the Serial interrupt is enabled, both RI flag and TI flag generate a Serial Port interrupt. This configuration provides compatibility with the 8096BH.

Table C-5. 8XC196KC/KD Interrupt Sources, Vectors, and Register Bits

Interrupt Source	Interrupt Vector(s)	Number	Interrupt Enabled by Setting (1)	Source Selected by (2)
A/D Conversion Complete	A/D Conversion Complete	INT01	INT_MASK.1	—
A/D Conversion Start	Software Timer	INT05	INT_MASK.5	—
HSI FIFO Fourth Entry	HSI FIFO 4	INT10	INT_MASK1.2	—
HSI FIFO Full	HSI FIFO Full	INT14	INT_MASK1.6	—
	HSI Data Available	INT02	INT_MASK.2	IOC1.7 = 1
HSI Holding Register Loaded	HSI Data Available	INT02	INT_MASK.2	IOC1.7 = 0
HSI.0	HSI.0 Pin	INT04	INT_MASK.4	—
HSO.0–HSO.5	High-Speed Output	INT03	INT_MASK.3	—
NMI	NMI	INT15	—	—
P0.7	EXTINT	INT07	INT_MASK.7	IOC1.1 = 1
P2.2	EXTINT	INT07	INT_MASK.7	IOC1.1 = 0
	EXTINT1	INT13	INT_MASK1.5	—
RI Flag	Receive	INT09	INT_MASK1.1	—
	Serial Port	INT06	INT_MASK.6	—
Software Timers 0–3	Software Timer	INT05	INT_MASK.5	—
TI Flag	Transmit	INT08	INT_MASK1.0	—
	Serial Port	INT06	INT_MASK.6	—
Timer 1 Overflow	Timer Overflow	INT00	INT_MASK.0	IOC1.2 = 1
Timer 2 Capture	Timer 2 Capture	INT11	INT_MASK1.3	—
Timer 2 Overflow	Timer Overflow	INT00	INT_MASK.0	IOC1.3 = 1
	Timer 2 Overflow	INT12	INT_MASK1.4	—
Timer 2 Reset	Software Timer	INT05	INT_MASK.5	—
TRAP Instruction	Software Trap	Special	—	—
Unimplemented Opcode	Unimplemented Opcode	Special	—	—

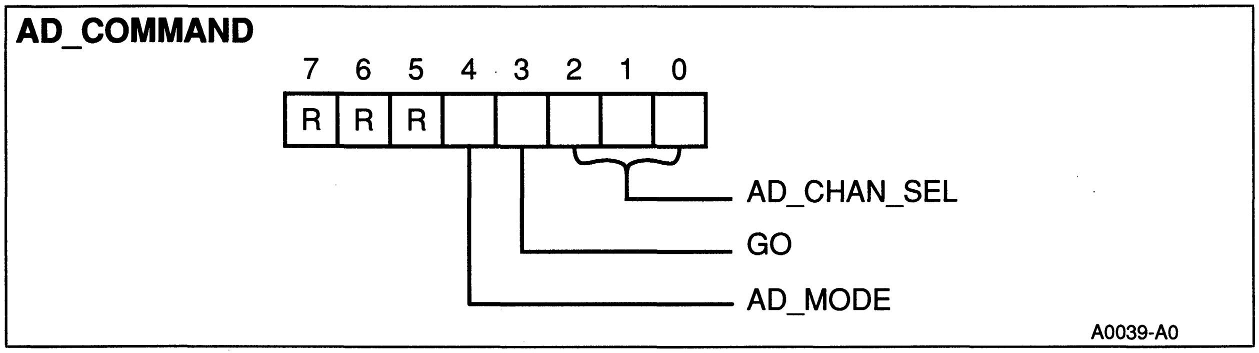
NOTES:

1. This column lists, for each interrupt source, the mask bit that must be set to enable the interrupt. Five interrupt sources can each generate two different interrupts — an 8096BH-compatible interrupt and a new, separate interrupt (HSI FIFO Full, EXTINT1, Receive, Transmit, Timer 2 Overflow). In all cases, only one interrupt should be enabled for each source. (That is, the mask bit should be set for only one of the two possible interrupts).
2. Three of the 8096BH-compatible interrupts (HSI Data Available, EXTINT, and Timer Overflow) can be generated by either of two sources. This column shows the IOC1 register bit and value that selects each source.

A/D Command Register

AD_COMMAND
02H
HWindow 0 (Write), HWindow 15 (Read)

The A/D Command Register selects the A/D channel number to be converted, controls whether the A/D converter starts immediately or with an HSO command, and selects either 8-bit or 10-bit conversion mode.

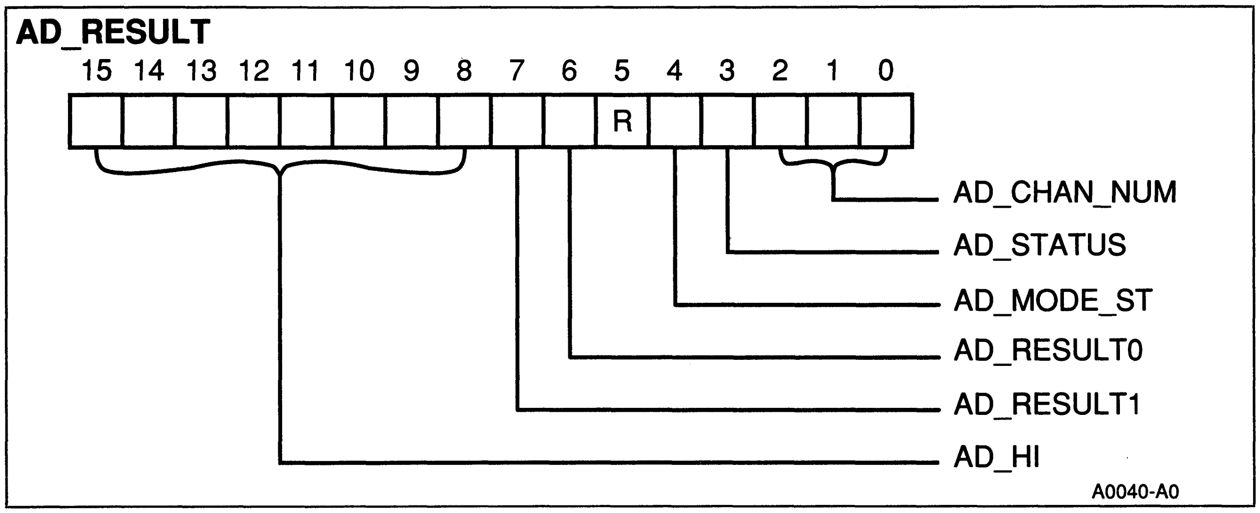


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–2	AD_CHAN_SEL	A/D Channel Selection	XXX	These three bits select the channel number for conversion: **Bit 2 1 0 — Channel** 0 0 0 ACH0 0 0 1 ACH1 0 1 0 ACH2 0 1 1 ACH3 1 0 0 ACH4 1 0 1 ACH5 1 1 0 ACH6 1 1 1 ACH7
3	GO	A/D Conversion Source	X	This bit determines at what point a conversion is to start: 1= Start immediately 0= HSO initiates conversion
4	AD_MODE	A/D Conversion Mode	X	This bit determines whether an 8-bit or a 10-bit conversion is to be performed. 1= 8-bit conversion 0= 10-bit conversion
5–7	—	—	XXX	Reserved; always write as zero.

A/D Result Register

AD_RESULT
03/02H
HWindow 0 (Read), HWindow 15 (Write)

The AD_RESULT register consists of two bytes. The high byte contains the eight most-significant bits from the A/D converter. The low byte indicates the A/D channel number that was used for the conversion, indicates whether a conversion is currently in progress, and contains the two least-significant bits from a ten-bit A/D conversion. The AD_RESULT register is cleared when a new conversion is started; therefore, to prevent losing data, both bytes must be read before a new conversion starts.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–2	AD_CHAN_NUM	A/D Channel Number	000	The A/D channel number that was used for the conversion.
3	AD_STATUS	A/D Status	0	Indicates the status of the A/D converter. Up to 8 state times are required to set this bit following a start command. When testing this bit, wait at least the 8 state times. 1= A/D conversion is in progress 0= A/D is idle
4	AD_MODE_ST	A/D Conversion Mode	0	Indicates whether this is an 8-bit or a 10-bit conversion. 1= 8-bit conversion 0= 10-bit conversion
5	—	—	0	Reserved; always write as zero.

AD_RESULT

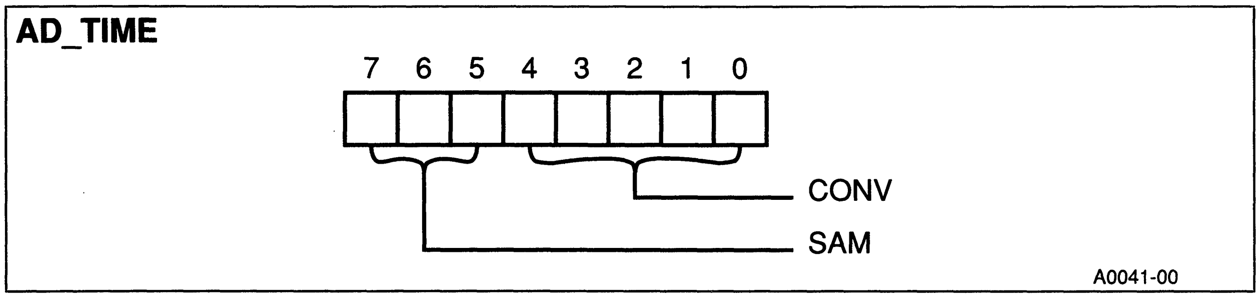
Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
6	AD_RESULT0	Bit 0 of 10-Bit Conv.	1	Bit 0 (the LSB) of a 10-bit conversion.
7	AD_RESULT1	Bit 1 of 10-Bit Conv.	1	Bit 1 of a 10-bit conversion.
8–15	AD_HI	A/D High Result	1111 1110	The 8 most-significant bits of the result from the A/D converter.

A/D Conversion Time Register

AD_TIME
03H
HWindow 1 (Read/Write)

The AD_TIME register programs the sample time and conversion time for the A/D converter. These values are used when IOC2.3 is set. When IOC2.3 is clear (80C196KB-compatible mode) IOC2.4 controls the sample and conversion times.

The sample time (SAM) is the length of time that the analog input channel is actually connected to the sample capacitor. Sample time must be long enough to allow the sample capacitor to charge properly, but not so long that the input will change and cause errors. The conversion time (CONV) determines the length of time required to convert the analog voltage on the sample capacitor to a digital value. Conversion time must be long enough to allow the comparator to settle and resolve the voltage, but not so long that the sample capacitor will discharge and lose accuracy.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–4	CONV	A/D Convert Time	11111	These bits specify the conversion time. CONV must be from 2 to 31, inclusive.
5–7	SAM	A/D Sample Time	111	These bits specify the sample time. SAM must be from 1 to 7, inclusive.

NOTES:

1. The register programs the speed at which the A/D can run — not the speed at which it can convert correctly. Consult the data sheet for recommended values.
2. Initialize the A/D registers in this order: AD_TIME, IOC2, and AD_COMMAND.
3. Do not start a conversion using the AD_TIME register (IOC2.3=1) when an 80C196KB-compatible conversion (IOC2.3=0) is in progress, and vice versa.

AD_TIME

The following formulas are used to compute sample and conversion times.

$$\text{SAM} = \frac{(T_{SAM} \times F_{OSC} - 2)/4}{2}$$

$$\text{CONV} = \left[\frac{(T_{CONV} \times F_{OSC} - 3)}{2} / B\right] - 1$$

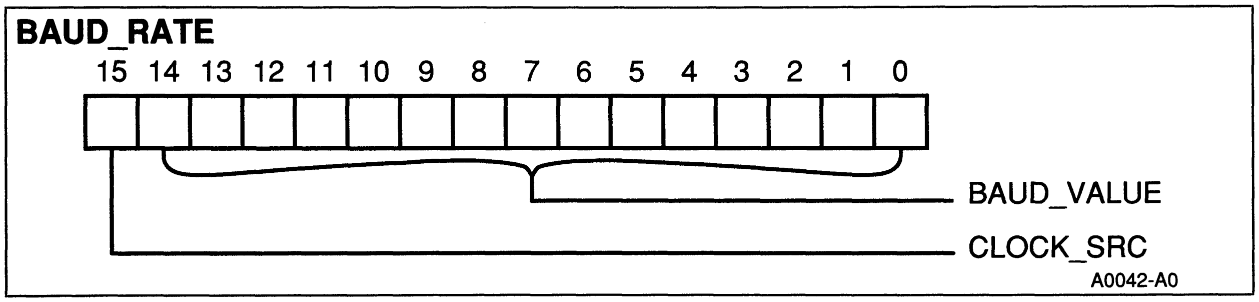
where:

T_{SAM}	is the sample time, in µsec, from the data sheet
T_{CONV}	is the conversion time, in µsec, from the data sheet
F_{OSC}	is the XTAL1 frequency, in MHz
B	is the number of bits to be converted (8 or 10)

Baud Rate Register

BAUD_RATE
0EH
HWindow 0 (Write)

The Baud Rate register selects the serial port baud rate and clock source. It must be written with two bytes, the least-significant byte first. The most-significant bit selects the clock source. The lower 15 bits represent BAUD_VALUE, an unsigned integer that determines the baud rate. BAUD_VALUE has a maximum value of 32,767 and can equal zero only when using XTAL1 in asynchronous modes 1, 2, and 3.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–14	BAUD_VALUE	Baud Rate	0000 00X0 0000 00x	These bits constitute the BAUD_VALUE. Load the least-significant byte first.
15	CLOCK_SRC	Serial Port Clock Source	0	This bit determines whether the serial port is clocked from an internal or an external source. 1= XTAL1 (internal source) 0= T2CLK (external source)

BAUD_RATE

The following formulas are used in determining baud rates.

Synchronous Mode 0: BAUD_VALUE = $\frac{F_{OSC}}{\text{Baud Rate} \times 8} - 1$ or $\frac{\text{T2CLK}}{\text{Baud Rate}}$

Asynchronous Modes 1, 2, and 3: BAUD_VALUE = $\frac{F_{OSC}}{\text{Baud Rate} \times 16} - 1$ or $\frac{\text{T2CLK}}{\text{Baud Rate} \times 8}$

where:

F_{OSC} is the XTAL1 frequency, in MHz

Common baud rate values using XTAL1 at 16 MHz are shown below.

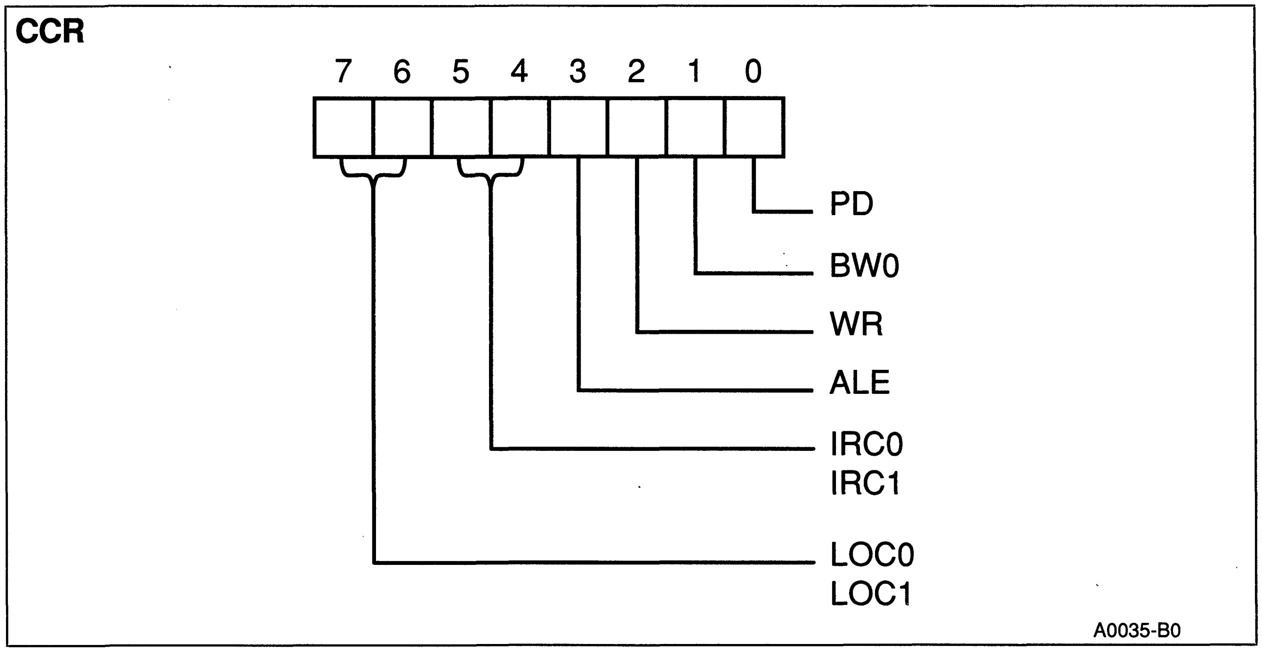
Baud	BAUD_VALUE	
Rate	Mode 0	Modes 1, 2, 3
9600	8340H	8067H
4800	8682H	80CFH
2400	8D04H	81A0H
1200	9A0AH	8340H
300	E82BH	8D04H

Chip Configuration Register CCR

The Chip Configuration Register (CCR) controls Powerdown mode, bus width, bus control signals, internal READY mode, and internal memory protection.

In normal operating mode, the CCR is loaded from the Chip Configuration Byte (CCB) at location 2018H in either internal or external memory, depending on the state of the EA# pin. (EA# low selects external memory; EA# high selects internal memory.) In programming mode, the CCR is loaded from the Programming Chip Configuration Byte (PCCB). The CCB or PCCB is the first byte fetched from memory after a device reset. The CCR is loaded only once during the reset sequence; once it is loaded, the CCR cannot be changed until the next device reset.

If the READY pin is pulled low during the CCR fetch, the bus controller automatically inserts a maximum of three wait states into the CCR bus cycle. This allows a CCR fetch from slow memory. CCR.4 and CCR.5 control the number of wait states inserted into the bus cycle.



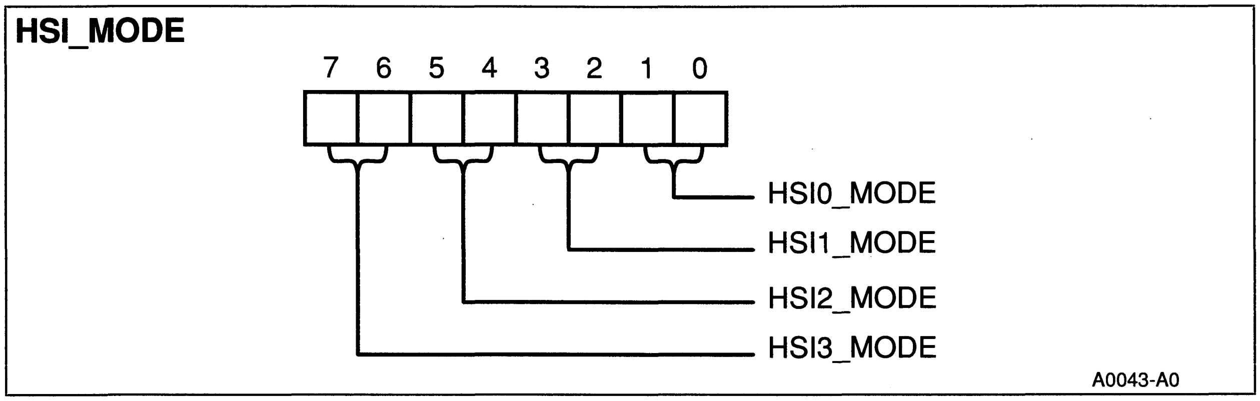
CCR

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	PD	Powerdown Enable	1	Controls whether the IDLPD #2 instruction causes the device to enter Powerdown mode. Clearing this bit at reset can prevent accidental entry into Powerdown mode. 1= enable Powerdown mode 0= disable Powerdown mode
1	BW0	Buswidth Control	1	Selects dynamic or 8-bit bus width. 1= dynamic bus width; bus width is controlled by the BUSWIDTH pin BUSWIDTH=1, 16-bit bus BUSWIDTH=0, 8-bit bus 0= the device is locked into 8-bit mode and the BUSWIDTH pin is ignored
2	WR	Select Write Strobe Mode	1	Selects the write strobe signals to be generated for 16-bit cycles: 1= WR# and BHE# are generated in Standard Bus and Address Valid Strobe modes. 0= WRL# and WRH# are generated in Write Strobe and Address Valid with Write Strobe modes.
3	ALE	Select Address Valid Strobe Mode	1	Selects the address valid signals to be generated. 1= ALE is generated to latch the valid address in Standard Bus and Write Strobe modes. 0= ADV# is generated in place of ALE and can be used as a simple chip select for external memory.
4–5	IRC0–IRC1	Internal Ready Control	10	Limit the number of wait states that can be inserted while the READY pin is held low. Wait states are inserted into the bus cycle either until the READY pin is pulled high or until this internal number is reached. **IRC1 IRC0 Max. Wait States** 0 0 1 0 1 2 1 0 3 1 1 READY pin controlled
6–7	LOC0–LOC1	Lock Bits	00	Determine the programming protection scheme for internal memory. **LOC1 LOC0 Protection** 0 0 read and write protect 0 1 read protect only 1 0 write protect only 1 1 no protection

HSI Mode Register

HSI_MODE
03H
HWindow 0 (Write), HWindow 15 (Read)

The HSI Mode register controls, for each HSI pin, which of four types of events trigger a capture into the HSI FIFO: each positive transition, each negative transition, every transition (both positive and negative), or a series of eight positive transitions. The pins must be individually enabled through the IOC0 register.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–1	HSI0_MODE	HSI.0 Mode	11	HSI.0 transition mode
2–3	HSI1_MODE	HSI.1 Mode	11	HSI.1 transition mode
4–5	HSI2_MODE	HSI.2 Mode	11	HSI.2 transition mode
6–7	HSI3_MODE	HSI.3 Mode	11	HSI.3 transition mode

Each two-bit field defines the transition mode for the corresponding pin:

Transition Mode Encoding

Bit 1	Bit 0	Description
0	0	Eight positive transitions trigger a capture into the HSI FIFO.
0	1	Each positive transition triggers a capture into the HSI FIFO.
1	0	Each negative transition triggers a capture into the HSI FIFO.
1	1	Every transition (both positive and negative) triggers a capture into the HSI FIFO.

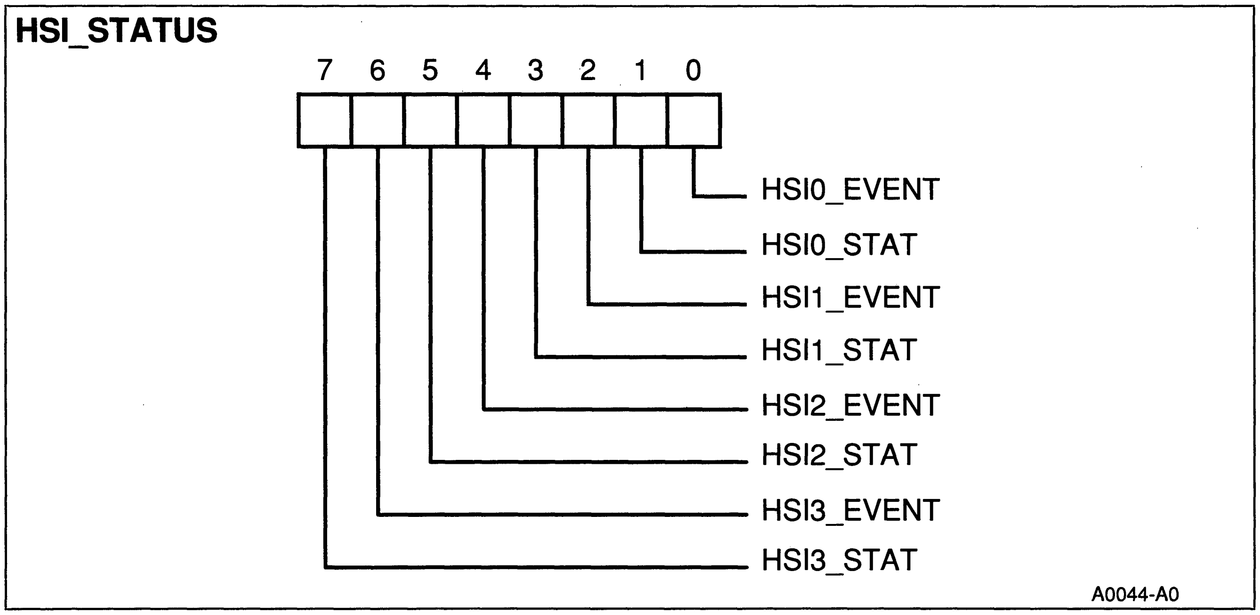
HSI Status Register

HSI_STATUS
06H
HWindow 0 (Read), HWindow 15 (Write bits 0, 2, 4, 6)

The HSI_STATUS register indicates HSI event status and current pin states. The HSI_TIME register contains the associated time tag. Reading HSI_TIME unloads the holding register. If you read HSI_TIME before HSI_STATUS, the status information associated with the HSI_TIME time tag is lost.

