MC68HC705P9/D
REV.1

# HC05

# MC68HC705P9

TECHNICAL
DATA

MOTOROLA

# MC68HC705P9

# HCMOS MICROCONTROLLER UNIT

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

# TABLE OF CONTENTS

Paragraph | Title | Page

## SECTION 1 GENERAL DESCRIPTION

## SECTION 2 MEMORY

# TABLE OF CONTENTS (Continued)

Paragraph Title Page

## SECTION 3 CENTRAL PROCESSOR UNIT

## SECTION 4 INTERRUPTS

## SECTION 5 RESETS

# TABLE OF CONTENTS
# (Continued)

Paragraph Title Page

## SECTION 6
## LOW POWER MODES

## SECTION 7
## PARALLEL I/O

## SECTION 8
## CAPTURE/COMPARE TIMER

# TABLE OF CONTENTS
# (Continued)

Paragraph Title Page

## SECTION 9
## SERIAL I/O PORT (SIOP)

## SECTION 10
## ANALOG-TO-DIGITAL CONVERTER (ADC)

## SECTION 11
## EPROM/OTPROM

## SECTION 12
## INSTRUCTION SET

# TABLE OF CONTENTS (Concluded)

Paragraph Title Page

## SECTION 13 ELECTRICAL SPECIFICATIONS

## SECTION 14 MECHANICAL SPECIFICATIONS

# LIST OF FIGURES

# LIST OF FIGURES
# (Continued)

# LIST OF FIGURES (Concluded)

# LIST OF TABLES

# SECTION 1
# GENERAL DESCRIPTION

The MC68HC705P9 is a member of the low-cost, high-performance M68HC05 Family of 8-bit microcontroller units (MCUs). The M68HC05 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the popular M68HC05 central processor unit (CPU) and are available with a variety of subsystems, memory sizes and types, and package types.

On-chip memory of the MC68HC705P9 includes 2112 bytes of erasable, programmable ROM (EPROM). In packages without the transparent window for EPROM erasure, the 2112 EPROM bytes serve as one-time programmable ROM (OTPROM).

## 1.1 Features

Features of the MC68HC705P9 MCU include the following:

- Popular M68HC05 Central Processor Unit
- Memory-Mapped Input/Output (I/O) Registers
- 2112 Bytes of EPROM/OTPROM Including 48 Bytes of Page Zero EPROM/OTPROM and 16 Locations for User Vectors
- 128 Bytes of User RAM
- 20 Bidirectional I/O Port Pins and One Input-Only Port Pin
- Synchronous Serial I/O Port (SIOP)
- On-Chip Oscillator with Crystal or Ceramic Resonator Connections or External Clock Connections
- 16-Bit Capture/Compare Timer
- 4-Channel, 8-Bit Analog-to-Digital Converter (ADC)
- Fully Static Operation with no Minimum Clock Speed
- Bootloader ROM
- Power-Saving Stop, Wait, and Data-Retention Modes
- 8 X 8 Unsigned Multiply Instruction
- Single 3.3 V–5.0 V Power Supply Requirement
- 28-Pin Ceramic Dual-In-Line Package (Cerdip) with 2112 Bytes of EPROM
- 28-Pin Plastic Dual-In-Line Package (PDIP) with 2112 Bytes of OTPROM
- 28-Pin