If the HSI holding register contains no events, the event status bits in this register are indeterminate; however, the pin state bits can be read at any time. This allows reading the HSI pins as inputs even when they are not enabled to the HSI unit. Writing to HSI_STATUS in HWindow 15 sets the event status bits but does not affect the pin state bits. Note that a current pin state is not necessarily related to the corresponding event status bit, since other events may have occurred since the pin state was last written.

The IOC0 register controls the alternate functions of HSI.0–HSI.3, and the HSI_MODE register controls the pin events that will trigger a capture into the HSI FIFO.



HSI_STATUS

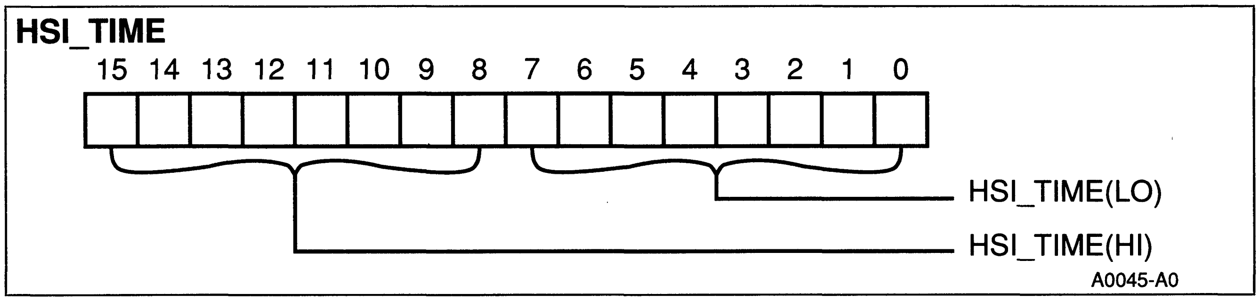
Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	HSI0_EVENT	HSI.0 Pin Event	0	1= an event occurred on pin HSI.0 0= no event
1	HSI0_STAT	HSI.0 Pin Status	X	Current state of the HSI.0 pin.
2	HSI1_EVENT	HSI.1 Pin Event	0	1= an event occurred on pin HSI.1 0= no event
3	HSI1_STAT	HSI.1 Pin Status	X	Current state of the HSI.1 pin.
4	HSI2_EVENT	HSI.2 Pin Event	0	1= an event occurred on pin HSI.2 0= no event
5	HSI2_STAT	HSI.2 Pin Status	X	Current state of the HSI.2 pin.
6	HSI3_EVENT	HSI.3 Pin Event	0	1= an event occurred on pin HSI.3 0= no event
7	HSI3_STAT	HSI.3 Pin Status	X	Current state of the HSI.3 pin.

HSI Time Register

HSI_TIME
04, 05H
HWindow 0 (Read), HWindow 15 (Write)

The HSI Time register contains the time, with respect to the Timer 1 count value, at which an HSI event was triggered. When an event occurs on an HSI pin, the HSI_TIME register is loaded with the current 16-bit Timer 1 value. This time is stored into the HSI FIFO, along with the four event status bits in the HSI_STATUS register.

Reading the HSI_TIME register unloads the holding register. If you read HSI_TIME before HSI_STATUS, the status information associated with the HSI_TIME time tag is lost. If HSI holding register contains no events, HSI_TIME is indeterminate. Writing to HSI_TIME in HWindow 15 loads the holding register, overwriting any other data.

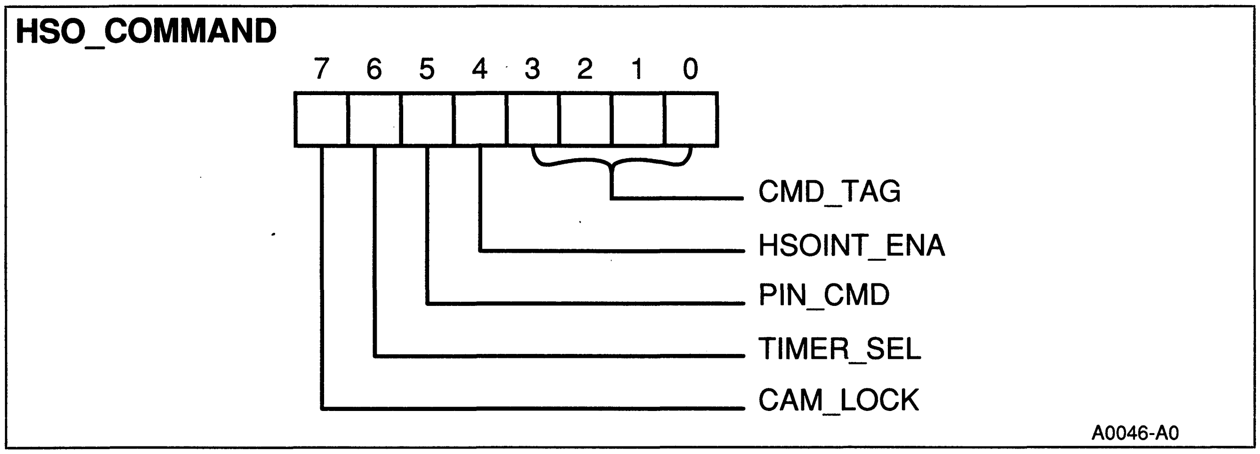


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	HSI_TIME (LO)	HSI Event Time	XXXX XXXX	The lower eight bits of the HSI event time.
8–15	HSI_TIME (HI)	HSI Event Time	XXXX XXXX	The upper eight bits of the HSI event time.

HSO Command Register

HSO_COMMAND
06H
HWindow 0 (Write), HWindow 15 (Read)

The HSO module can trigger events at specific times with minimal CPU overhead. The HSO Command Register determines what event or events will occur within the HSO module at the time specified in the HSO_TIME register.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–3	CMD_TAG	HSO Commands	XXXX	Determines what event or events will occur at the time specified in HSO_TIME. (See the "CMD_TAG Encoding" table.)
4	HSOINT_ENA	Enable/ Disable HSO Interrupt	X	Determines whether an HSO event generates an interrupt. 1= generate an interrupt 0= no interrupt When this bit is set, programmed HSO pin events generate the High-Speed Output interrupt (INT03, 2006H) and internal events generate the Software Timer interrupt (INT05, 200AH). When an interrupt is generated, the HSO event status bits in the IOS1 and IOS2 registers must be interrogated to determine which event caused the interrupt.
5	PIN_CMD	Set/Clear Selected HSO Pin	X	Determines whether the effect of a CMD_TAG command sets or clears the specified pin or pins. 1= Set the pin(s) 0= Clear the pin(s)

HSO_COMMAND

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
6	TIMER_SEL	Select Timer 1/ Timer 2	X	Selects the timer reference for the HSO command. 1= Select Timer 2 0= Select Timer 1
7	CAM_LOCK	Lock Entry Into CAM	X	When IOC2.6 is set (command locking enabled), this bit controls whether the HSO command is locked into the CAM or cleared after execution. 1= Lock command into CAM 0= Clear command from CAM after execution Writing a "1" to IOC2.7 clears all entries (locked or not) from the CAM, as does a chip reset.

CMD_TAG Encoding

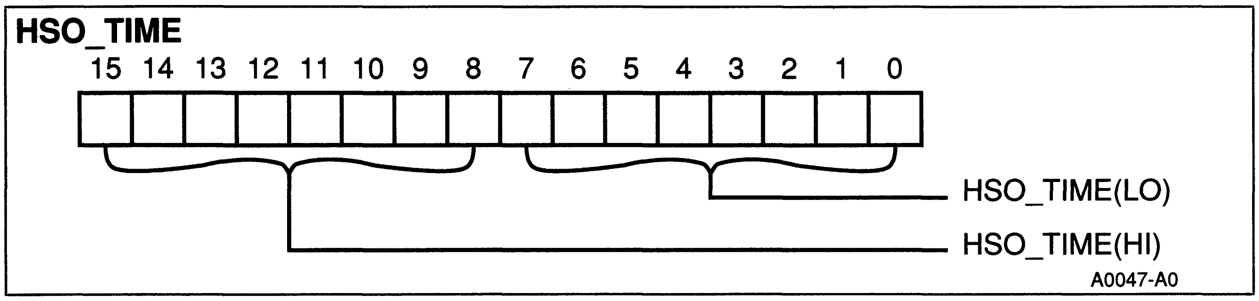
Bit 3	Bit 2	Bit 1	Bit 0	Command Mnemonic	Definition
0	0	0	0	HSO0	Switch High-Speed Output 0
0	0	0	1	HSO1	Switch High-Speed Output 1
0	0	1	0	HSO2	Switch High-Speed Output 2
0	0	1	1	HSO3	Switch High-Speed Output 3
0	1	0	0	HSO4	Switch High-Speed Output 4
0	1	0	1	HSO5	Switch High-Speed Output 5
0	1	1	0	HSO01 *	Switch High-Speed Outputs 0 and 1
0	1	1	1	HSO23 *	Switch High-Speed Outputs 2 and 3
1	0	0	0	SWT0	Program Software Timer 0
1	0	0	1	SWT1	Program Software Timer 1
1	0	1	0	SWT2	Program Software Timer 2
1	0	1	1	SWT3	Program Software Timer 3
1	1	0	0	HSOALL *	Switch High-Speed Outputs 0, 1, 2, 3, 4, 5
1	1	0	1	—	Reserved; do not use
1	1	1	0	T2RST	Reset Timer 2
1	1	1	1	A_D	Start an A/D Conversion

* In these configurations, two or more pins are set or cleared simultaneously.

HSO Time Register

HSO_TIME
05/04H
HWindow 0 (Write), HWindow 15 (Read)

The HSO Time register specifies the time at which an HSO command is to be executed. The command is specified by HSO_COMMAND.0–HSO_COMMAND.3, and the timer reference is selected by HSO_COMMAND.6. When loading events into the CAM, write to the HSO_COMMAND register first, then write to HSO_TIME.

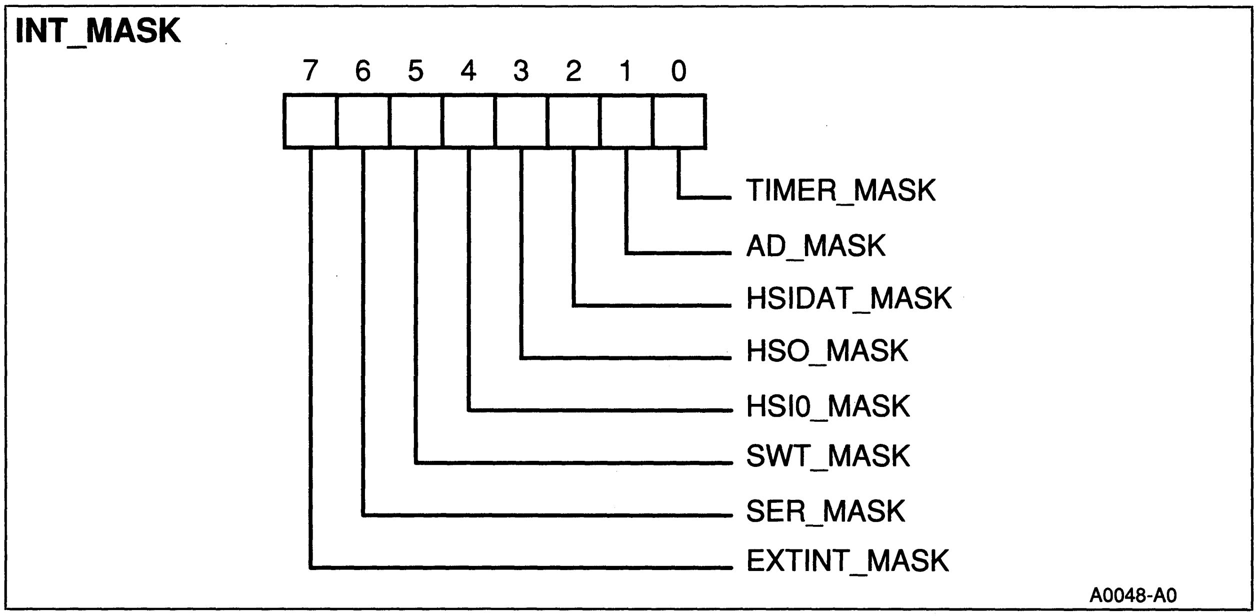


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	HSO_TIME (LO)	HSO Command Execution Time, Low Byte	XXXX	This is the low byte of HSO_TIME, which specifies the time at which an HSO command is to be executed.
8–15	HSO_TIME (HI)	HSO Command Execution Time, High Byte	XXXX	This is the high byte of HSO_TIME, which specifies the time at which an HSO command is to be executed.

Interrupt Mask Register

INT_MASK
08H
All HWindows (Read/Write)

The Interrupt Mask register enables or disables (masks) individual interrupts. (PSW.2 globally enables or disables servicing of all maskable interrupts.) INT_MASK can be read from or written to as a byte register in all HWindows. INT_MASK is the low byte of the Program Status Word (PSW); therefore, PUSHF or PUSHA saves this register on the stack and POPF or POPA restores it.



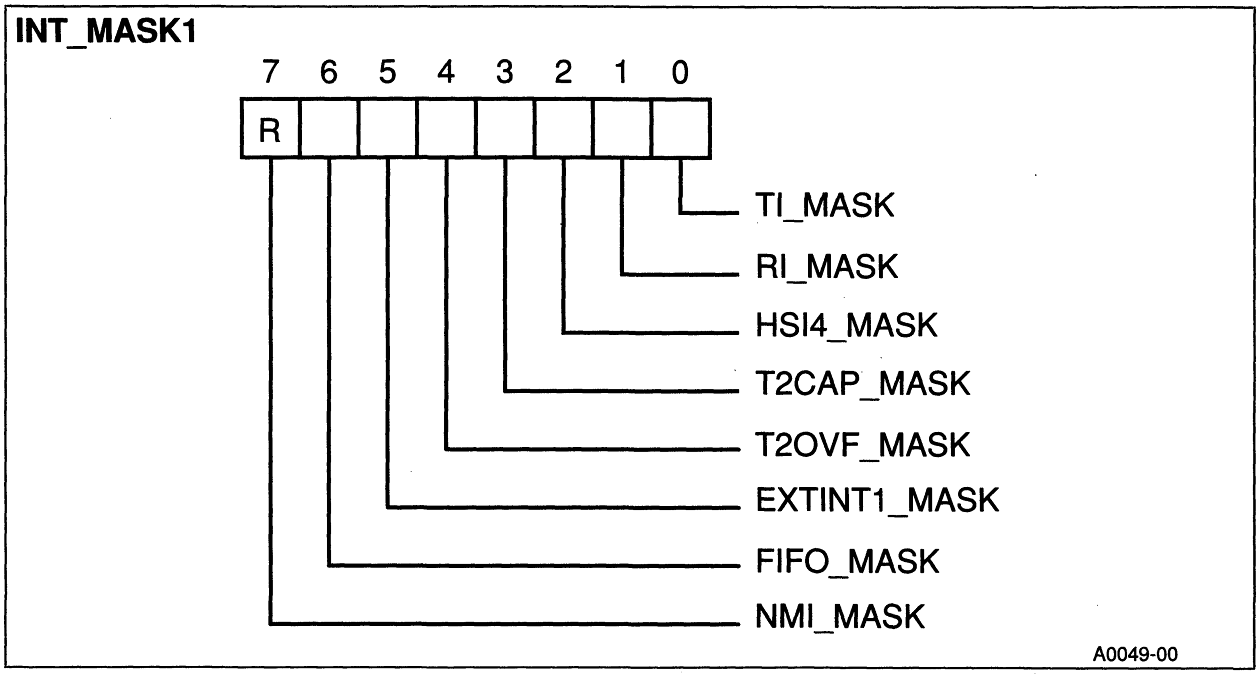
INT_MASK

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	TIMER_MASK	Timer 1 or Timer 2 Overflow	0	Setting this bit enables the Timer Overflow interrupt (INT00, 2000H). Both Timer 1 and Timer 2 can generate the INT00 interrupt. Setting IOC1.2 selects Timer 1 as a source; setting IOC1.3 selects Timer 2 as a source. Timer 2 can generate either INT00 or INT12, but should not be configured for both.
1	AD_MASK	A/D Conversion Complete	0	Setting this bit enables the A/D Conversion Complete interrupt (INT01, 2002H).
2	HSIDAT_MASK	HSI Data Available/ FIFO Full	0	Setting this bit enables the HSI Data Available interrupt (INT02, 2004H). IOC1.7 selects the source of the interrupt.
3	HSO_MASK	HSO Output Event	0	Setting this bit enables the High-Speed Output interrupt (INT03, 2006H).
4	HSI0_MASK	HSI.0 External Interrupt	0	Setting this bit enables the HSI.0 Pin interrupt (INT04, 2008H).
5	SWT_MASK	Software Timer	0	Setting this bit enables the Software Timer interrupt (INT05, 200AH).
6	SER_MASK	Serial Port	0	Setting this bit enables the Serial Port interrupt (INT06, 200CH), which is the 8096BH-compatible configuration. If this bit is set, INTMASK1.0 and INTMASK1.1 should be cleared, disabling the Receive and Transmit interrupts.
7	EXTINT_MASK	EXTINT or P0.7 Interrupt	0	Setting this bit enables the EXTINT interrupt (INT07, 200EH). IOC1.1 selects the interrupt source (P0.7 or P2.2).

Interrupt Mask Register 1

INT_MASK1
13H
All HWindows (Read/Write)

The Interrupt Mask 1 register enables or disables (masks) individual interrupts. (PSW.2 globally enables or disables servicing of all maskable interrupts.) INT_MASK1 can be read from or written to as a byte register in all HWindows. PUSHA saves this register on the stack, and POPA restores it.



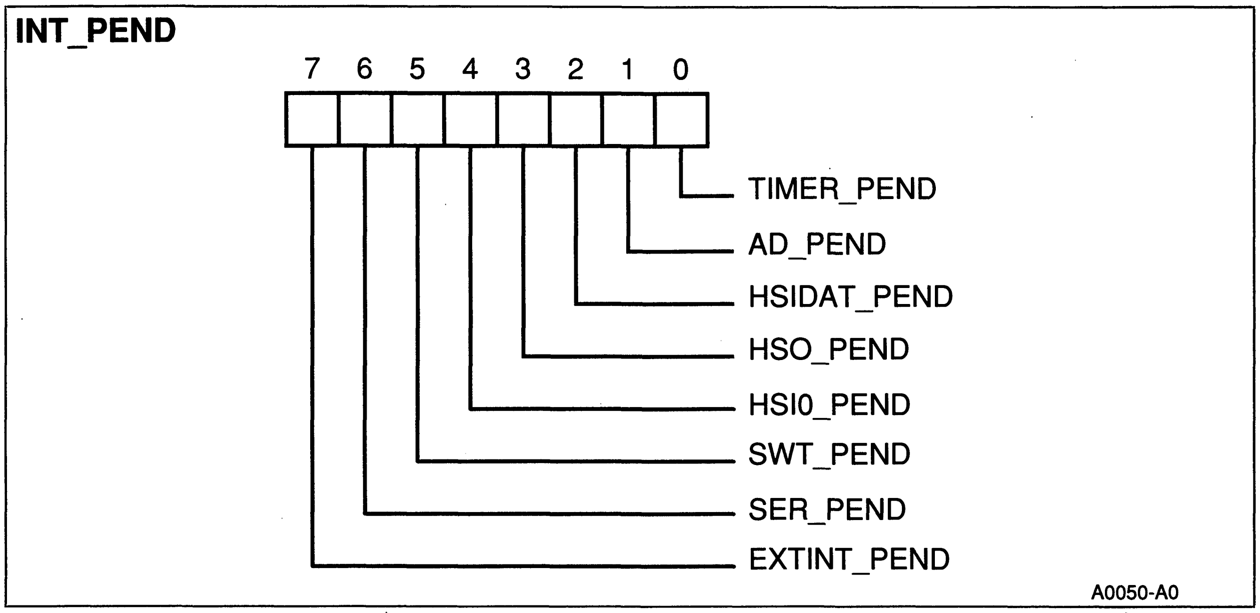
INT_MASK1

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	TI_MASK	Transmit Interrupt	0	Setting this bit enables the Transmit interrupt (INT08, 2030H). If this bit is set, INT_MASK1.1 should also be set and INT_MASK.6 should be cleared.
1	RI_MASK	Receive Interrupt	0	Setting this bit enables the Receive interrupt (INT09, 2032H). If this bit is set, INT_MASK1.0 should also be set and INT_MASK.6 should be cleared.
2	HSI4_MASK	HSI FIFO 4 Interrupt	0	Setting this bit enables the HSI FIFO 4 interrupt (INT10, 2034H).
3	T2CAP_MASK	Timer 2 Capture Interrupt	0	Setting this bit enables the Timer 2 Capture interrupt (INT11, 2036H).
4	T2OVF_MASK	Timer 2 Overflow Interrupt	0	Setting this bit enables the Timer 2 Overflow interrupt (INT12, 2038H). IOC2.5 selects the overflow boundary for INT12.
5	EXTINT1_MASK	EXTINT Pin Interrupt	0	Setting this bit enables the EXTINT1 interrupt (INT13, 203AH). P2.2 can generate either the EXTINT interrupt (INT07) or the EXTINT1 interrupt (INT13).
6	FIFO_MASK	HSI FIFO Full Interrupt	0	Setting this bit enables the HSI FIFO Full interrupt (INT14, 203CH).
7	NMI_MASK	NMI	0	This non-functional mask bit exists for design symmetry. The Nonmaskable Interrupt (NMI) is enabled for both 0 and 1. Always write zero to this bit.

Interrupt Pending Register

INT_PEND
09H
All HWindows (Read/Write)

When hardware detects a pending interrupt, it sets the corresponding bit in INT_PEND or INT_PEND1. When the vector is taken, the hardware clears the pending bit. The INT_PEND register can be read from or written to in all HWindows. Software can generate an interrupt by setting the corresponding interrupt pending bit.



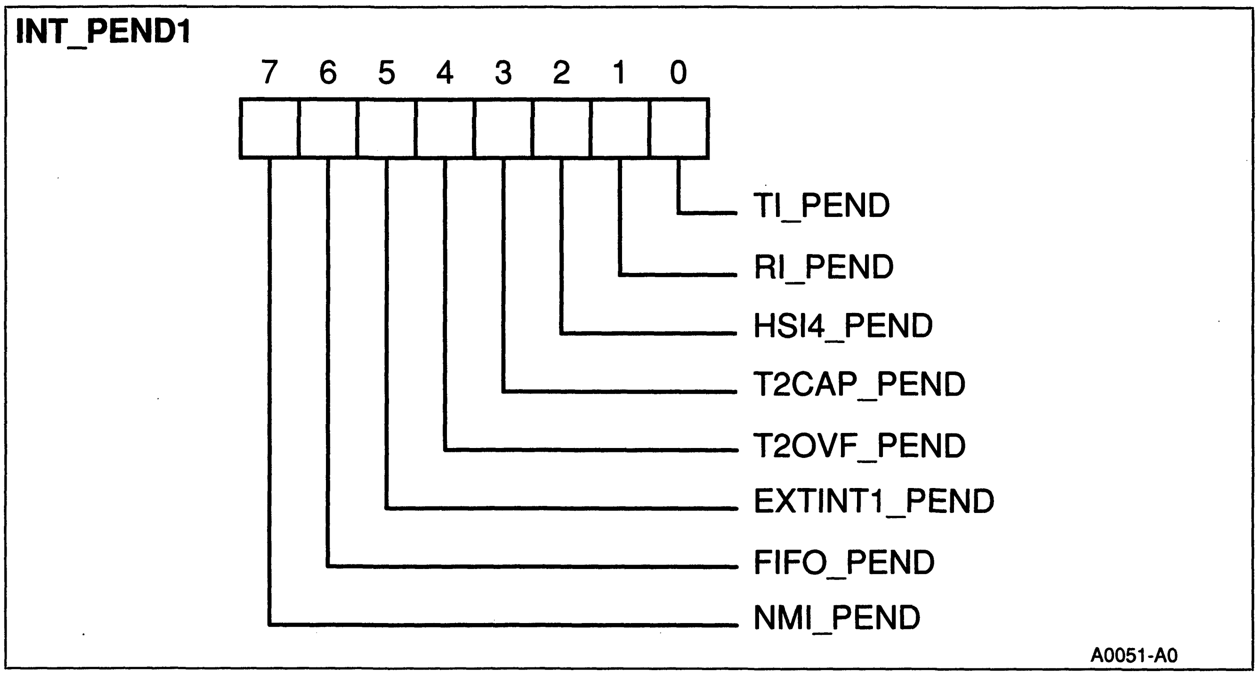
INT_PEND

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	TIMER_PEND	Timer 1 or Timer 2 Overflow	0	When set, this bit indicates a pending Timer Overflow interrupt (INT00). It is cleared when the interrupt vectors to 2000H.
1	AD_PEND	A/D Conversion Complete	0	When set, this bit indicates a pending A/D Conversion Complete interrupt (INT01). It is cleared when the interrupt vectors to 2002H.
2	HSIDAT_PEND	HSI Data Available/ FIFO Full	0	When set, this bit indicates a pending HSI Data Available interrupt (INT02). It is cleared when the interrupt vectors to 2004H.
3	HSO_PEND	HSO Output Event	0	When set, this bit indicates a pending High-Speed Output interrupt (INT03). It is cleared when the interrupt vectors to 2006H.
4	HSI0_PEND	HSI.0 External Interrupt	0	When set, this bit indicates a pending HSI.0 Pin interrupt (INT04). It is cleared when the interrupt vectors to 2008H.
5	SWT_PEND	Software Timer	0	When set, this bit indicates a pending Software Timer interrupt (INT05). It is cleared when the interrupt vectors to 200AH.
6	SER_PEND	Serial Port	0	When set, this bit indicates a pending Serial Port interrupt (INT06). It is cleared when the interrupt vectors to 200CH.
7	EXTINT_PEND	EXTINT Pin or P0.7 Interrupt	0	When set, this bit indicates a pending EXTINT interrupt (INT07). It is cleared when the interrupt vectors to 200EH.

Interrupt Pending Register 1

INT_PEND1
12H
All HWindows (Read/Write)

When hardware detects a pending interrupt, it sets the corresponding bit in INT_PEND or INT_PEND1. When the vector is taken, the hardware clears the pending bit. The INT_PEND1 register can be read from or written to in all HWindows. Software can generate an interrupt by setting the corresponding interrupt pending bit.



INT_PEND1

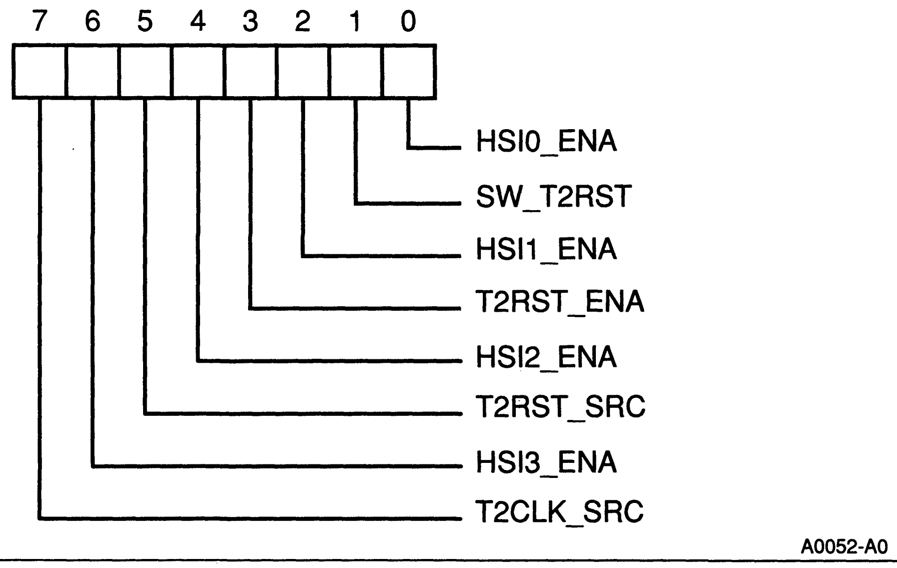
Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	TI_PEND	Transmit Interrupt	0	When set, this bit indicates a pending Transmit interrupt (INT08). It is cleared when the interrupt vectors to 2030H.
1	RI_PEND	Receive Interrupt	0	When set, this bit indicates a pending Receive interrupt (INT09). It is cleared when the interrupt vectors to 2032H.
2	HSI4_PEND	HSI FIFO 4 Interrupt	0	When set, this bit indicates a pending HSI FIFO 4 interrupt (INT10). It is cleared when the interrupt vectors to 2034H.
3	T2CAP_PEND	Timer 2 Capture Interrupt	0	When set, this bit indicates a pending Timer 2 Capture interrupt (INT11). It is cleared when the interrupt vectors to 2036H.
4	T2OVF_PEND	Timer 2 Overflow Interrupt	0	When set, this bit indicates a pending Timer 2 Overflow interrupt (INT12). It is cleared when the interrupt vectors to 2038H.
5	EXTINT1_PEND	EXTINT Pin Interrupt	0	When set, this bit indicates a pending EXTINT1 interrupt (INT13). It is cleared when the interrupt vectors to 203AH.
6	FIFO_PEND	HSI FIFO Full Interrupt	0	When set, this bit indicates a pending HSI FIFO Full interrupt (INT14). It is cleared when the interrupt vectors to 203CH.
7	NMI_PEND	NMI	0	When set, this bit indicates a pending NMI interrupt (INT15). It is cleared when the interrupt vectors to 203EH.

Input/Output Control Register 0

IOC0
15H
HWindow 0 (Write), HWindow 15 (Read)

The IOC0 register selects the external clock and reset sources for Timer 2 and enables or disables the HSI input function of the four HSI pins. When IOC0 is read in HWindow 15, IOC0.1 will always read as "1" because its value is not latched.

IOC0



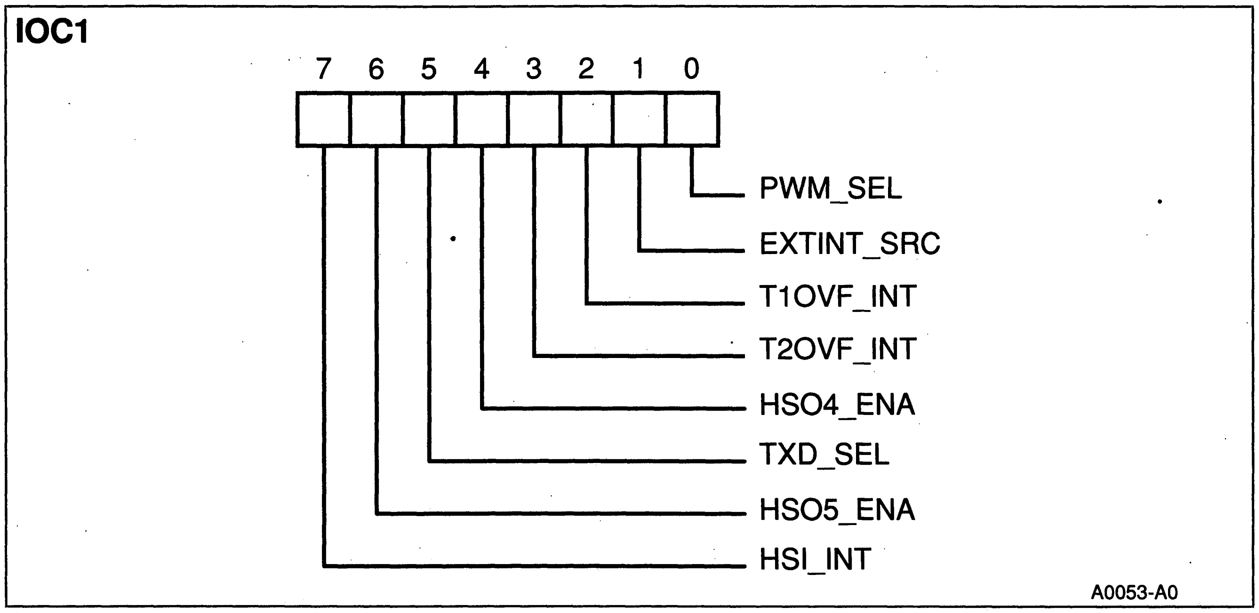
IOC0

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	HSI0_ENA	Enable HSI.0 as HSI input	0	This bit controls whether events on the HSI.0 pin are loaded into the HSI FIFO. 1= enabled as HSI input 0= disabled as HSI input
1	SW_T2RST	Timer 2 Software Reset	X	Writing a "1" to this bit resets Timer 2. This bit will always read back as "1" in HWindow 15. 1= reset Timer 2 each write 0= no action
2	HSI1_ENA	Enable HSI.0 as HSI input	0	This bit controls whether events on the HSI.1 pin are loaded into the HSI FIFO. 1= enabled as HSI input 0= disabled as HSI input
3	T2RST_ENA	Timer 2 External Reset Source	0	This bit enables external reset of Timer 2. The external reset source is the rising edge of either T2RST or HSI.0, as selected by IOC0.5. 1= enable external reset 0= disable
4	HSI2_ENA	Enable HSI.0 as HSI input	0	This bit controls whether events on the HSI.2 pin are loaded into the HSI FIFO. 1= enabled as HSI input 0= disabled as HSI input
5	T2RST_SRC	Timer 2 Reset Source	0	This bit selects the external reset source for Timer 2. IOC0.3 must be set to enable the external reset. 1= HSI.0 pin rising edge 0= T2RST pin (P2.4) rising edge
6	HSI3_ENA	Enable HSI.0 as HSI input	0	This bit controls whether events on the HSI.3 pin are loaded into the HSI FIFO. 1= enabled as HSI input 0= disabled as HSI input
7	T2CLK_SRC	Timer 2 Clock Source	0	This bit selects the external clock source for Timer 2. IOC3.0 must be cleared to enable the external clock source. 1= HSI.1 pin 0= T2CLK pin (P2.3)

Input/Output Control Register 1

IOC1
16H
HWindow 0 (Write), HWindow 15 (Read)

The IOC1 register selects the output functions of P2.5 and P2.0; enables or disables HSO.4 and HSO.5 as outputs; and selects interrupt sources for EXTINT (INT07, 200EH), Timer Overflow (INT00, 2000H), and HSI Data Available (INT02, 2004H).



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	PWM_SEL	Select P2.5/ PWM Output	1	This bit controls whether P2.5 functions as a PWM output pin or as a standard output port pin. 1= PWM output pin 0= standard output port pin

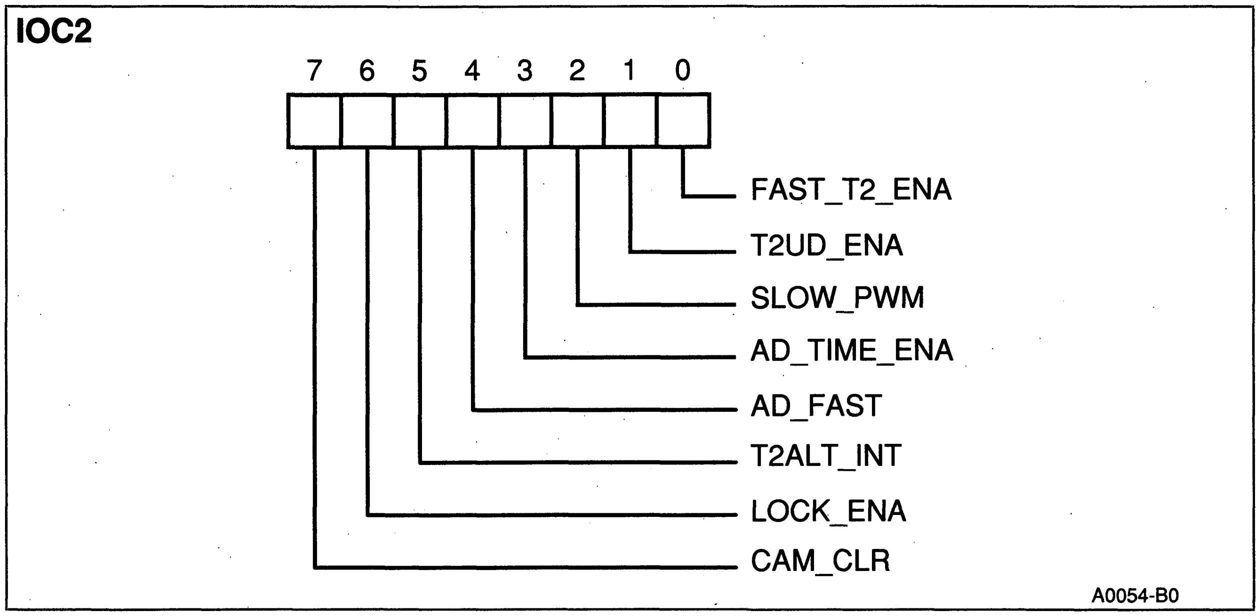
IOC1

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
1	EXTINT_SRC	Select External Interrupt INT07 Source	0	This bit selects the EXTINT external interrupt source (INT07, 200EH). INT_MASK.7 must be set to enable the interrupt. 1= P0.7 0= P2.2
2	T1OVF_INT	Enable Timer 1 Overflow Interrupt	0	Both Timer 1 and Timer 2 can generate the Timer Overflow interrupt (INT00, 2000H). This bit controls whether an overflow on Timer 1 generates the interrupt. INT_MASK.0 must be set to enable the interrupt. IOS1.5 indicates the interrupt status. 1= enable Timer 1 as source 0= disable Timer 1 as source
3	T2OVF_INT	Enable Timer 2 Overflow Interrupt	0	Both Timer 1 and Timer 2 can generate the Timer Overflow interrupt (INT00, 2000H). This bit controls whether an overflow on Timer 2 generates the interrupt. INT_MASK.0 must be set to enable the interrupt. IOS1.4 indicates the interrupt status. 1= enable Timer 2 as source 0= disable Timer 2 as source
4	HSO4_ENA	Enable HSO.4 Pin as Output	0	HSO.4 is multiplexed with the HSI.2 input pin. This bit enables the output function of HSO.4. 1= enabled as output 0= disabled as output
5	TXD_SEL	Select P2.0/ TXD Output	1	This bits controls whether P2.0 functions as the TXD output of the serial port transmitter or as a standard output port pin. 1= serial port TXD output 0= standard output port pin
6	HSO5_ENA	Enable HSO.5 Pin as Output	0	HSO.5 is multiplexed with the HSI.3 input pin. This bit enables the output function of HSO.5. 1= enabled as output 0= disabled as output
7	HSI_INT	Select HSI Interrupt Source	0	This bit selects the source of the HSI Data Available interrupt (INT02, 2004H). INT_MASK.2 must be set to enable the interrupt. 1= HSI FIFO full 0= HSI Holding Register loaded

Input/Output Control Register 2

IOC2
0BH
HWindow 0 (Write), HWindow 15 (Read)

The IOC2 register controls three Timer 2 options, the clock prescalers for the PWM and the A/D converter, and the source for A/D conversion time determination. IOC2 also enables and disables locking commands into the HSO CAM, and it can clear all entries from the HSO CAM.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	FAST_T2_ENA	Enable Timer 2 Fast Increment	0	This bit controls whether Timer 2 operates in fast increment or normal mode. 1= fast increment mode; count every state 0= normal mode; count every eight states When fast increment mode is enabled, do not use Timer 2 as the HSO reference and do not reset Timer 2.
1	T2UD_ENA	Enable Timer 2 Up/Down Count	0	This bit controls whether Timer 2 functions as an up counter or as an up/down counter 1= if P2.6 = 1, count down if P2.6 = 0, count up 0= count up only

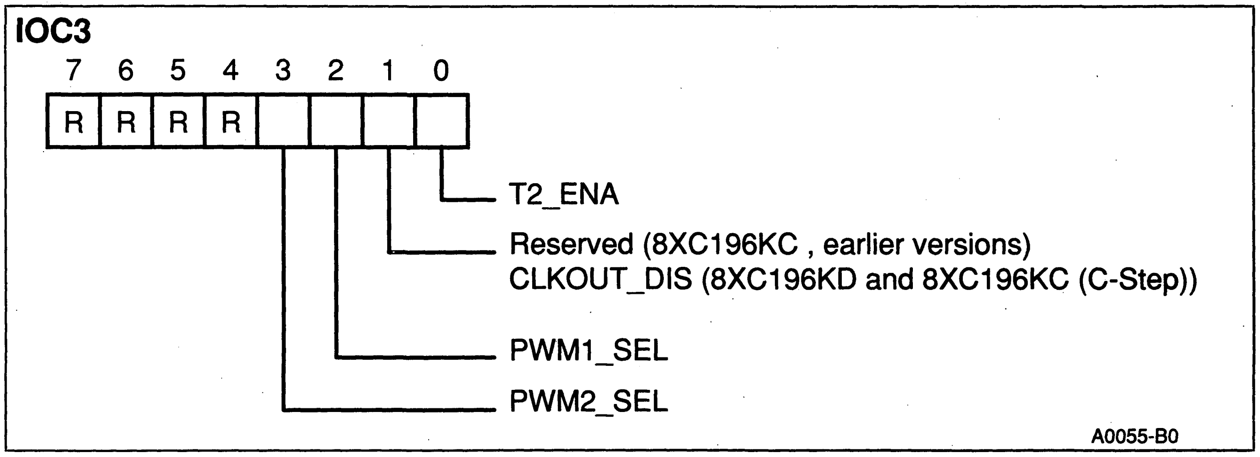
IOC2

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
2	SLOW_PWM	Enable PWM Clock Prescaler	0	This bit controls the PWM output period by enabling or disabling a clock prescaler (divide-by-two) on PWM.1, PWM.2, and PWM.3. 1= enable; PWM output period is 512 state times 0= disable; PWM output period is 256 state times
3	AD_TIME_ENA	Enable AD_TIME Register	0	This bit selects whether the A/D conversion times are controlled by the AD_TIME register or by the fast and normal conversion modes of the 80C196KB. 1= AD_TIME register 0= 80C196KB-compatible mode When this bit is clear, IOC2.4 enables or disables the A/D clock prescaler for complete 80C196KB compatibility.
4	AD_FAST	Disable A/D Clock Prescaler	0	In 80C196KB-compatible mode (IOC2.3 cleared), this bit controls the A/D conversion time period by enabling or disabling the A/D clock prescaler (divide-by-two). 1= disable; conversion time is 89.5 state times, 80C196KB normal mode 0= enable; conversion time is 156.5 state times, 80C196KB fast mode If IOC2.3 is set, this bit is ignored.
5	T2ALT_INT	Select Timer 2 Overflow Boundary	0	This bit selects the overflow boundary for the Timer 2 Overflow interrupt (INT12, 2038H). INT_MASK1.4 must be set to enable the interrupt. 1=7FFFH/8000H boundary 0=0FFFFH/0000H boundary
6	LOCK_ENA	Enable Locked CAM Entries	0	This bit enables and disables command locking. When this bit is set, HSO_COMMAND.7 controls whether individual commands are locked into the CAM or cleared after execution. 1= Enable command locking 0= Disable command locking Writing a "1" to IOC2.7 clears all entries (locked or not) from the CAM, as does a chip reset.
7	CAM_CLR	Clear All CAM Entries	X	Setting this bit clears all entries (even locked entries) from the HSO CAM. This bit is not latched; it will always read as "1" in HWindow 15.

Input/Output Control Register 3

IOC3
0CH
HWindow 1 (Read/Write)

The IOC3 register selects either an internal or an external clock source for Timer 2 and selects the function of pin P1.2 and pin P1.3. (This register was previously called T2CONTROL or T2CNTC.)


IOC3
7 6 5 4 3 2 1 0
R R R R
T2_ENA
Reserved (8XC196KC , earlier versions)
CLKOUT_DIS (8XC196KD and 8XC196KC (C-Step))
PWM1_SEL
PWM2_SEL
A0055-B0

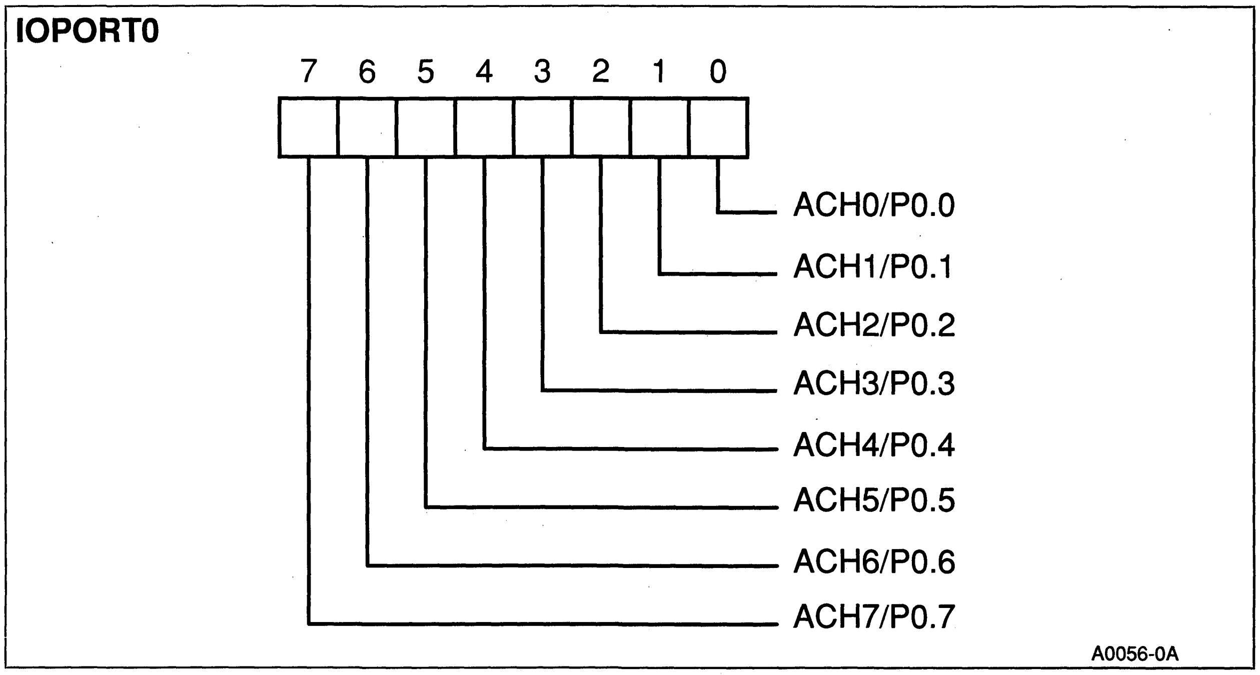
IOC3

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	T2_ENA	Timer 2 Internal Clock Enable	0	This bit controls whether Timer 2 is clocked internally or externally. 1= internal source 0= external source: if IOC0.7=1, HSI.1 if IOC0.7=0, T2CLK (P2.3) IOC2.0 controls whether Timer 2 counts every clock (fast increment mode) or every eight clocks (normal mode).
1	Reserved	Reserved	0	On earlier versions of the 8XC196KC, this bit is reserved; always write as zero. On the 8XC196KC (B-Step), this bit will always read back as "1" in HWindow 1 (precharged bus).
	CLKOUT_DIS	CLKOUT Disable	0	This bit has been implemented on the 8XC196KC (C-Step) and 8XC196KD to enable or disable the CLKOUT signal. This can be used to reduce noise in systems that do not require the CLKOUT signal. The actual value of the bit will read back in HWindow 1. 1= disable CLKOUT 0= enable CLKOUT
2	PWM1_SEL	PWM1 Select	0	This bit selects the P1.3 pin function. 1= PWM1 output 0= quasi-bidirectional port pin When this bit is set, P1.3 has strong pull-ups and pull-downs. This pin can be switched between functions without a device reset.
3	PWM2_SEL	PWM2 Select	0	This bit selects the P1.4 pin function. 1= PWM2 output 0= quasi-bidirectional port pin When this bit is set, P1.4 has strong pull-ups and pull-downs. This pin can be switched between functions without a device reset.
4–7	—	—	1111	Reserved; always write as zero.

Input/Output Port 0 Register

IOPORT0
0EH
HWindow 0 (Read)

Port 0 is an input-only port. To reduce noise, Port 0 is monitored only when a read occurs. The pins can be used as analog inputs to the A/D converter (ACH*x*) and digital inputs (P0.*x*) at the same time, but this is not recommended. Pin P0.7 can also be configured as an external interrupt.

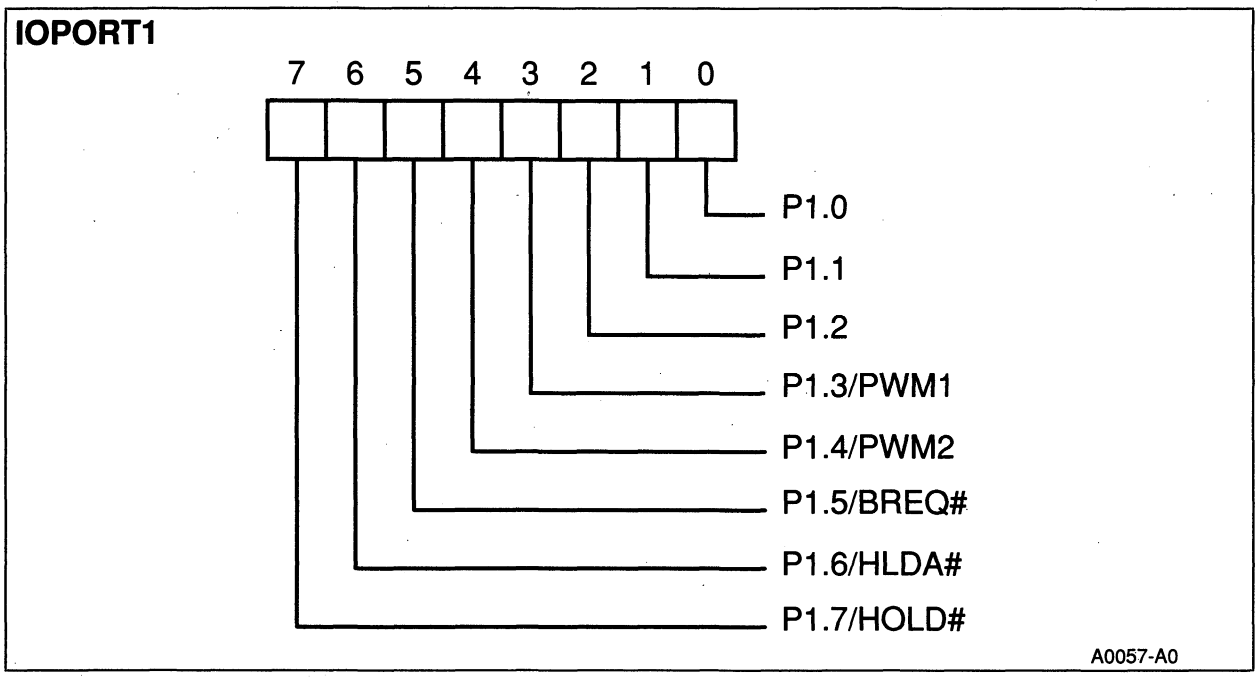


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	ACH0/P0.0	ACH0/P0.0	X	Analog Channel 0/Input Pin P0.0
1	ACH1/P0.1	ACH1/P0.1	X	Analog Channel 1/Input Pin P0.1
2	ACH2/P0.2	ACH2/P0.2	X	Analog Channel 2/Input Pin P0.2
3	ACH3/P0.3	ACH3/P0.3	X	Analog Channel 3/Input Pin P0.3
4	ACH4/P0.4	ACH4/P0.4	X	Analog Channel 4/Input Pin P0.4
5	ACH5/P0.5	ACH5/P0.5	X	Analog Channel 5/Input Pin P0.5
6	ACH6/P0.6	ACH6/P0.6	X	Analog Channel 6/Input Pin P0.6
7	ACH7/P0.7	ACH7/P0.7	X	Analog Channel 7/Input Pin P0.7

Input/Output Port 1 Register

IOPORT1
0FH
HWindow 0 (Read/Write)

All Port 1 pins are quasi-bidirectional unless the alternate function is selected. Three pins share functions with the bus-hold signals; two pins share functions with PWM outputs. When the pins are configured as I/O pins, they can be read or written. When the pins are configured for their alternate functions, they can be read but not written.



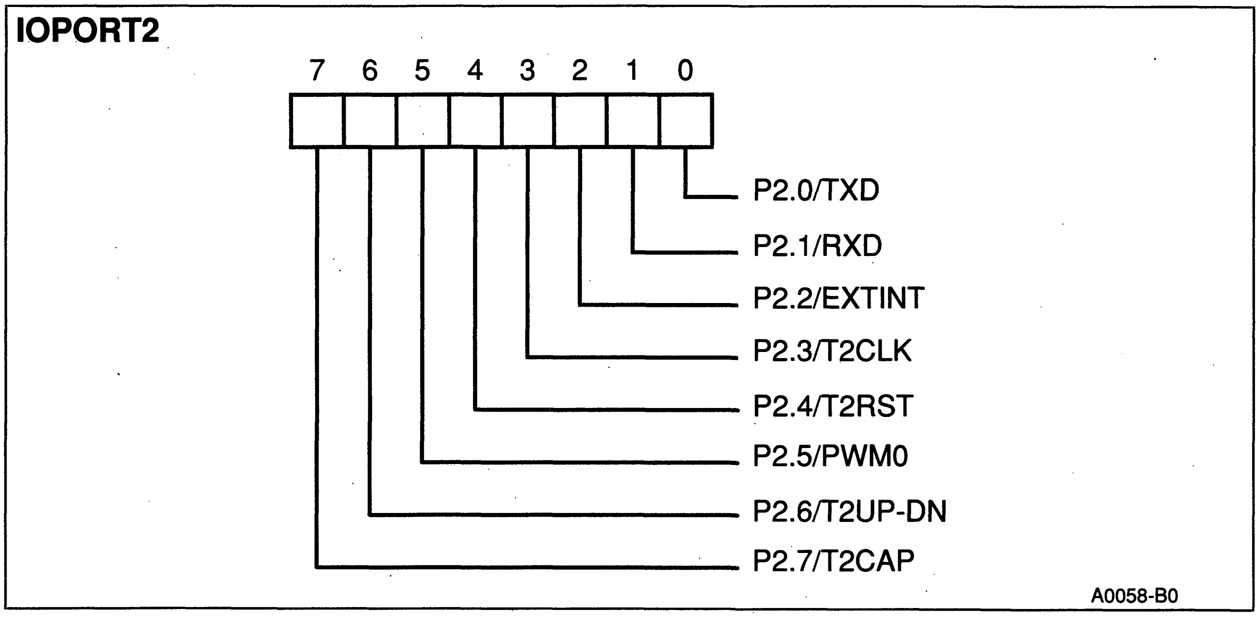
IOPORT1

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	P1.0	P1.0	1	I/O Pin P1.0
1	P1.1	P1.1	1	I/O Pin P1.1
2	P1.2	P1.2	1	I/O Pin P1.2
3	P1.3/PWM1	P1.3/PWM1	1	I/O Pin P1.3/PWM1 Output Setting IOC3.2 enables P1.3 as the PWM1 output pin.
4	P1.4/PWM2	P1.4/PWM2	1	I/O Pin P1.4/PWM2 Output Setting IOC3.3 enables P1.4 as the PWM2 output pin.
5	P1.5/BREQ#	P1.5/ BREQ#	1	I/O Pin P1.5/Bus Request Setting WSR.7 enables P1.5 as BREQ#. Once the HOLD protocol is enabled, the pin functions as BREQ# until the device is reset. BREQ# is activated as an output when the bus controller has a pending external memory cycle. Once BREQ# is asserted, it remains asserted until the HOLD# is removed.
6	P1.6/HLDA#	P1.6/ HLDA#	1	I/O Pin P1.6/Hold Acknowledge Setting WSR.7 enables P1.6 as HLDA#. Once the HOLD protocol is enabled, the pin functions as HLDA# until the device is reset. HLDA# is activated as an output when the 8XC196KC/KD releases the bus in response to another device asserting HOLD#.
7	P1.7/HOLD#	P1.7/ HOLD#	1	I/O Pin P1.7/Hold Input Setting WSR.7 enables P1.7 as HOLD#. Once the HOLD protocol is enabled, the pin functions as HOLD# until the device is reset. HOLD# is activated as an input by a device to request control of the bus.

Input/Output Port 2 Register

IOPORT2
10H
HWindow 0 (Read/Write)

Port 2 contains input-only, output-only, and quasi-bidirectional port pins. The alternate functions must be enabled through the appropriate control registers.



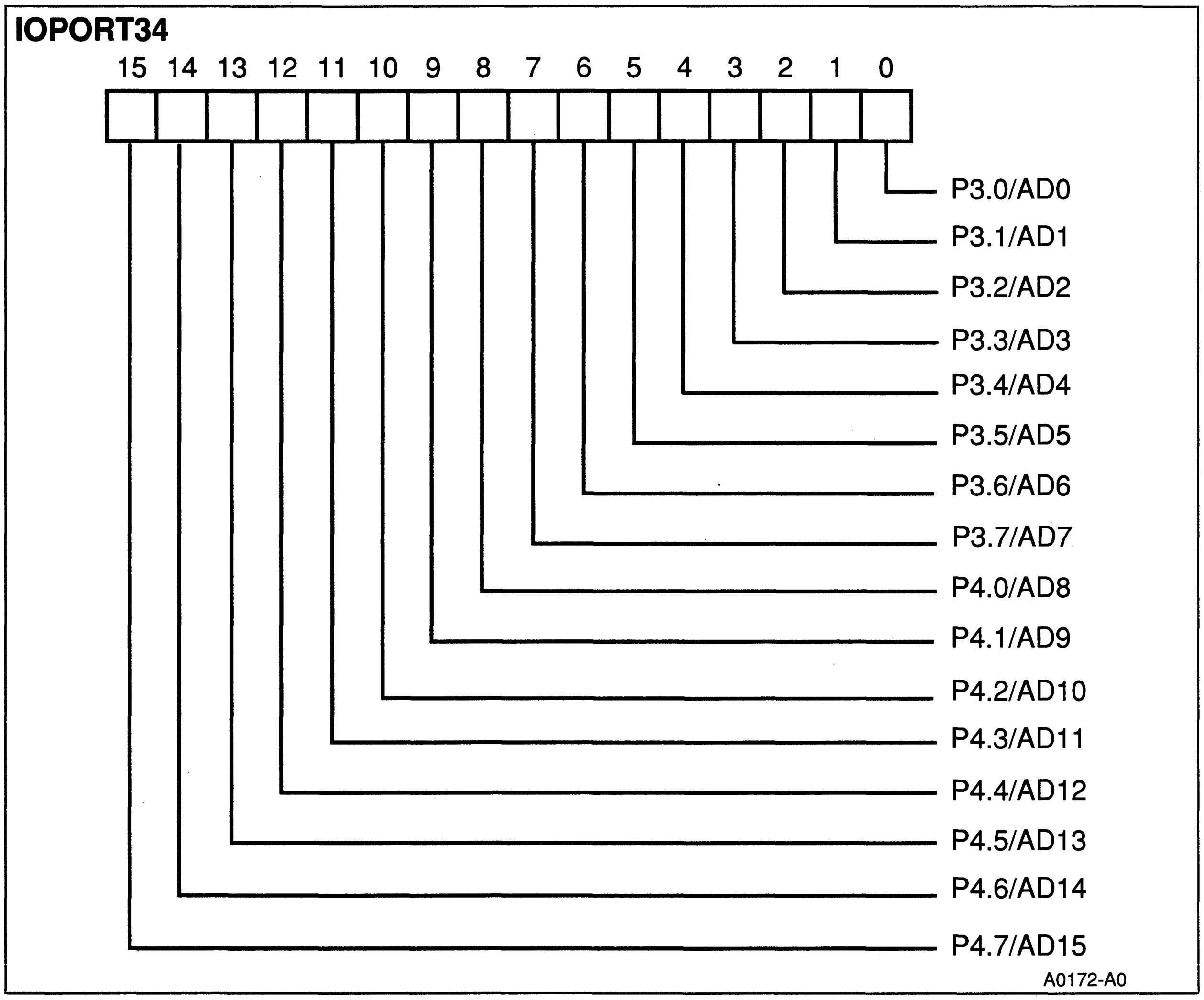
IOPORT2

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	P2.0/TXD	P2.0/TXD	1	Output Pin P2.0 Setting IOC1.5 enables the pin as TXD, which serves as the transmit pin for serial port modes 1, 2, and 3 and the shift clock for mode 0.
1	P2.1/RXD	P2.1/RXD	0	Input Pin P2.1 Setting SP_CON.3 enables the pin as RXD. In modes 1, 2, and 3, RXD is used to receive serial port data. In mode 0, it functions as an input or an open-drain output for data
2	P2.2/EXTINT	P2.2/EXTINT	0	Input Pin 2.2. Clearing IOC1.1 selects P2.2 as the source for the EXTINT interrupt (INT07, 200EH). P2.2 can generate either EXTINT (INT07) or EXTINT1 (INT13, 203AH).
3	P2.3/T2CLK	P2.3/T2CLK	0	Input Pin P2.3 Clearing IOC0.7 enables the pin as the external clock input for Timer 2.
4	P2.4/T2RST	P2.4/T2RST	0	Input Pin P2.4 Clearing IOC0.5 enables the pin as the external reset to Timer 2. IOC0.3 must be set to enable the external reset.
5	P2.5/PWM0	P2.5/PWM0	0	Output Pin P2.5 Setting IOC1.0 enables this pin as the PWM0 output.
6	P2.6/T2UP-DN	P2.6/ T2UP-DN	1	Quasi-bidirectional pin P2.6. Setting IOC2.1 enables the pin as the direction control for Timer 2. Timer 2 counts up when the pin is low and counts down when the pin is high.
7	P2.7/T2CAP	P2.7/T2CAP	1	Quasi-bidirectional pin P2.7 A rising edge on P2.7 captures the value of Timer 2 in the T2CAPTURE register and generates a Timer 2 Capture interrupt (INT11, 2036H).

Input/Output Ports 3 and 4 Register

IOPORT34
1FFEH

Ports 3 and 4 contain bidirectional port pins with open-drain outputs. The pins are shared with the multiplexed address/data bus. The pins automatically switch to their system bus functions when EA# is low and to their open-drain port functions when EA# is high. Ports 3 and 4 can be read and written only as a word, at location 1FFEH.

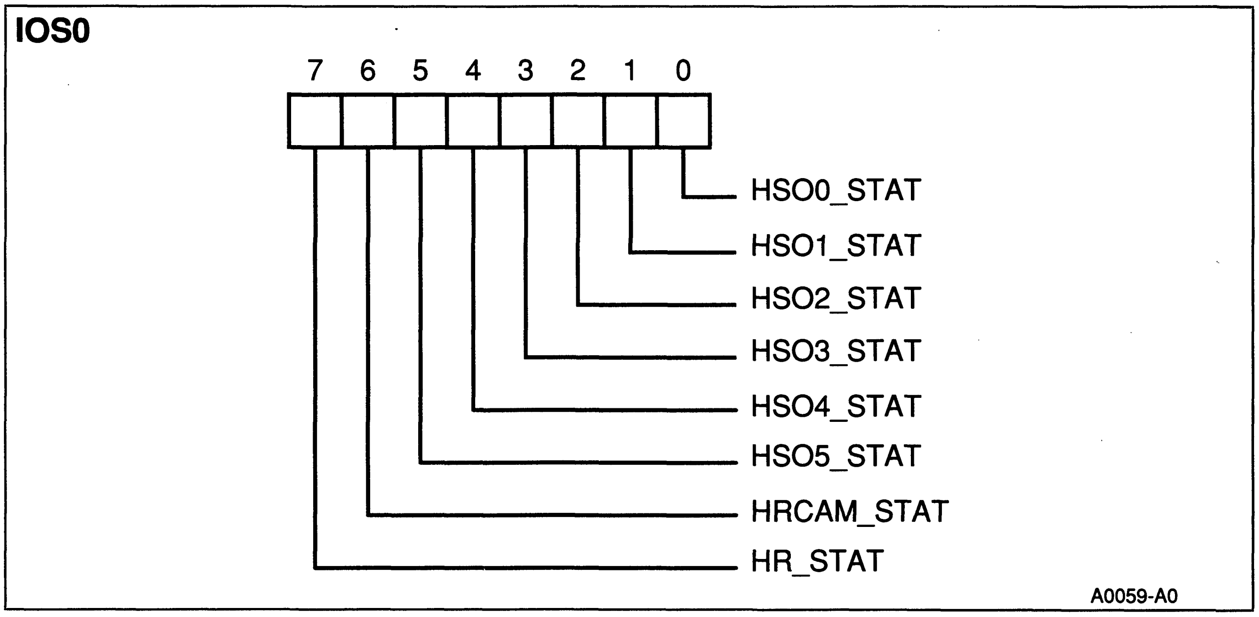


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	P3.0–P3.7 /AD0–AD7	P3.0–P3.7 /AD0–AD7	1111 1111	Bidirectional port pins P3.0–P3.7 during external accesses (EA# = 0); otherwise, AD0–AD7.
8–15	P4.0–P4.7 /AD8–AD15	P4.0–P4.7 /AD8–AD15	1111 1111	Bidirectional port pins P4.0–P4.7 during external accesses (EA# = 0); otherwise, AD8–AD15.

Input/Output Status Register 0

IOS0
15H
HWindow 0 (Read), HWindow 15 (Write)

The IOS0 register indicates the current state of the HSO pins. Writing to the corresponding bits (IOS0.0–IOS0.5) in HWindow 15 can set or clear the HSO pins. IOS0.6 and IOS0.7 indicate the current state of the HSO CAM file and the HSO holding register. (IOS0.6 and IOS0.7 cannot be written.)



IOS0

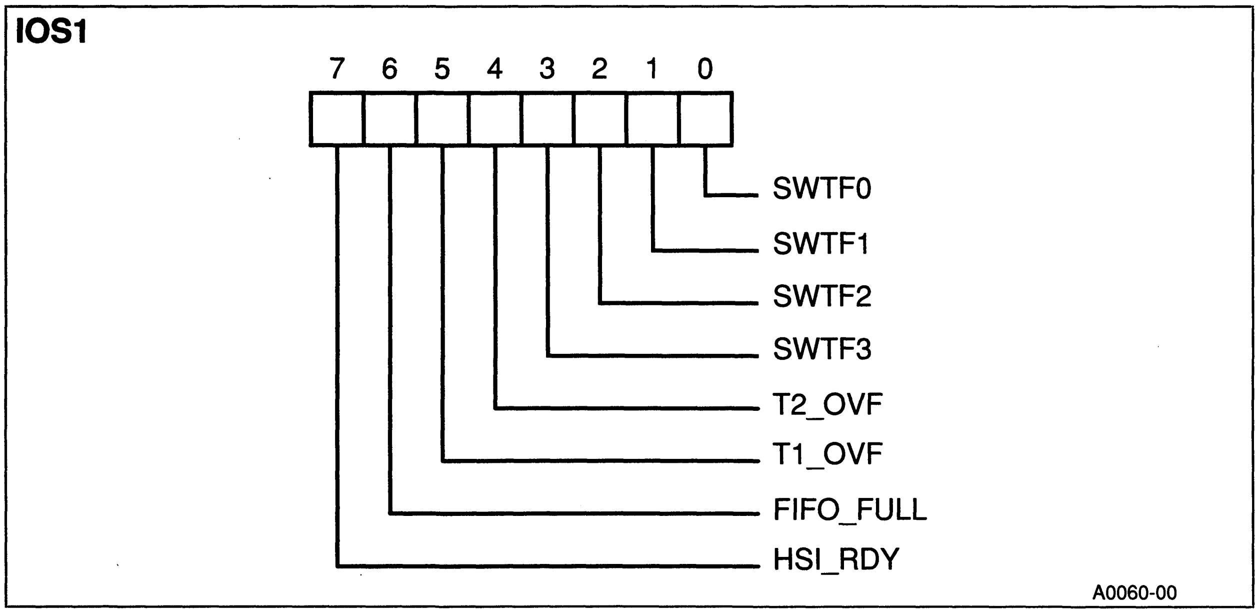
Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	HSO0_STAT	HSO.0 State	0	Current state of the HSO.0 pin.
1	HSO1_STAT	HSO.1 State	0	Current state of the HSO.1 pin.
2	HSO2_STAT	HSO.2 State	0	Current state of the HSO.2 pin.
3	HSO3_STAT	HSO.3 State	0	Current state of the HSO.3 pin.
4	HSO4_STAT	HSO.4 State	0	Current state of the HSO.4 pin. HSO.4 is a bidirectional port pin and is multiplexed with the HSI.2 pin. Setting IOC1.4 enables the output function of HSO.4.
5	HSO5_STAT	HSO.5 State	0	Current state of the HSO.5 pin. HSO.5 is a bidirectional port pin and is multiplexed with the HSI.3 pin. Setting IOC1.6 enables the output function of HSO.5.
6	HRCAM_STAT	HSO CAM and Holding Register State	0	Current state of the HSO holding register and CAM. 0= HSO holding register is empty and at least one CAM slot is empty 1= HSO holding register is full To avoid overwriting a current value, do not write to the holding register until either this bit or IOS0.7 is cleared.
7	HR_STAT	HSO Holding Register State	0	Current state of the HSO holding register. 0= HSO holding register is empty 1= HSO holding register is full To avoid overwriting a current value, do not write to the holding register until either this bit or IOS0.6 is cleared.

Input/Output Status Register 1

IOS1
16H
HWindow 0 (Read), HWindow 15 (Write)

The IOS1 register contains flags that indicate which events triggered interrupts. IOS1.0 through IOS1.5 indicate the status of software timers, Timer 2, and Timer 1. Reading IOS1 clears bits IOS1.0–IOS1.5. Writing to IOS1.0–IOS1.5 sets or clears the bits, but does not trigger interrupts.

IOS1.6 and IOC1.7 indicate the status of the HSI FIFO register and the HSI holding register. IOS1.6 and IOS1.7 cannot be written.



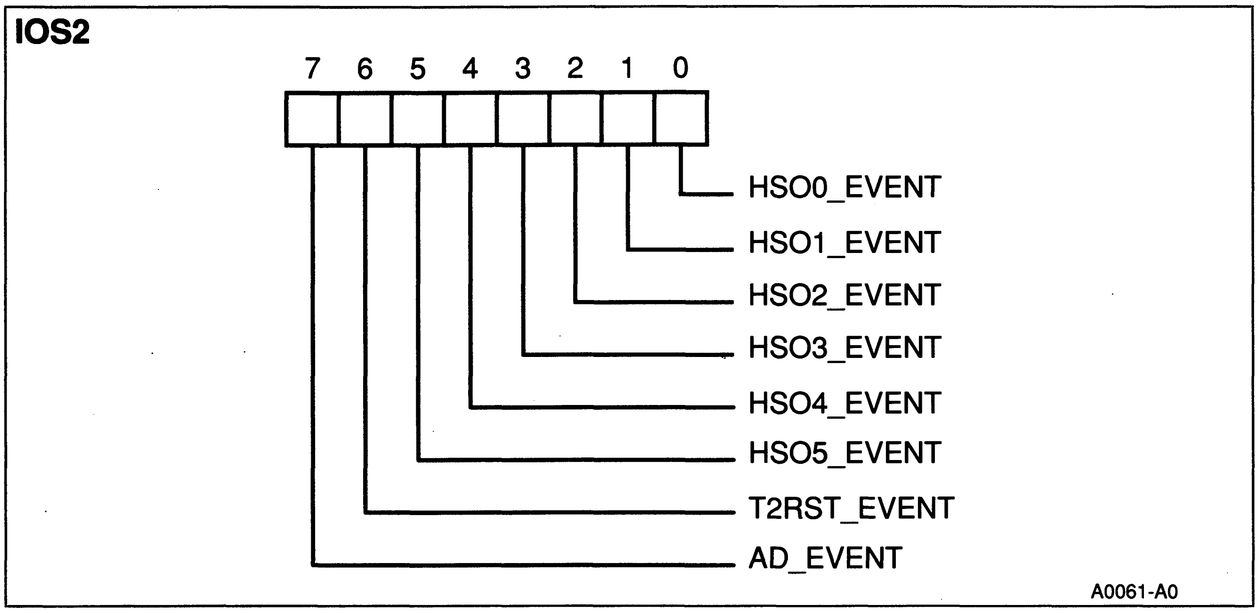
IOS1

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	SWTF0	Software Timer 0 Flag	0	This bit, when set, indicates that Software Timer 0 has expired, triggering INT05 (200AH). INT_MASK.5 must be set to enable the interrupt.
1	SWTF1	Software Timer 1 Flag	0	This bit, when set, indicates that Software Timer 1 has expired, triggering INT05 (200AH). INT_MASK.5 must be set to enable the interrupt.
2	SWTF2	Software Timer 2 Flag	0	This bit, when set, indicates that Software Timer 2 has expired, triggering INT05 (200AH). INT_MASK.5 must be set to enable the interrupt.
3	SWTF3	Software Timer 3 Flag	0	This bit, when set, indicates that Software Timer3 has expired, triggering INT05 (200AH). INT_MASK.5 must be set to enable the interrupt.
4	T2_OVF	Timer 2 Overflow Flag	0	Both Timer 1 and Timer 2 can generate a Timer Overflow interrupt (INT00, 2000H). This bit, when set, indicates that Timer 2 triggered the interrupt. INT_MASK.0 must be set to enable the interrupt.
5	T1_OVF	Timer 1 Overflow Flag	0	Both Timer 1 and Timer 2 can generate a Timer Overflow interrupt (INT00, 2000H). This bit, when set, indicates that Timer 1 triggered the interrupt. INT_MASK.0 must be set to enable the interrupt.
6	FIFO_FULL	Sixth FIFO Entry Flag	0	This bit, when set, indicates that the HSI FIFO has six or more entries, independent of the holding register. This event can generate either an HSI Data Available interrupt (INT02, 2004H) or an HSI FIFO Full interrupt (INT14, 203CH), but should not be configured for both.
7	HSI_RDY	HSI Holding Register Data Ready	0	This bit, when set, indicates that the HSI holding register has been loaded. When IOC1.7 is clear, this event generates an HSI Data Available interrupt (INT02, 2004H). INT_MASK.2 must be set to enable the interrupt.

Input/Output Status Register 2

IOS2
17H
HWindow 0 (Read), HWindow 15 (Write)

The IOS2 register contains flags that indicate which HSO events have occurred. Writing to IOS2 sets or clears the status bits, but does not trigger interrupts. Reading IOS2 clears all bits.

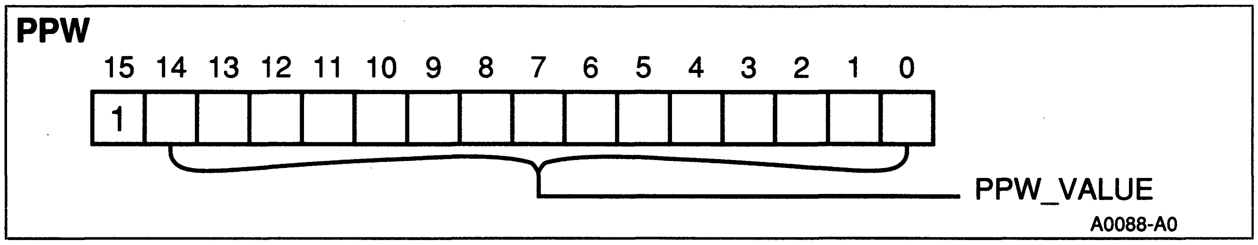


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	HSO0_EVENT	HSO.0 Pin Event	0	This bit, when set, indicates that an HSO command toggled the HSO.0 pin.
1	HSO1_EVENT	HSO.1 Pin Event	0	This bit, when set, indicates that an HSO command toggled the HSO.1 pin.
2	HSO2_EVENT	HSO.2 Pin Event	0	This bit, when set, indicates that an HSO command toggled the HSO.2 pin.
3	HSO3_EVENT	HSO.3 Pin Event	0	This bit, when set, indicates that an HSO command toggled the HSO.3 pin.
4	HSO4_EVENT	HSO.4 Pin Event	0	This bit, when set, indicates that an HSO command toggled the HSO.4 pin.
5	HSO5_EVENT	HSO.5 Pin Event	0	This bit, when set, indicates that an HSO command toggled the HSO.5 pin.
6	T2RST_EVENT	Timer 2 Reset Event	0	This bit, when set, indicates that an HSO command reset Timer 2.
7	AD_EVENT	A/D Conversion Start Event	0	This bit, when set, indicates that an HSO command started an A/D conversion.

Programming Pulse Width Register — PPW (no direct access)

The Programming Pulse Width register is accessible only during Auto Programming mode. It is loaded from external address 4015/4014H.

The value in the PPW register determines the programming pulse width. The programming pulse width must be at least 100μs for programming to function correctly. The most-significant bit (PPW.15) must always be set. The high byte of PPW usually contains 80H and the low byte usually contains 05H, although they can hold other values.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–14	PPW_VALUE	Programming Pulse Width	0000 0101 0000 000	These bits usually equal 05H, the correct value for XTAL1 = 8 MHz. Write the desired PPW_VALUE to these bits.
15	PPW_MSB	PPW Most-Significant Bit	1	This bit must always equal "1."

Use the following formula and round the result to the next higher integer value to determine the proper PPW_VALUE.

$$\text{PPW\_VALUE} = (0.6944 \times F_{OSC}) - 1$$

where:

PPW_VALUE	is a 15-bit word
F_{OSC}	is the XTAL1 frequency, in MHz

For example, assume XTAL1 is 8 MHz:

$$\begin{aligned}\text{PPW\_VALUE} &= (0.6944 \times 8) - 1 \\ &= 5.5552 - 1 \\ &= 4.5552 \\ &\equiv 5\end{aligned}$$

Program Status Word

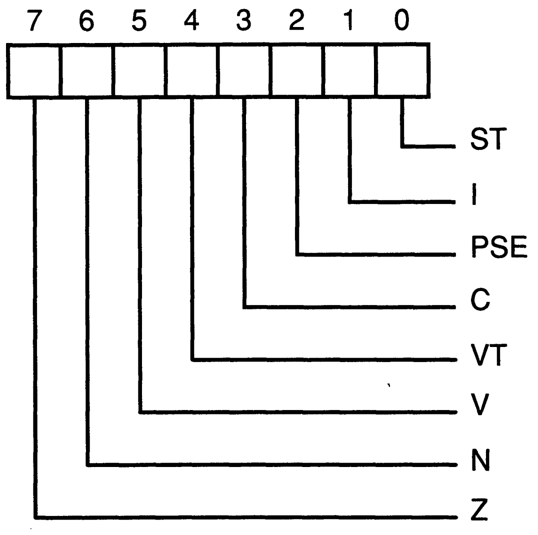
PSW
(no direct access)

The PSW actually consists of two bytes. The high byte is the status word, which is described here; the low byte is the INT_MASK register. The status word contains one bit (PSW.1) that globally enables or disables servicing of all maskable interrupts, one bit (PSW.2) that enables or disables the Peripheral Transaction Server (PTS), and six Boolean flags that reflect the state of a user's program.

The status word portion of the PSW cannot be accessed directly. To access the status word, push the value onto the stack (PUSHF), then pop the value to a register (POP *test_reg*). The PUSHF and PUSHA instructions save the PSW in the system stack; POPF and POPA restore it.

The EI and DI instructions set and clear PSW.1; the EPTS and DPTS instructions set and clear PSW.2. Various instructions test, set, and clear the Boolean flags. (Appendix B contains one table that shows the effect of instructions on the PSW flags and one that shows the effect of PSW flags on conditional jump instructions.)

PSW



A0038-A0

PSW

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	ST	Sticky Bit Flag		This flag is set to indicate that, during a right shift, a "1" has been shifted into the Carry flag and then shifted out. This bit is undefined after a multiply operation. The Sticky Bit flag can be used with the Carry flag to allow finer resolution in rounding decisions. (See the description of the Carry flag for details.)
1	I	Interrupt Disable (Global)		This bit globally enables or disables the servicing of all maskable interrupts (that is, all interrupts except NMI, TRAP, and Unimplemented Opcode). The bits in INT_MASK and INT_MASK1 individually enable or disable the interrupts. The EI instruction sets this bit; DI clears it. 1= enable interrupt servicing 0= disable interrupt servicing
2	PSE	PTS Enable		This bit globally enables or disables the Peripheral Transaction Server (PTS). The EPTS instruction sets this bit; DPTS clears it. 1= enable PTS 0= disable PTS
3	C	Carry Flag		This flag is set to indicate the state of an arithmetic carry from the most-significant bit of the ALU or the state of the last bit shifted out of an operand. If a subtraction operation generates a borrow, the Carry flag is cleared. **C** **Value of Bits Shifted Off** 0 <½LSB 1 ≥½LSB Normally, the result is rounded up if the Carry flag is set. The Sticky Bit flag allows a finer resolution in the rounding decision. **C** **ST** **Value of Bits Shifted Off** 0 0 =0 0 1 >0 and <½ LSB 1 0 =½LSB 1 1 >½ LSB and <1 LSB

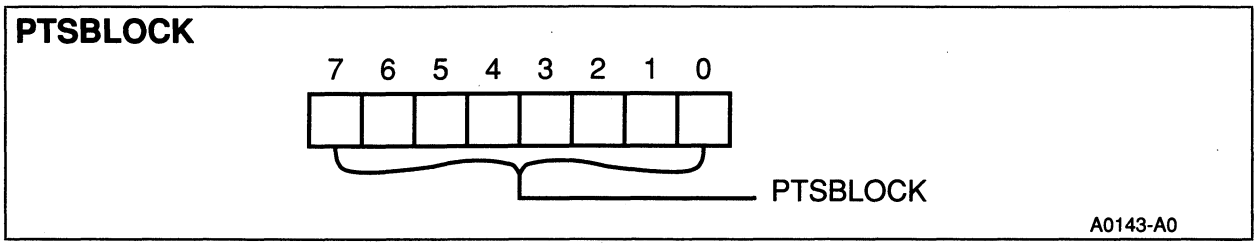
PSW

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
4	VT	Overflow-Trap Flag		This flag is set when the Overflow flag is set, but it is cleared only by the CLRVT, JVT, and JNVT instructions. This allows testing for a possible overflow condition at the end of a sequence of related arithmetic operations, which is generally more efficient than testing the Overflow flag after each operation.
5	V	Overflow Flag		This flag is set to indicate that an operation generated a result outside the range for the destination data type. For shift operations (SHL, SHLB, and SHLL), the flag is set if the most-significant bit of the operand changes during the shift. For divide operations, the flag is set under the following conditions: **Instruction** **Result** DIVU > 255 (0FFH) DIVUB > 65535 (0FFFFH) DIV < –127 (81H) or > +127 (7FH) DIVB < –32767 (8001H) or > +32767 (7FFFH)
6	N	Negative Flag		This flag is set to indicate that an operation generated a negative result. The flag is correct even if an overflow occurs. For all shift operations and the NORML instruction, the flag is set to equal the most-significant bit of the result, even if the shift count is zero.
7	Z	Zero Flag		This flag is set to indicate that the result of an operation was zero. For add-with-carry and subtract-with-borrow operations, the flag is never set, but it is cleared if the result is non-zero. This way, the Zero flag indicates the correct zero or non-zero result for multiple-precision calculations.

PTS Block Register

PTSBLOCK
Offset 7 from PTS Control Block

The PTSBLOCK register is used during block transfer, HSO, and HSI modes. PTSBLOCK controls the number of transfers that will take place.



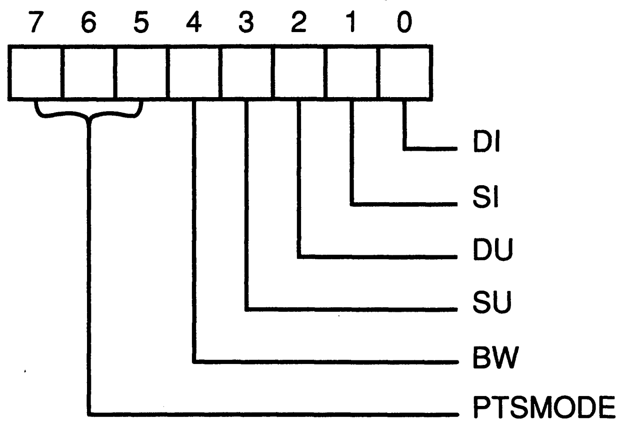
Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	PTSBLOCK	PTS Block Count	XXXX XXXX	Controls the number of transfers that will take place. Values depend on the PTS mode: **Mode** **Values** Block transfer 1–32 HSI 1–7 HSO 1–8 In all modes, writing zero to this register causes the maximum number of transfers to take place (32 for block transfer mode, 7 for HSI mode, and 8 for HSO mode).

PTS Control Register

PTSCON
Offset 1 from PTS Control Block

Three bits of the PTSCON register determine the PTS mode: single transfer, block transfer, A/D, HSO, or HSI. The PTS mode defines the functions of the remaining five bits. PTSCON has one configuration for the single and block transfer modes and one for the AD, HSO, and HSI modes. The configurations are described separately here. (PSW.2, controlled by the DPTS and EPTS instructions, globally enables or disables the PTS.)

PTSCON (Single and Block Transfer Modes)

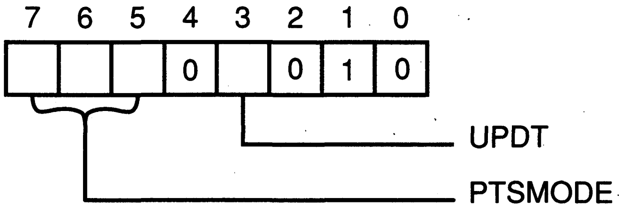


A0036-A0

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	DI	PTSDST Auto-Increment	0	Setting this bit causes the PTS destination register to increment at the end of each PTS cycle.
1	SI	PTSSRC Auto-Increment	0	Setting this bit causes the PTS source register to increment at the end of each PTS cycle.
2	DU	Update PTSDST	0	Setting this bit causes the PTSDST register to retain its final value at the end of a PTS cycle. Clearing it causes the register revert to the value that existed at the beginning of the PTS cycle.
3	SU	Update PTSSRC	0	Setting this bit causes the PTSSRC register to retain its final value at the end of a PTS cycle. Clearing it causes the register to revert to the value that existed at the beginning of the PTS cycle.
4	BW	Byte/Word Transfer	0	Setting this bit specifies a byte transfer. Clearing it specifies a word transfer.
5–7	PTSMODE	PTS Mode	000	These bits specify the PTS mode: **Bit 7 6 5** 0 0 0 Single Transfer 1 0 0 Block Transfer

PTSCON

PTSCON (A/D, HSO, and HSI Modes)



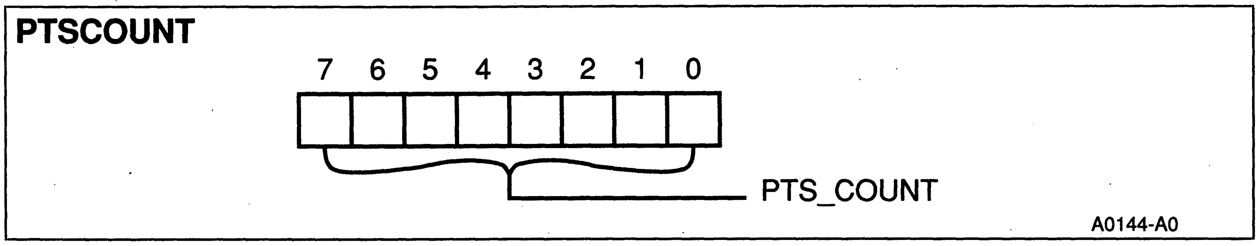
A0037-A0

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	—	—	0	Always zero.
1	—	—	1	Always one.
2	—	—	0	Always zero.
3	UPDT	Update Register	0	Setting this bit causes the associated register to be loaded with the value that exists at the end of a PTS cycle. Clearing it causes the register to be loaded with the value that existed at the beginning of the PTS cycle. **Mode Register** A/D PTS_S/D HSI PTSDST HSO PTSSRC
4	—	—	0	Always zero.
5–7	PTSMODE	PTS Mode	000	These bits specify the PTS mode: **Bit 7 6 5** 1 1 0 A/D 0 1 1 HSO 0 0 1 HSI

PTS Count Register

PTSCOUNT
Offset 0 from PTS Control Block

The PTSCOUNT register is used in all PTS modes. PTSCOUNT defines the number of PTS cycles to be executed consecutively without CPU intervention. Since PTSCOUNT is an 8-bit value, the maximum number of cycles is 256. PTSCOUNT is decremented at the end of each PTS cycle. When PTSCOUNT reaches zero, hardware clears the corresponding PTSSEL bit and sets the PTSSRV bit, which requests the end-of-PTS interrupt. When the end-of-PTS interrupt is called, hardware clears the PTSSRV bit. The PTSSEL bit must be set manually to re-enable the PTS channel.

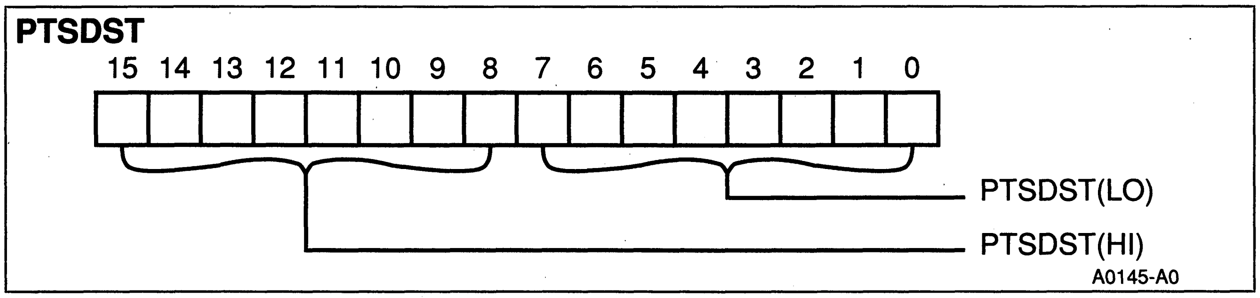


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	PTS_COUNT	Consecutive PTS Cycles	XXXX XXXX	Defines the number of PTS cycles to be executed consecutively without CPU intervention. Maximum value is 256.

PTS Destination Register

PTSDST
Offset 4 from PTS Control Block

The PTSDST register is used in single transfer, block transfer, and HSI modes. PTSDST points to the destination memory location. PTSDST is optionally incremented at the end of a PTS cycle. In single transfer mode, PTSCON.0 and PTSCON.2 control whether PTSDST is incremented. In block transfer mode, PTSCON.0 controls whether PTSDST is incremented after each transfer and PTSCON.2 controls whether PTSDST retains its final value or reverts to its original value. In HSI mode, PTSCON.3 determines whether PTSDST is updated at the end of the PTS cycle.

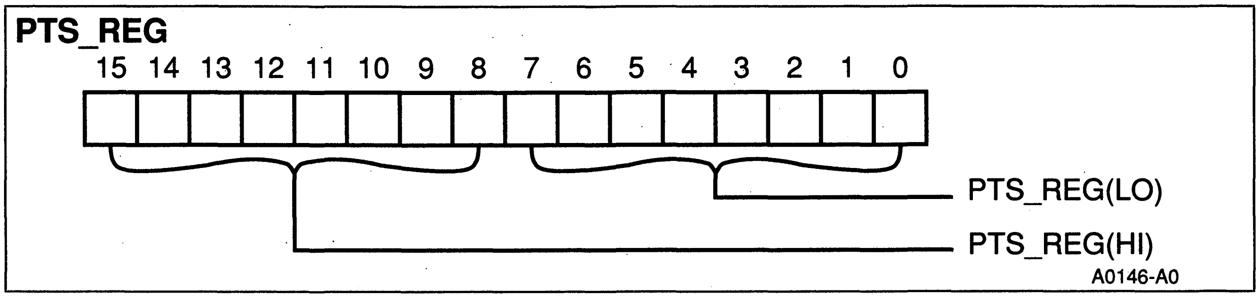


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	PTSDST (LO)	PTS Destination Address, Low Byte	XXXX XXXX	Low byte of the PTS destination address.
8–15	PTSDST (HI)	PTS Destination Address, High Byte	XXXX XXXX	High byte of the PTS destination address.

PTS AD Register

PTS_REG
Offset 3 from PTS Control Block

The PTS_REG register is used in A/D mode. PTS_REG points to address 02H. When read, this location contains the AD_RESULT register; when written, it contains the AD_COMMAND register.

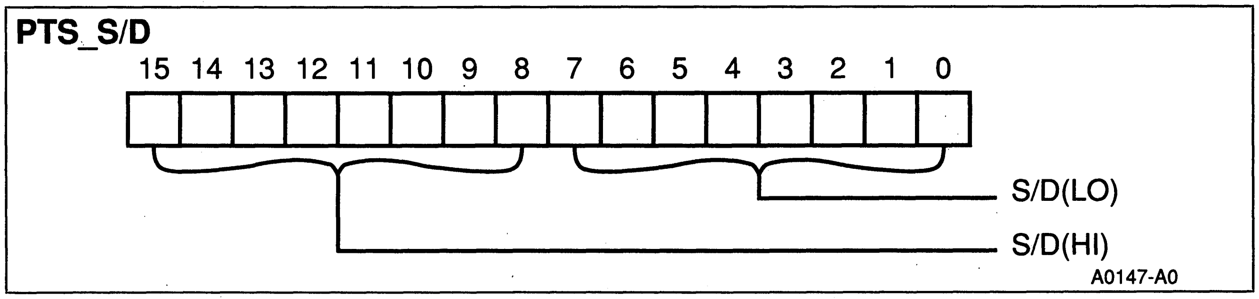


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	PTS_REG (LO)	Register Address, Low Byte	XXXX XXXX	Low byte of the AD_RESULT register address.
8–15	PTS_REG (HI)	Register Address, High Byte	XXXX XXXX	High byte of the AD_RESULT register address.

PTS Source/Destination Register

PTS_S/D
Offset 3 from PTS Control Block

The PTS_S/D register is used in A/D mode. PTS_S/D points to the memory location that contains the A/D command/data table. In a PTS A/D cycle, the word that PTS_S/D points to is loaded into a temporary internal register, PTS_S/D is incremented by 2, then the AD_RESULT register is stored at the updated PTS_S/D address. PTSCON.3 controls whether PTS_S/D is updated at the end of a PTS cycle, to point to the next word in the A/D table.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	S/D (LO)	Source/ Destination Address, Low Byte	XXXX XXXX	Low byte of the source/destination address.
8–15	S/D (HI)	Source/ Destination Address, High Byte	XXXX XXXX	High byte of the source/destination address.

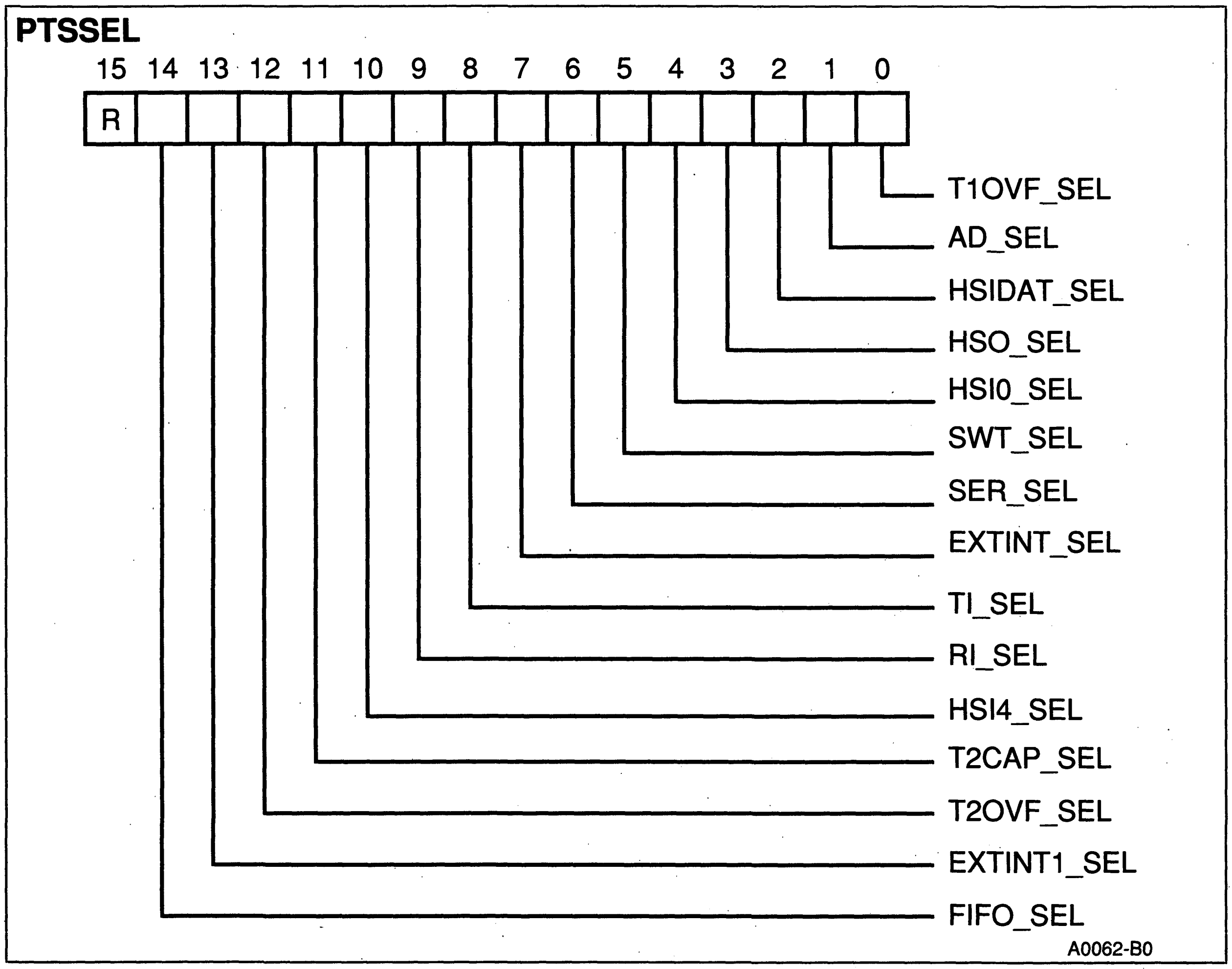
PTS Select Register (HI/LO)

PTSSEL
05/04H
HWindow 1 (Read/Write)

The PTSSEL register consists of two bytes. PTSSEL selects either a PTS cycle or a normal interrupt service routine for each of fifteen interrupt requests. Setting a bit selects a PTS cycle; clearing a bit selects a normal interrupt service routine.

When PTSCOUNT reaches zero, hardware clears the corresponding PTSSEL bit and sets the PTSSRV bit, which requests the end-of-PTS interrupt. When the end-of-PTS interrupt is called, hardware clears the PTSSRV bit. The PTSSEL bit must be set manually to re-enable the PTS channel.

Each interrupt can be masked by the corresponding bit in the INT_MASK or INT_MASK1 register. The PTS is globally enabled or disabled by PSW.2.



PTSSEL

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	T1OVF_SEL	Timer Overflow Interrupt Select	0	Setting this bit causes a Timer Overflow interrupt (INT00) to be handled by a PTS cycle.
1	AD_SEL	A/D Conversion Complete Interrupt Select	0	Setting this bit causes an A/D Conversion Complete interrupt (INT01) to be handled by a PTS cycle.
2	HSIDAT_SEL	HSI Data Available/ FIFO Full Interrupt Select	0	Setting this bit causes an HSI Data Available interrupt (INT02) to be handled by a PTS cycle.
3	HSO_SEL	HSO Output Event Interrupt Select	0	Setting this bit causes a High-Speed Output interrupt (INT03) to be handled by a PTS cycle.
4	HSI0_SEL	HSI.0 External Interrupt Select	0	Setting this bit causes an HSI.0 Pin interrupt (INT04) to be handled by a PTS cycle.
5	SWT_SEL	Software Timer Interrupt Select	0	Setting this bit causes a Software Timer interrupt (INT05) to be handled by a PTS cycle.
6	SER_SEL	Serial Port Interrupt Select	0	Setting this bit causes a Serial Port interrupt (INT06) to be handled by a PTS cycle.
7	EXTINT_SEL	EXTINT Pin or P0.7 Interrupt Select	0	Setting this bit causes an EXTINT pin or P0.7 interrupt (INT07) to be handled by a PTS cycle.

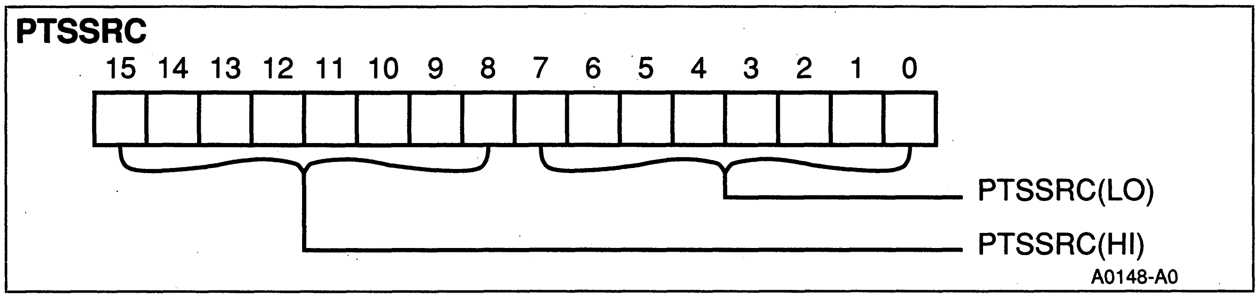
PTSSEL

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
8	TI_SEL	Transmit Interrupt Select	0	Setting this bit causes a TI interrupt (INT08) to be handled by a PTS cycle.
9	RI_SEL	Receive Interrupt Select	0	Setting this bit causes an RI interrupt (INT09) to be handled by a PTS cycle.
10	HSI4_SEL	HSI FIFO 4 Interrupt Select	0	Setting this bit causes an HSI FIFO 4 interrupt (INT10) to be handled by a PTS cycle.
11	T2CAP_SEL	Timer 2 Capture Interrupt Select	0	Setting this bit causes a Timer 2 Capture interrupt (INT11) to be handled by a PTS cycle.
12	T2OVF_SEL	Timer 2 Overflow Interrupt Select	0	Setting this bit causes a Timer 2 Overflow interrupt (INT12) to be handled by a PTS cycle.
13	EXTINT1_SEL	EXTINT Pin Interrupt Select	0	Setting this bit causes an EXTINT1 interrupt (INT13) to be handled by a PTS cycle.
14	FIFO_SEL	HSI FIFO Full Interrupt Select	0	Setting this bit causes an HSI FIFO Full interrupt (INT14) to be handled by a PTS cycle.
15	—	—	0	Reserved; always write as zero.

PTS Source Register

PTSSRC
Offset 2 from PTS Control Block

The PTSSRC register is used in single transfer, block transfer, and HSO modes. PTSSRC points to the source memory location. PTSSRC is optionally incremented at the end of a PTS cycle. In single transfer mode, PTSCON.1 and PTSCON.3 control whether PTSDST is incremented. In block transfer mode, PTSCON.1 controls whether PTSDST is incremented after each transfer and PTSCON.3 controls whether PTSDST retains its final value or reverts to its original value. In HSO mode, PTSCON.3 determines whether PTSSRC is updated at the end of the PTS cycle.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	PTSSRC (LO)	PTS Source Address, Low Byte	XXXX XXXX	Low byte of the PTS source address.
8–15	PTSSRC (HI)	PTS Source Address, High Byte	XXXX XXXX	High byte of the PTS source address.

PTS Service Register

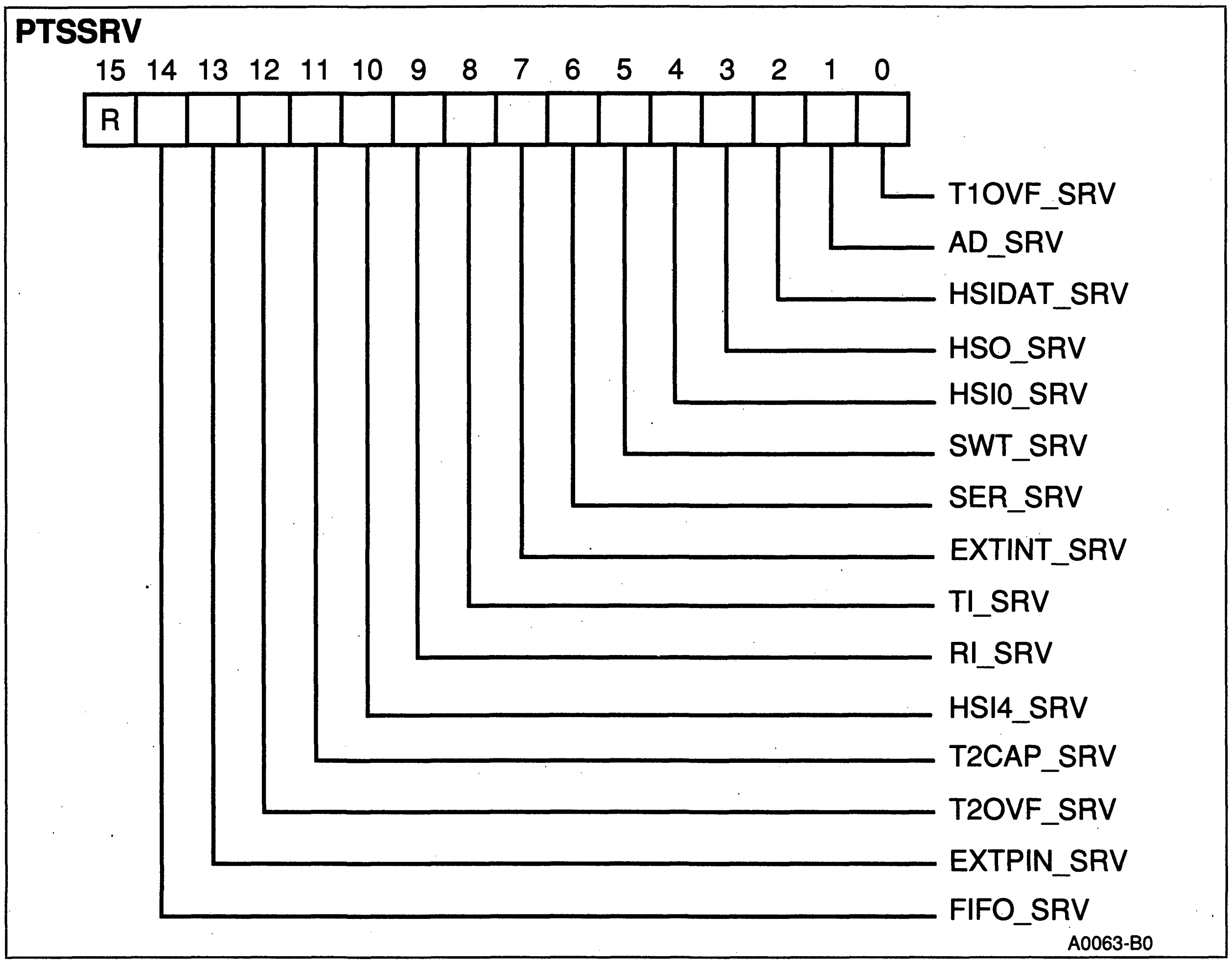
PTSSRV
07/06H
HWindow 1 (Read/Write)

The PTSSRV register consists of two bytes. PTSSRV is used by the hardware to indicate that the final PTS cycle has been serviced. When PTSCOUNT reaches zero, hardware clears the corresponding PTSSEL bit and sets the PTSSRV bit, which requests the end-of-PTS interrupt. When the end-of-PTS interrupt is called, hardware clears the PTSSRV bit. The PTSSEL bit must be set manually to re-enable the PTS channel.

The PTS is globally enabled or disabled by PSW.2, which is set with the EPTS instruction and cleared with the DPTS instruction. Interrupts are enabled or disabled (masked) by the corresponding bits in the INT_MASK and INT_MASK1 registers. Individual interrupts are enabled as PTS cycles by the corresponding bits in the PTSSEL register.

The end-of-PTS interrupt vectors through the same location that the corresponding normal interrupt vector would. For example, if PTSSEL.8 is set, the TI interrupt is handled by its PTS vector at 2050H with its end-of-PTS vector at 2030H. (Refer to Table C-4 for interrupt vector locations.)

PTSSRV


PTSSRV
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
T1OVF_SRV
AD_SRV
HSIDAT_SRV
HSO_SRV
HSI0_SRV
SWT_SRV
SER_SRV
EXTINT_SRV
TI_SRV
RI_SRV
HSI4_SRV
T2CAP_SRV
T2OVF_SRV
EXTPIN_SRV
FIFO_SRV
A0063-B0

PTSSRV

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0	T1OVF_SRV	Timer 1 Overflow Interrupt Serve	0	This bit is set when PTSCOUNT for the Timer 1 Overflow PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2000H.
1	AD_SRV	A/D Conversion Complete Interrupt Serve	0	This bit is set when PTSCOUNT for the A/D Complete PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2002H.
2	HSIDAT_SRV	HSI Data Available/ FIFO Full Interrupt Serve	0	This bit is set when PTSCOUNT for the HSI PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2004H.
3	HSO_SRV	HSO Output Event Interrupt Serve	0	This bit is set when PTSCOUNT for the HSO PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2006H.
4	HSI0_SRV	HSI.0 External Interrupt Serve	0	This bit is set when PTSCOUNT for the HSI.0 PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2008H.
5	SWT_SRV	Software Timer Interrupt Serve	0	This bit is set when PTSCOUNT for the Software Timer PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 200AH.
6	SER_SRV	Serial Port Interrupt Serve	0	This bit is set when PTSCOUNT for the Serial Port PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 200CH.
7	EXTINT_SRV	EXTINT Pin or P0.7 Interrupt Serve	0	This bit is set when PTSCOUNT for the EXTINT PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 200EH.

PTSSRV

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
8	TI_SRV	Transmit Interrupt Serve	0	This bit is set when PTSCOUNT for the TI PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2030H.
9	RI_SRV	Receive Interrupt Serve	0	This bit is set when PTSCOUNT for the RI PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2032H.
10	HSI4_SRV	HSI FIFO 4 Interrupt Serve	0	This bit is set when PTSCOUNT for the HSI4 PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2034H.
11	T2CAP_SRV	Timer 2 Capture Interrupt Serve	0	This bit is set when PTSCOUNT for the T2 Capture PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2036H.
12	T2OVF_SRV	Timer 2 Overflow Interrupt Serve	0	This bit is set when PTSCOUNT for the Timer 2 Overflow PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 2038H.
13	EXTPIN_SRV	EXTINT Pin Interrupt Serve	0	This bit is set when PTSCOUNT for the EXTPIN PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 203AH.
14	FIFO_SRV	HSI FIFO Full Interrupt Serve	0	This bit is set when PTSCOUNT for the FIFO PTS channel reaches zero, which initiates an end-of-PTS interrupt through location 203CH.
15	—	—	0	Reserved; always write as zero.

PWM0 Control Register

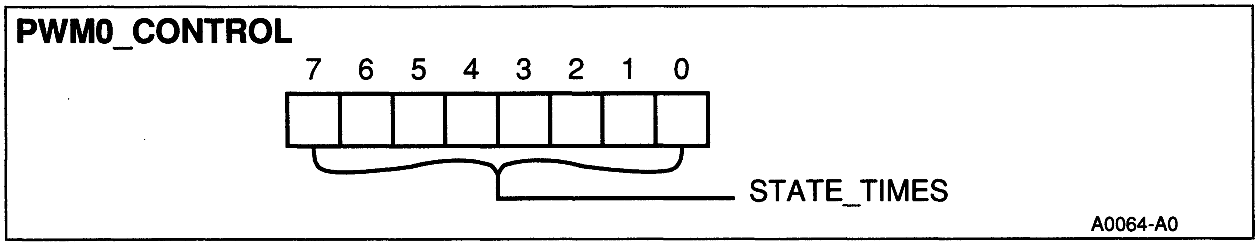
PWM0_CONTROL
17H
HWindow 0 (Write), HWindow 15 (Read)

PWM0 is multiplexed with P2.5. IOC1.0 must be set to enable the PWM0 output function.

The PWM's eight-bit counter is incremented every state time (with the PWM prescaler disabled) or every two state times (with the PWM prescaler enabled). When the counter is equal to zero, the PWM0 output is driven high. It remains high until the counter value matches the value in this register, at which time the output is pulled low. When the counter overflows, the output is again switched high. When PWM0_CONTROL equals zero, the output is always low.

The PWM0_CONTROL register, in conjunction with IOC2.2, determines how long the PWM0 output is held high during the pulse, effectively controlling the duty cycle. The value written to PWM0_CONTROL register can be from 0 to 255 state times (0% to 99.6% duty cycle).

Setting IOC2.2 enables the PWM's divide-by-two clock prescaler; clearing IOC2.2 disables it. When the prescaler is enabled, the total length of the pulse is 512 state times and the PWM0_CONTROL value is multiplied by 4. When it is disabled, the total pulse length is 256 state times and the PWM0_CONTROL value is multiplied by 2.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	STATE_TIMES	PWM0 High State Times	0000 0000	These bits determine the number of state times that the PWM0 output is held high during the pulse. Write a hexadecimal value (00H – FFH) to this register.

PWM0_CONTROL

The following formulas are used to determine the PWM period and the time that the output is held high:

	Clock Prescaler Disabled (IOC2.2=0)	Clock Prescaler Enabled (IOC2.2=1)
PWM Period (in µsec.) =	$\frac{512}{F_{OSC}}$	$\frac{1024}{F_{OSC}}$
PWM*x* High (in µsec.) =	$\frac{\text{PWMx\_CONTROL} \times 2}{F_{OSC}}$	$\frac{\text{PWMx\_CONTROL} \times 4}{F_{OSC}}$

where:

F_{OSC}	is the XTAL1 frequency, in MHz

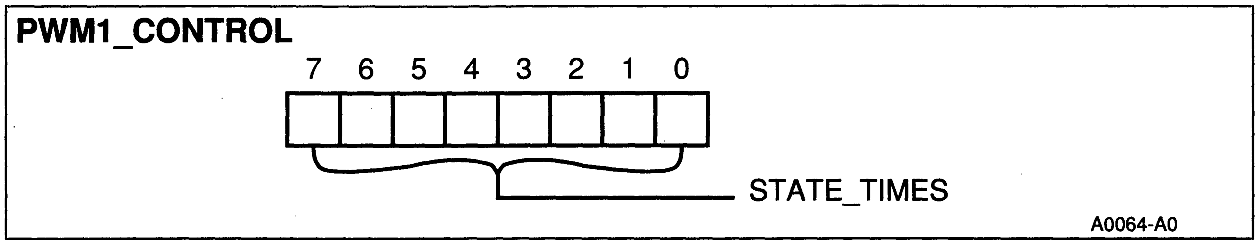
For example, if F_{OSC} equals 16 MHz, then the period of the PWM output waveform is 32 µs. If IOC2.2 is clear and PWM0_CONTROL equals 8AH (138 decimal), PWM0 is held high for 17.25 µs (and low for 14.8 µs) of the total 32 µs, resulting in a duty cycle of approximately 54%. When IOC2.2 is set, the same values would produce a period of 64 µs and PWM0 would be held high for 34.5 µs (and low for 29.5 µs), for the same duty cycle, approximately 54%.

PWM1 Control Register

PWM1_CONTROL
16H
HWindow 1 (Read/Write)

PWM1 is multiplexed with P1.3. IOC3.2 must be set to enable the PWM1 output function.

The PWM1_CONTROL register, in conjunction with IOC2.2, determines how long the PWM1 output is held high during the pulse, effectively controlling the duty cycle. The value written to PWM1_CONTROL register can be from 0 to 255 state times (0% to 99.6% duty cycle). Please refer to the PWM0_CONTROL register description for additional information.



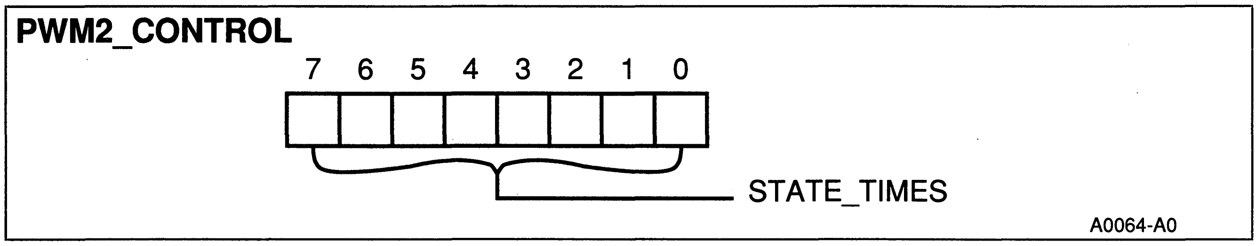
Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	STATE_TIMES	PWM1 High State Times	0000 0000	These bits determine the number of state times that the PWM1 output is held high during the pulse. Write a hexadecimal value (00H – FFH) to this register.

PWM2 Control Register

PWM2_CONTROL
17H
HWindow 1 (Read/Write)

PWM2 is multiplexed with P1.4. IOC3.3 must be set to enable the PWM2 output function.

The PWM2_CONTROL register, in conjunction with IOC2.2, determines how long the PWM2 output is held high during the pulse, effectively controlling the duty cycle. The value written to PWM2_CONTROL register can be from 0 to 255 state times (0% to 99.6% duty cycle). Please refer to the PWM0_CONTROL register description for additional information.

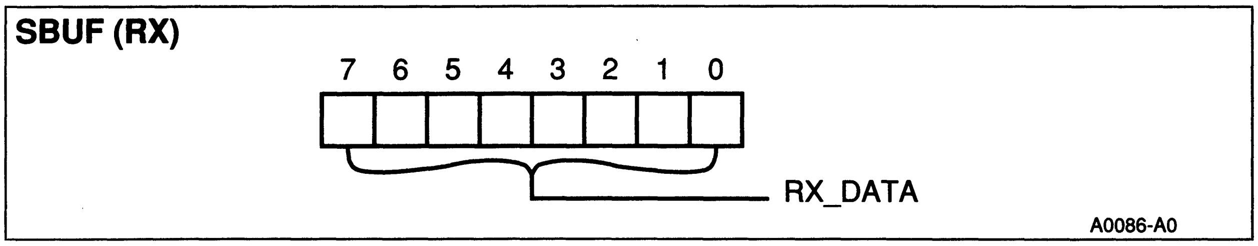


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	STATE_TIMES	PWM2 High State Times	0000 0000	These bits determine the number of state times that the PWM2 output is held high during the pulse. Write a hexadecimal value (00H – FFH) to this register.

Serial Port Receive Buffer Register

SBUF (RX)
07H
HWindow 0 (Read), HWindow 15 (Write)

The Serial Port Receive Buffer register contains data received from the serial port. The serial port receiver is double-buffered and can begin receiving a second data byte before the first byte is read. Data is held in the receive shift register until the last data bit is received, then the data byte is loaded into SBUF (RX). If data in the shift register is loaded into SBUF (RX) before the previous byte is read, the overflow error bit is set (SP_STAT.2). The data in SBUF (RX) will always be the last byte received, never a combination of the last two bytes.

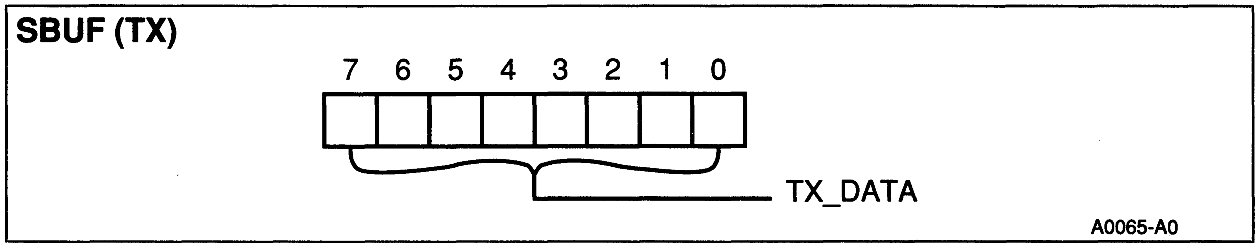


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	RX_DATA	Data Received	0000 0000	These bits contain the last byte of data received from the serial port.

Serial Port Transmit Buffer Register

SBUF (TX)
07H
HWindow 0 (Write), HWindow 15 (Read)

The Serial Port Transmit Buffer register contains data that is ready for transmission. In modes 1, 2, and 3, writing to SBUF (TX) starts a transmission. In mode 0, writing to SBUF (TX) starts a transmission only if the receiver is disabled (SP_CON.3=0).

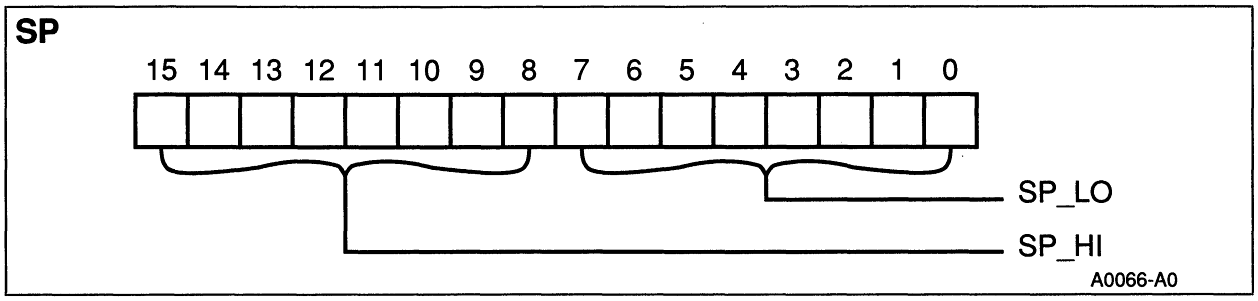


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	TX_DATA	Data to Transmit	0000 0000	These bits contain a byte of data to be transmitted by the serial port.

Stack Pointer

SP
19/18H
All HWindows (Read/Write)

The system's stack pointer may point anywhere in the 64K internal or external memory; it must be word aligned and must always be initialized before use. The stack pointer is decremented before a PUSH and incremented after a POP, so the stack pointer should be initialized to two bytes above the highest stack location. If stack operations are not being performed, locations 18H and 19H may be used as standard RAM.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	SP_LO	Stack Pointer (LO)	XXXX XXXX	The low byte of the system's stack pointer.
8–15	SP_HI	Stack Pointer (HI)	XXXX XXXX	The high byte of the system's stack pointer.

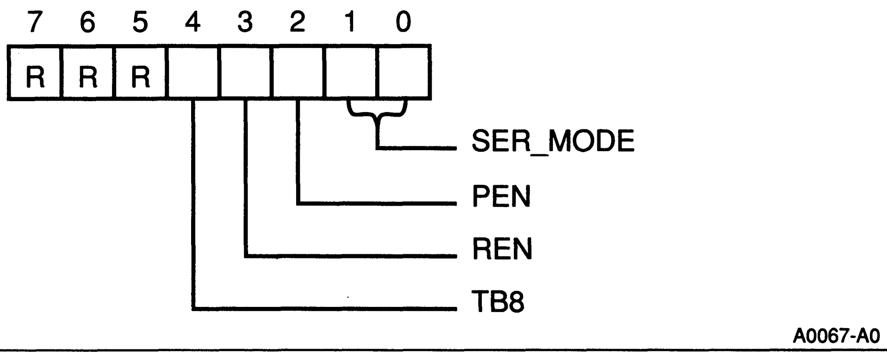
Serial Port Control Register

SP_CON
11H
HWindow 0 (Write), HWindow 15 (Read)

The Serial Port Control register selects the communications mode and enables or disables the receiver, even parity checking, and nine-bit data transmission.

TXD shares a pin with P2.0. IOC1.5 must be set to enable the TXD function. TXD serves as the transmit pin for serial port modes 1, 2, and 3 and the shift clock for mode 0. RXD shares a pin with P2.1. SP_CON.3 must be set to enable the RXD function. RXD receives serial port data in modes 1, 2, and 3 and functions as an input or an open-drain output for data in mode 0.

SP_CON



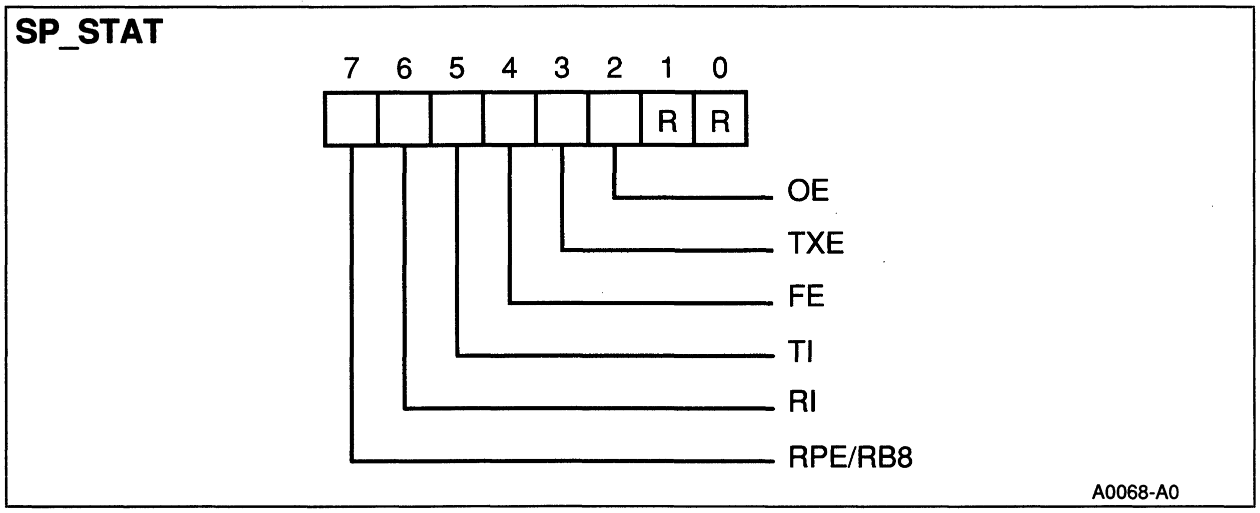
SP_CON

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–1	SER_MODE	Mode Selection	11	These two bits select the communications mode. **Bit 1 Bit 0** 0 0 Mode 0 0 1 Mode 1 1 0 Mode 2 1 1 Mode 3
2	PEN	Even Parity Enable	0	In modes 1 and 3, setting this bit enables the parity function. This bit must be cleared if mode 2 is used. When this bit is set, TB8 takes the even parity value on transmissions. With parity enabled, SP_STAT.7 becomes the Receive Parity Error bit.
3	REN	Receive Enable	1	Setting this bit enables the RXD function of the P2.1/RXD pin. When this bit is set, a high-to-low transition on the pin starts a reception in mode 1, 2, or 3. In mode 0, this bit must be clear for transmission to begin and must be set for reception to begin. Clearing this bit stops a reception in progress and inhibits further receptions.
4	TB8	Transmit Ninth Data Bit	0	This is the ninth data bit that will be transmitted in mode 2 or 3. This bit is cleared after each transmission, so it must be set before SBUF (TX) is written. When SPCON.2 is set, this bit takes on the even parity value.
5–7	—	—	000	Reserved; always write as zeros.

Serial Port Status Register

SP_STAT
11H
HWindow 0 (Read), HWindow 15 (Write)

The Serial Port Status register contains bits that indicate the status of the serial port.



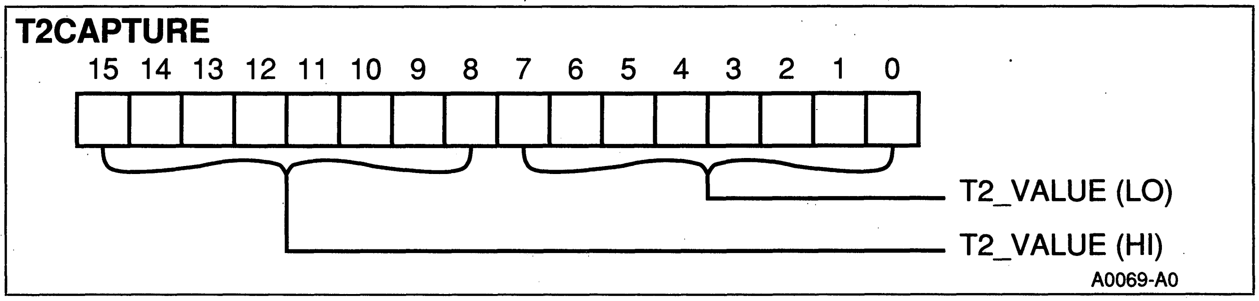
SP_STAT

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–1	—	—	11	Reserved; always write as zero.
2	OE	Overrun Error	0	This bit is set if data in the receive shift register is loaded into SBUF (RX) before the previous bit is read. Reading SP_STAT clears this bit.
3	TXE	SBUF (TX) Empty	0	This bit is set if the transmit buffer is empty and ready to accept up to two characters. It is cleared when a byte is written to SBUF (TX).
4	FE	Framing Error	1	This bit is set if a stop bit is not found within the appropriate period of time. Reading SP_STAT clears this bit.
5	TI	Transmit Interrupt	0	This bit is set at the beginning of the stop bit transmission. Reading SP_STAT clears this bit.
6	RI	Receive Interrupt	0	This bit is set when the last data bit is sampled. Reading SP_STAT clears this bit. This bit need **not** be clear for the serial port to receive data.
7	RPE/RB8	Received Parity Error/ Received Bit 8	0	RPE is set if parity is disabled (SP_CON.2=0) and the ninth data bit received is high. RB8 is set if parity is enabled (SP_CON.2=1) and a parity error occurred. Reading SP_STAT clears this bit.

Timer 2 Capture Register

T2CAPTURE
0D/0CH
HWindow 15 (Read/Write)

A rising edge on P2.7 causes the value of Timer 2 to be captured into the T2CAPTURE register and generates a Timer 2 Capture interrupt (INT11, 2036H). INT_MASK1.3 must be set to enable the interrupt.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	T2_VALUE (LO)	Timer 2 Low Byte	XXXX XXXX	These bits contain the low byte of the captured value of Timer 2.
8–15	T2_VALUE (HI)	Timer 2 High Byte	XXXX XXXX	These bits contain the high byte of the captured value of Timer 2.

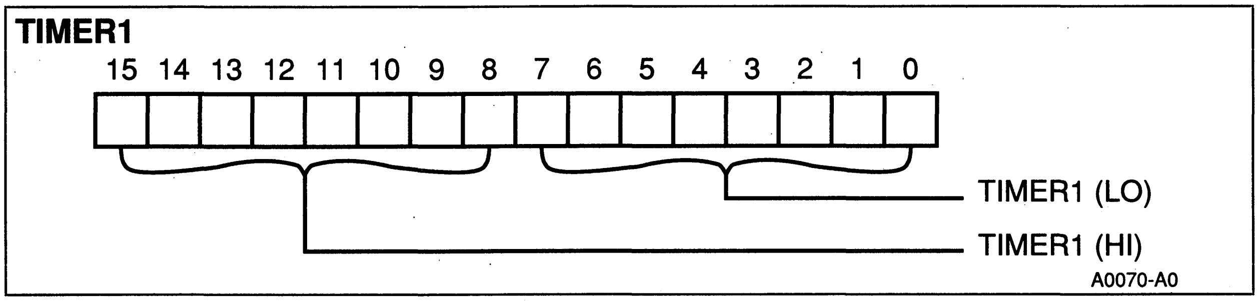
Timer 1 Register

TIMER1
0B/0AH
HWindow 0 (Read), HWindow 15 (Write)

The two bytes of the TIMER1 register contain the value of Timer 1. This register can be written, allowing Timer 1 to be initialized to a value other than zero.

Timer 1 is a 16-bit, free-running timer that is incremented every eight state times. Setting IOC1.2 causes an overflow to generate a Timer Overflow interrupt (INT00, 2000H) if INT_MASK.0 is also set, enabling the interrupt.

Use caution when writing to Timer 1 if the HSI and HSO modules are in use. HSO time entries in the CAM depend on exact matches with Timer 1. Also, changing Timer 1 between incoming events on the HSI lines will corrupt relative references between events.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	TIMER1 (LO)	Timer 1 Value, High Byte	0000 0000	These bits constitute the low byte of the Timer 1 value.
8–15	TIMER1 (HI)	Timer 1 Value, Low Byte	0000 0000	These bits constitute the high byte of the Timer 1 value.

Timer 2 Register

TIMER2
0D/0CH
HWindow 0 (Read/Write)

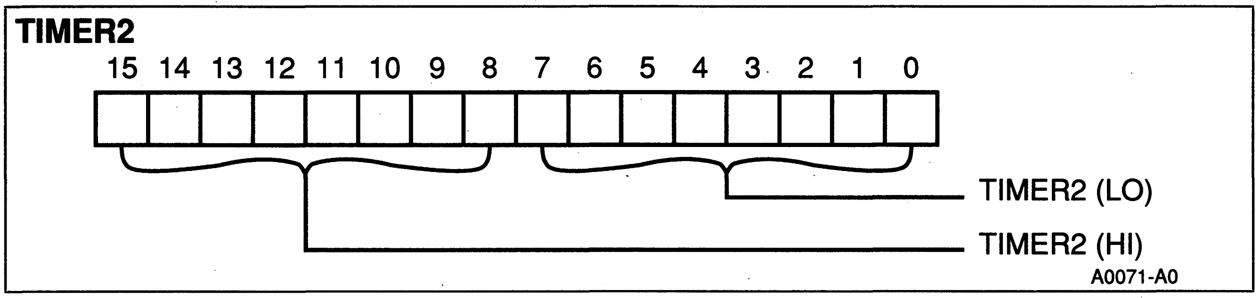
The two bytes of the TIMER2 register contain the value of Timer 2. This register can be written, allowing Timer 2 to be initialized to a value other than zero.

Timer 2 is a 16-bit counter that can be clocked either internally or externally, depending on the state of IOC3.0. If an external clock source is selected, IOC0.7 controls which of two external sources is used. Depending on the state of IOC2.0, Timer 2 counts every eight clocks (normal mode) or every clock (fast increment mode).

IOC2.1 controls the direction of Timer 2. With IOC2.1 clear, Timer 2 counts up. With IOC2.1 set, Timer 2 counts up or down, depending on the state of P2.6. Use caution when this feature is enabled and working with the HSO. If Timer 2 does not count through all 64K values, events in the CAM may never match the timer.

A Timer 2 overflow can generate either a Timer Overflow interrupt (INT00, 2000H) or a Timer 2 Overflow interrupt (INT12, 2038H). Set or clear IOC1.3, INT_MASK.0, and INT_MASK1.4 to configure Timer 2 for the desired interrupt.

This register can be cleared by hardware, software, or an HSO command, depending on the states of IOC0.3, IOC0.1, and HSO_COMMAND.0–HSO_COMMAND.3, respectively.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	TIMER2 (LO)	Timer 2 Value, Low Byte	0000 0000	These bits constitute the low byte of the Timer 2 value.
8–15	TIMER2 (HI)	Timer 2 Value, High Byte	0000 0000	These bits constitute the high byte of the Timer 2 value.

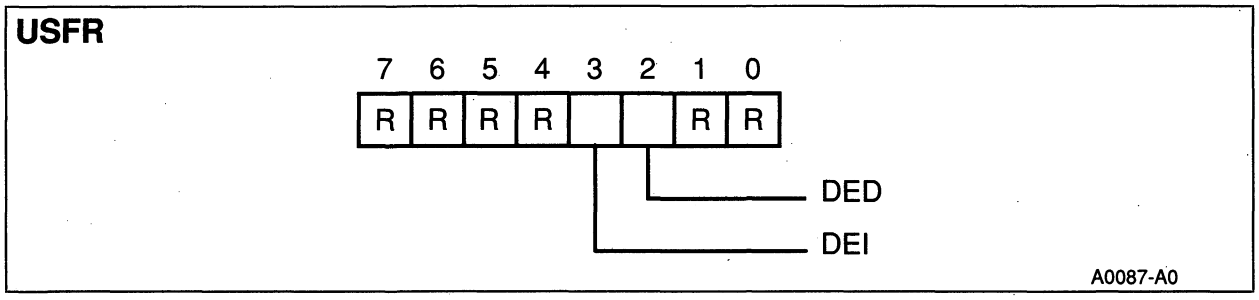
UPROM Special Function Register

USFR
1FF6H (Read)

The UPROM (Uneraseable PROM) SFR contains two bits that disable external fetches of data and instructions. These bits can be programmed, but cannot be erased. (Refer to Chapter 14, "Programming the Non-Volatile Memory," for details.) The remaining bits are reserved.

WARNING

These bits can be programmed, but can never be erased. Programming these bits makes dynamic failure analysis impossible. For this reason, devices with programmed UPROM bits cannot be returned to Intel for failure analysis.

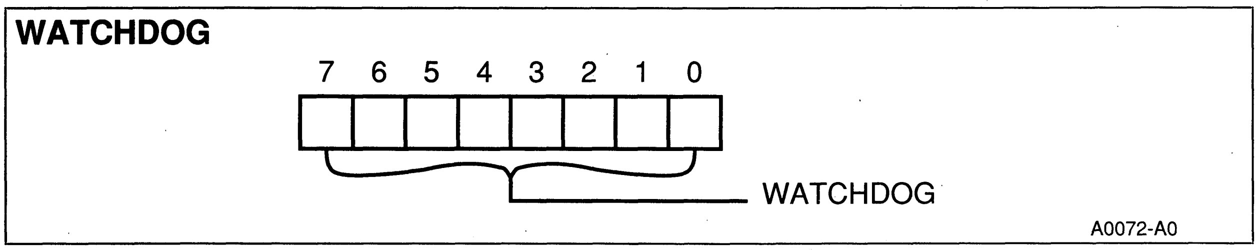


Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–1	—	—	00	Reserved; always write as zero.
2	DED	Disable External Data fetch	—	Setting this bit prevents the bus controller from executing external data reads and writes. Any attempt to access data through the bus controller initiates a reset.
3	DEI	Disable External Instruction fetch	—	Setting this bit prevents the bus controller from executing external instruction fetches. Any attempt to load an external address initiates a reset.
4–7	—	—	0000	Reserved; always write as zero.

Watchdog Timer Register

WATCHDOG
0AH
HWindow 0 (Clear/Enable), HWindow 15 (Read)

Unless it is cleared every 64K state times, the watchdog timer resets the 8XC196KC/KD. To clear the watchdog timer, send a "1EH" followed immediately by an "E1H" to location 0AH in HWindow 0. Clearing this register the first time enables the watchdog with an initial value of 0000H, which is incremented once every state time. After it is enabled, the watchdog can be disabled only by a reset. The eight most-significant bits of the watchdog value can be read in HWindow 15.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	WATCHDOG	Watchdog Timer Value	XXXX XXXX	This byte contains the 8 most-significant bits of the current value of the Watchdog timer.

Window Select Register

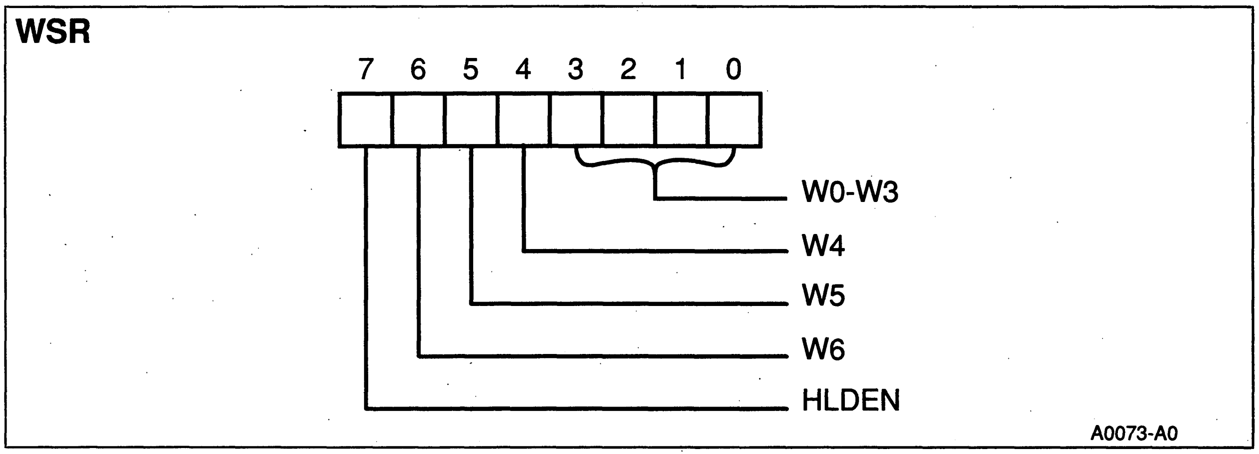
WSR
14H
All HWindows (Read/Write)

The WSR has two functions. One bit enables and disables the bus-hold protocol. The remaining bits select horizontal windows (HWindows) and vertical windows (VWindows). PUSHA saves this register on the stack, and POPA restores it.

Horizontal windows (HWindows) allow access to the Special Function Registers (SFRs) by mapping a 24-byte window into the lowest 24 bytes of the Lower Register File. HWindows 0, 1, and 15 are implemented on the 8XC196KC/KD. All other HWindows are reserved.

Vertical windows (VWindows) map sections of RAM into the upper section of the Lower Register File, in 32-, 64-, or 128-byte increments. The 8XC196KC has 512 bytes of internal RAM, which can be mapped into sixteen 32-byte VWindows, eight 64-byte VWindows, or four 128-byte VWindows. The 8XC196KD has 1024 bytes of internal RAM, which can be mapped into thirty-two 32-byte VWindows, sixteen 64-byte VWindows, or eight 128-byte VWindows.

HWindows and VWindows cannot be active at the same time. To switch between windows, write the window number into bits 0–3 and set or clear bits 4–6, as appropriate. See the window selection tables for clarification and refer to the "Memory Space Partitioning" chapter for additional information.



WSR

Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–3	W0–W3	Window Select Bits 0–3	0000	Write the number of the desired HWindow or VWindow into these four bits and select the window size by setting either WSR.4, WSR.5, or WSR.6. Valid HWindows are 0, 1, and 15. Valid VWindows are 0–15 for the 8XC196KC and 0–31 for the 8XC196KD. To select VWindows 16–31, you must also set both WSR.4 and WSR.6.
4	W4	Window Select Bit 4	X	Set this bit to specify a 128-byte VWindow or, for the 8XC196KD only, to specify a 32-byte VWindow from 16–31. Otherwise, clear it. On the 8XC196KC, setting this bit selects a 128-byte VWindow. On the 8XC196KD, if WSR.6 is clear, setting this bit selects a 128-byte VWindow. If WSR.6 is set, this bit functions as the high bit of the VWindow number for 32-byte VWindows 16–31.
5	W5	Window Select Bit 5	X	Set this bit to specify a 64-byte VWindow. Otherwise, clear it.
6	W6	Window Select Bit 6	X	Set this bit to specify a 32-byte VWindow. Otherwise, clear it.
7	HLDEN	HOLD#/ HLDA# Protocol Enable	0	This bit enables or disables the bus-hold protocol. 1= enable 0= disable

HWindow Selections

Desired HWindow	Device	WSR Bits								Hex Value
		7	6	5	4	3	2	1	0	
HWindow 0	8XC196KC/KD	X	0	0	0	0	0	0	0	00H
HWindow 1	8XC196KC/KD	X	0	0	0	0	0	0	1	01H
HWindow 15	8XC196KC/KD	X	0	0	0	1	1	1	1	15H

WSR

128-Byte VWindow Selections

Desired VWindow	Address to Remap	Device	WSR Bits 7	6	5	4	3	2	1	0	Hex Value
0	0000H	8XC196KC/KD	X	0	0	1	0	0	0	0	10H
1	0080H	8XC196KC/KD	X	0	0	1	0	0	0	1	11H
2	0100H	8XC196KC/KD	X	0	0	1	0	0	1	0	12H
3	0180H	8XC196KC/KD	X	0	0	1	0	0	1	1	13H
4	0200H	8XC196KD	X	0	0	1	0	1	0	0	14H
5	0280H	8XC196KD	X	0	0	1	0	1	0	1	15H
6	0300H	8XC196KD	X	0	0	1	0	1	1	0	16H
7	0380H	8XC196KD	X	0	0	1	0	1	1	1	17H

64-Byte VWindow Selections

Desired VWindow	Address to Remap	Device	WSR Bits 7	6	5	4	3	2	1	0	Hex Value
0	0000H	8XC196KC/KD	X	0	1	0	0	0	0	0	20H
1	0040H	8XC196KC/KD	X	0	1	0	0	0	0	1	21H
2	0080H	8XC196KC/KD	X	0	1	0	0	0	1	0	22H
3	00C0H	8XC196KC/KD	X	0	1	0	0	0	1	1	23H
4	0100H	8XC196KC/KD	X	0	1	0	0	1	0	0	24H
5	0140H	8XC196KC/KD	X	0	1	0	0	1	0	1	25H
6	0180H	8XC196KC/KD	X	0	1	0	0	1	1	0	26H
7	01C0H	8XC196KC/KD	X	0	1	0	0	1	1	1	27H
8	0200H	8XC196KD	X	0	1	0	1	0	0	0	28H
9	0240H	8XC196KD	X	0	1	0	1	0	0	1	29H
10	0280H	8XC196KD	X	0	1	0	1	0	1	0	2AH
11	02C0H	8XC196KD	X	0	1	0	1	0	1	1	2BH
12	0300H	8XC196KD	X	0	1	0	1	1	0	0	2CH
13	0340H	8XC196KD	X	0	1	0	1	1	0	1	2DH
14	0380H	8XC196KD	X	0	1	0	1	1	1	0	2EH
15	03C0H	8XC196KD	X	0	1	0	1	1	1	1	2FH

WSR

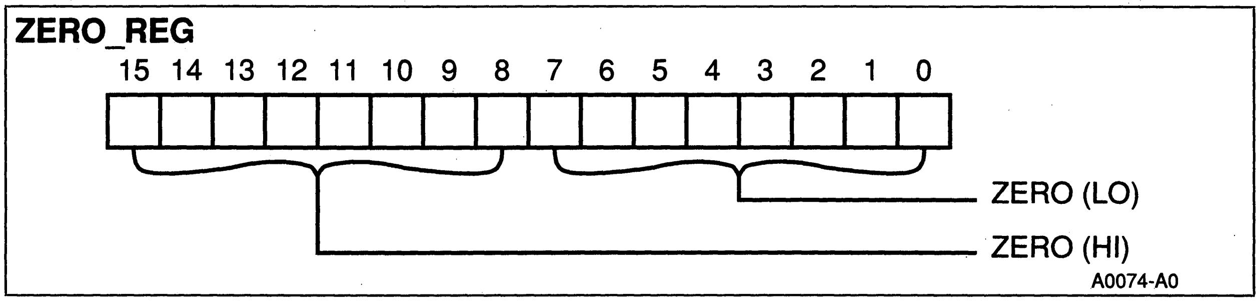
32-Byte VWindow Selections

Desired VWindow	Address to Remap	Device	WSR Bits 7	6	5	4	3	2	1	0	Hex Value
0	0000H	8XC196KC/KD	X	1	0	0	0	0	0	0	40H
1	0020H	8XC196KC/KD	X	1	0	0	0	0	0	1	41H
2	0040H	8XC196KC/KD	X	1	0	0	0	0	1	0	42H
3	0060H	8XC196KC/KD	X	1	0	0	0	0	1	1	43H
4	0080H	8XC196KC/KD	X	1	0	0	0	1	0	0	44H
5	00A0H	8XC196KC/KD	X	1	0	0	0	1	0	1	45H
6	00C0H	8XC196KC/KD	X	1	0	0	0	1	1	0	46H
7	00E0H	8XC196KC/KD	X	1	0	0	0	1	1	1	47H
8	0100H	8XC196KC/KD	X	1	0	0	1	0	0	0	48H
9	0120H	8XC196KC/KD	X	1	0	0	1	0	0	1	49H
10	0140H	8XC196KC/KD	X	1	0	0	1	0	1	0	4AH
11	0160H	8XC196KC/KD	X	1	0	0	1	0	1	1	4BH
12	0180H	8XC196KC/KD	X	1	0	0	1	1	0	0	4CH
13	01A0H	8XC196KC/KD	X	1	0	0	1	1	0	1	4DH
14	01C0H	8XC196KC/KD	X	1	0	0	1	1	1	0	4EH
15	01E0H	8XC196KC/KD	X	1	0	0	1	1	1	1	4FH
16	0200H	8XC196KD	X	1	0	1	0	0	0	0	50H
17	0220H	8XC196KD	X	1	0	1	0	0	0	1	51H
18	0240H	8XC196KD	X	1	0	1	0	0	1	0	52H
19	0260H	8XC196KD	X	1	0	1	0	0	1	1	53H
20	0280H	8XC196KD	X	1	0	1	0	1	0	0	54H
21	02A0H	8XC196KD	X	1	0	1	0	1	0	1	55H
22	02C0H	8XC196KD	X	1	0	1	0	1	1	0	56H
23	02E0H	8XC196KD	X	1	0	1	0	1	1	1	57H
24	0300H	8XC196KD	X	1	0	1	1	0	0	0	58H
25	0320H	8XC196KD	X	1	0	1	1	0	0	1	59H
26	0340H	8XC196KD	X	1	0	1	1	0	1	0	5AH
27	0360H	8XC196KD	X	1	0	1	1	0	1	1	5BH
28	0380H	8XC196KD	X	1	0	1	1	1	0	0	5CH
29	03A0H	8XC196KD	X	1	0	1	1	1	0	1	5DH
30	03C0H	8XC196KD	X	1	0	1	1	1	1	0	5EH
31	03E0H	8XC196KD	X	1	0	1	1	1	1	1	5FH

Zero Register

ZERO_REG
01/00H
All HWindows (Read/Write)

The two-byte Zero register is always equal to zero. It is useful as a fixed source of the constant zero for comparisons and calculations. ZERO_REG can also be used as the WORD variable in a long-indexed reference. This combination of register selection and address mode enables direct addressing of any location in memory. A CMPL (compare long) instruction with ZERO_REG forces a compare with a "generated" 32-bit zero value.



Bit Number(s)	Bit Mnemonic	Bit Name	Reset State	Description
0–7	ZERO (LO)	Zero Register Low Byte	0000 0000	This is the low byte of the zero register.
8–15	ZERO (HI)	Zero Register High Byte	0000 0000	This is the high byte of the zero register.

GLOSSARY

This glossary defines acronyms, abbreviations, and terms that have special meaning in this manual. (Chapter 1 discusses notational conventions and general terminology.)

Absolute Error	The maximum difference between corresponding actual and ideal code transitions. *Absolute error* accounts for all deviations of an actual A/D converter from an ideal converter.
Accumulator	A register or storage location that forms the result of an arithmetic or logic operations. The *RALU* performs most calculations, but it does not use an accumulator. Instead, it operates directly on the lower Register File, which essentially provides 256 accumulators. Because data does not flow through a single accumulator, code executes faster and more efficiently.
Actual Characteristic	A graph of output code versus input voltage of an actual A/D converter. An *actual characteristic* may vary with temperature, supply voltage, and frequency conditions.
A/D Converter	Analog-to-digital converter.
ALU	Arithmetic-Logic Unit. The part of the CPU that processes arithmetic and logical operations.
Assert	The term *assert* refers to the act of making a signal active (enabled). The polarity (high/low) is defined by the signal name. Active-low signals are designated by a pound symbol (#) suffix; active-high signals have no suffix. To *assert* RD# is to drive it low; to *assert* ALE is to drive it high.
Attenuation	A decrease in amplitude; voltage decay.
Bit	A binary digit.
BIT	A single-bit operand that can take on the Boolean values, "true" and "false."
Break-Before-Make	The property of a multiplexer which guarantees that a previously selected channel is deselected before a new channel is selected. (That is, *break-before-make* ensures that the A/D converter will not short inputs together.)

Byte	Any 8-bit unit of data.
BYTE	An unsigned, 8-bit variable with values from 0 through 255.
CAM	See *HSO CAM*.
CCB	Chip Configuration Byte, which is loaded into the Chip Configuration Register (CCR) unless the device is entering programming modes, in which case the *PCCB* is used.
Channel-to-Channel Matching Error	The difference between corresponding code transitions of *actual characteristics* taken from different channels under the same temperature, voltage, and frequency conditions. This error is caused by differences in DC leakage current and on-channel resistance from one multiplexer channel to another.
Characteristic	A graph of output code versus input voltage; the *transfer function* of the A/D converter.
Clear	The term *clear* refers to the value of a bit or the act of giving it a value. If a bit is *clear,* its value is "0"; *clearing* a bit gives it a "0" value.
Code	The digital value output by the A/D converter.
Code Center	The voltage corresponding to the midpoint between two adjacent code transitions on the A/D converter.
Code Transition	The point at which the A/D converter output code changes from "Q" to "Q+1." The input voltage corresponding to a code transition is defined as the voltage that is equally likely to produce either of two adjacent codes.
Code Width	The voltage change corresponding to the difference between two adjacent *code transitions*. Code width deviations cause *differential non-linearity* and *non-linearity* errors.
Crosstalk	See *Off-isolation*.
DC Input Leakage	Leakage current to ground from an analog input pin.

Deassert	The term *deassert* refers to the act of making a signal inactive (disabled). The polarity (high/low) is defined by the signal name. Active-low signals are designated by a pound symbol (#) suffix; active-high signals have no suffix. To *deassert* RD# is to drive it high; to *deassert* ALE is to drive it low.
Differential Non-Linearity	The difference between the actual *code width* and the ideal one-LSB code width of the *terminal-based characteristic* of an A/D converter. It provides a measure of how much the input voltage may have changed in order to produce a one-count change in the conversion result. *Differential non-linearity* is a measure of local code-width error; *non-linearity* is a measure of overall code-transition error.
Doping	The process of introducing a periodic table Group III or Group V element into a Group IV element (e.g., silicon). A Group III impurity (e.g., indium or gallium) results in a *p-type* material. A Group V impurity (e.g., arsenic or antimony) results in an *n-type* material.
Double-Word	Any 32-bit unit of data.
DOUBLE-WORD	An unsigned, 32-bit variable with values from 0 through 4,294,967,295.
ESD	Electrostatic discharge.
Event	See *HSI Event* and *HSO Event.*
Feedthrough	The attenuation from an input voltage on the selected channel to the A/D output after the *sample window* closes. The ability of the A/D converter to reject an input on its selected channel after the *sample window* closes.
FET	Field-effect transistor.
FIFO	See *HSI FIFO.*
Full-Scale Error	The difference between the ideal and actual input voltage corresponding to the final (full-scale) code transition.
Hold Latency	The time it takes the 8XC196KC/KD to assert HLDA# after the external device asserts HOLD#.

Horizontal Windowing	The mapping of a 24-byte window of RAM into the lowest 24 bytes of the lower Register File, allowing access to the Special Function Registers (SFRs). The RALU has direct control of all peripheral modules, except Ports 3 and 4, through the SFRs.
HSI	High-Speed Input module. The 8XC196KC/KD module that can record times of external events.
HSI Event	An occurrence on one of the HSI pins (HSI.0–HSI.3) that causes the HSI module to capture information and store it in the *HSI FIFO*. An *HSI event* can be eight positive transitions, each positive transition, each negative transition, or every positive and every negative transition.
HSI FIFO	The HSI's First-In-First-Out file. The portion of the HSI that stores the Timer 1 value and the associated event status bits when an *HSI event* occurs.
HSIO	High-Speed Input/Output unit. The timer/counter-based I/O system that consists of four individual modules: Timer 1, Timer 2, HSI, and HSO.
HSO	High-Speed Output module. The 8XC196KC/KD module that can trigger programmable events at specified times. The HSO can also be used to generate analog outputs, as can the three *PWM*s.
HSO CAM	The HSO's Content-Addressable Memory file. The portion of the HSO that stores commands to be executed and the associated time tags.
HSO Event	An occurrence that the HSO module is programmed to trigger at a specific time. Of the 15 programmable *HSO events*, the 9 that toggle one or more of the HSO pins are called *external events*, and the remaining 6 that program software timers 0–3, reset Timer 2, and start an A/D conversion are called *internal events*. The HSO module can be programmed to generate the High-Speed Output interrupt as the result of an *external event* and the Software Timer interrupt as the result of an *internal event*.
HWindow	Horizontal window. HWindows 0, 1, and 15 are implemented on the 8XC196KC/KD. Each HWindow provides access to a unique set of SFRs.

Ideal Characteristic	A graph of output code versus input voltage of an ideal A/D converter. The *ideal characteristic* possesses unique features: its first code transition occurs when the input voltage is 0.5 LSB; its full-scale (final) code transition occurs when the input voltage equals the full-scale reference minus 1.5 LSB; and its code widths are all exactly one LSB. These properties result in a conversion without zero offset, full-scale, or linearity errors. *Quantizing error* is the only error seen in an ideal A/D converter.
Input Leakage	Current leakage from an analog input pin to ground.
Input Series Resistance	The effective series resistance from the analog input pin to the sample capacitor.
Integer	Any member of the set consisting of the positive and negative whole numbers and zero.
INTEGER	A 16-bit, signed variable with values from –32,768 through +32,767.
Interrupt Controller	The module responsible for handling interrupts that are to be serviced by user-written interrupt service routines. Also called the *Programmable Interrupt Controller (PIC).*
Interrupt Latency	The total delay between the time that an interrupt is generated (not acknowledged) and the time that the 8XC196KC/KD begins executing the interrupt service routine or PTS cycle.
Interrupt Service Routine	The software routine that services a standard interrupt.
LONG-INTEGER	A 32-bit, signed variable with values from –2,147,483,648 through +2,147,483,647.
LSB	Least-significant bit of a byte or least-significant byte of a word. For A/D conversions, 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 V, one LSB is 5.0 mV. Note that this differs from digital LSBs. When referring to an A/D converter, an uncertainty of two LSBs equals 10 mV. (This has

	been confused with an uncertainty of two digital bits, which would mean four counts, or 20 mV.)
Maskable Interrupts	All 8XC196KC/KD interrupts except Unimplemented Opcode, Software Trap, and NMI. *Maskable interrupts* can be disabled (masked) by the individual mask bits in the interrupt mask registers, and their servicing can be disabled by the global interrupt enable bit. Each *maskable interrupt* can be assigned to the PTS for processing.
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. (In other words, a converter is monotonic if every code change represents an input voltage change in the same direction.) Large *differential non-linearity* errors can cause the converter to exhibit non-monotonic behavior.
MSB	Most-significant bit of a byte or most-significant byte of a word.
***n*-Channel FET**	A field-effect transistor with an *n*-type conducting path (channel).
***n*-Type Material**	Semiconductor material with introduced impurities (*doping*) causing it to have an excess of negatively charged carriers.
No Missing Codes	An A/D converter has *no missing codes* if, for every output code, there exists a unique input voltage range which produces that code only. Large *differential non-linearity* errors can cause the converter to miss codes.
Non-Linearity	The maximum deviation of *code transitions* of the *terminal-based characteristic* from the corresponding code transitions of the *ideal characteristic*.
Nonmaskable Interrupts	Interrupts that cannot be masked (disabled) and cannot be assigned to the PTS for processing. The *nonmaskable interrupts* are Unimplemented Opcode, Software Trap, and NMI.
***npn* Transistor**	A transistor consisting of one part *p*-type material and two parts *n*-type material.

Off-Isolation	The attenuation from a deselected input to the A/D output. The ability of an A/D converter to reject (isolate) the signal on a deselected (off) input.
OTPROM	One-Time-Programmable Read-Only Memory, a version of EPROM.
***p*-Channel FET**	A field-effect transistor with a *p*-type conducting path.
***p*-Type Material**	Semiconductor material with introduced impurities (*doping*) causing it to have an excess of positively charged carriers.
PC	Program Counter.
PCCB	Programming Chip Configuration Byte, which is loaded into the Chip Configuration Register (CCR) when the device is entering programming modes; otherwise, the *CCB* is used.
PIC	Programmable Interrupt Controller. The module responsible for handling interrupts that are to be serviced by user-written interrupt service routines. Also called simply the *Interrupt Controller*.
Prioritized Interrupt	Any of the fifteen *maskable interrupts* or the *nonmaskable* NMI. Two of the *nonmaskable interrupts* (Unimplemented Opcode and Software Trap) are not prioritized; they go directly to the Interrupt Controller for servicing.
PSW	Program Status Word. The PSW contains one bit that globally enables or disables servicing of all maskable interrupts, one bit that enables or disables the *PTS*, and six Boolean flags that reflect the state of the user's program.
PTS	Peripheral Transaction Server, the 8XC196KC/KD's microcoded hardware interrupt processor.
PTSCB	PTS Control Block. A block of data required for each PTS interrupt. The microcode executes the proper *PTS cycle* based on the contents of the PTSCB.

PTS Cycle	The PTS microcode equivalent of an interrupt service routine. It is the complete microcoded response to a *single* PTS interrupt. In Single Transfer mode, each byte or word transfer is a *PTS cycle.* In Block Transfer mode, each byte or word transfer is called a *PTS transfer*, while the entire block transfer is called a *PTS cycle.*
PTS Interrupt	Any *maskable interrupt* that is assigned to the PTS for interrupt processing.
PTS Mode	A microcoded response that enables the PTS to quickly complete a specific task. These tasks include single byte/word transfers, block byte/word transfers, multiple A/D conversions, HSI FIFO dumps, and HSO CAM loads.
PTS Transfer	The movement of a single byte or word from the source memory location to the destination memory location. In Single Transfer mode, a *PTS transfer* constitutes a *PTS cycle*, since only one transfer is to occur in the PTS cycle. In Block Transfer mode, the movement of each byte or word within the block is a *PTS transfer*; the movement of the entire block is a *PTS cycle.*
PWM	Pulse Width Modulator. The 8XC196KC/KD has three PWMs and an *HSO* module, all of which can be used to generate analog outputs.
Quantizing Error	An unavoidable A/D conversion error that results simply from the conversion of a continuous voltage to its integer digital representation. Quantizing error is always ± 0.5 LSB and is the only error present in an ideal converter.
RALU	Register Arithmetic-Logic Unit. A part of the CPU that consists of the *ALU*, the *PSW*, the master *PC*, the microcode engine, a loop counter, and six registers.
Repeatability Error	The difference between corresponding code transitions from different *actual characteristics* taken from the same converter on the same channel with the same temperature, voltage, and frequency conditions. The amount of *repeatability error* depends on the ability of the comparator to resolve very similar voltages and the extent to which random noise contributes to the error.

Sample Delay	The time period between the time that A/D converter receives the "start conversion" signal and the time that the sample capacitor is connected to the selected channel.
Sample Delay Uncertainty	The variation in the *sample delay.*
Sample Time	The period of time that the *sample window* is open. (That is, the length of time that the analog input channel is actually connected to the sample capacitor.)
Sample Time Uncertainty	The variation in the *sample time.*
Sample Window	The period of time that begins when the sample capacitor is attached to a selected channel and ends when the sample capacitor is disconnected from the selected channel.
Sampled Inputs	All input pins, with the exception of RESET#, are *sampled inputs.* The input pin is sampled one state time before the read buffer is enabled. Sampling occurs during PH1 (while CLKOUT is low) and resolves the value (high or low) of the pin before it is presented to the internal bus. If the pin value changes during the sample time, the new value may or may not be recorded during the read. RESET# is a level-sensitive input. EXTINT is normally a sampled input; however, during Powerdown mode, the powerdown circuitry uses EXTINT as a level-sensitive input.
SAR	*Successive approximation* register; a component of the A/D converter.
Set	The term *set* refers to the value of a bit or the act of giving it a value. If a bit is *set,* its value is "1"; *setting* a bit gives it a "1" value.
SFR	Special Function Register.
SHORT-INTEGER	An 8-bit, signed variable with values from –128 through +127.
Sink Current	Current flowing **into** a device to ground. Always a positive value.

Source Current	Current flowing **out of** a device from V_{CC}. Always a negative value.
SP	Stack Pointer.
Special Interrupt	Any of the three *nonmaskable interrupts* (Unimplemented Opcode, Software Trap, or NMI).
Standard Interrupt	Any *maskable interrupt* that is assigned to the *Interrupt Controller* for processing by interrupt service routines.
State Time (or State)	The basic time unit of the 8XC196KC/KD; the combined period of the two internal timing signals, PH1 and PH2. (The internal clock generator produces PH1 and PH2 by halving the frequency of the signal on XTAL1. The rising edges of the active-high PH1 and PH2 signals generate CLKOUT, the output of the internal clock generator.) With a 20 MHz crystal, one *state time* equals 100 ns. Because the 8XC196KC/KD can operate at many frequencies, this manual defines time requirements in terms of *state times* rather than in specific units of time.
Successive Approximation	An A/D conversion method that 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*.
Temperature Drift	The rate at which typical specifications change with respect to a change in temperature. These changes are reflected in the *temperature coefficients*.
Terminal-Based Characteristic	An *actual characteristic* that has been translated and scaled to remove *zero offset* error and *full-scale error*. A *terminal-based characteristic* resembles an *actual characteristic* with *zero offset* and *full-scale error* removed.
Transfer Function	A graph of output code versus input voltage; the *characteristic* of the A/D converter.

Transfer Function Errors	Errors inherent in an analog-to-digital conversion process: *quantizing error*, *zero offset error*, *full-scale error*, *differential non-linearity*, and *non-linearity*. Errors that are hardware-dependent, rather than being inherent in the process itself, include *feedthrough*, *repeatability*, *channel-to-channel matching*, *off-isolation*, and *V_{CC} rejection* errors.
UART	Universal Asynchronous Receiver and Transmitter. A part of the serial I/O port.
V_{CC} Rejection	The property of an A/D converter that causes it to ignore (reject) changes in V_{CC} so that the *actual characteristic* is unaffected by those changes. The effectiveness of *V_{CC} rejection* is measured by the ratio of the change in V_{CC} to the change in the *actual characteristic*.
Vertical Windowing	The mapping of sections of RAM into the upper section of the lower Register File in 32-, 64-, or 128-byte increments. *Vertical windowing* enables the RALU to use register-direct addressing to access the RAM in the upper Register File.
VWindow	Vertical window. The 8XC196KC has 512 bytes of internal RAM, which can be mapped into sixteen 32-byte VWindows, eight 64-byte VWindows, or four 128-byte VWindows. Since the 8XC196KD has twice as much RAM, it also has twice as many VWindows.
WDT	Watchdog timer, an internal timer that resets the device if the software fails to operate properly.
Word	Any 16-bit unit of data.
WORD	An unsigned, 16-bit variable with values from 0 through 65535.
Zero Offset	The difference between the ideal and actual input voltage corresponding to the first *code transition*.

Index

INDEX

Intel around the world

United States and Canada
Intel Corporation
Robert Noyce Building
2200 Mission College Boulevard
P.O. Box 58119
Santa Clara, CA 95052-8119
USA
Phone: (800) 628-8686

Europe
Intel Corporation (UK) Ltd.
Pipers Way
Swindon
Wiltshire SN3 1RJ
UK

Phone:

England	(44) 1793 403 000
Germany	(49) 89 99143 0
France	(33) 1 4571 7171
Italy	(39) 2 575 441
Israel	(972) 2 589 7111
Netherlands	(31) 10 286 6111
Sweden	(46) 8 705 5600

Asia-Pacific
Intel Semiconductor Ltd.
32/F Two Pacific Place
88 Queensway, Central
Hong Kong, SAR
Phone: (852) 2844 4555

Japan
Intel Kabushiki Kaisha
P.O. Box 115 Tsukuba-gakuen
5-6 Tokodai, Tsukuba-shi
Ibaraki-ken 305
Japan
Phone: (81) 298 47 8522

South America
Intel Semicondutores do Brazil
Rue Florida, 1703-2 and CJ22
CEP 04565-001 Sao Paulo-SP
Brazil
Phone: (55) 11 5505 2296

To learn more about Intel Corporation, visit our Web site at www.intel.com

intel®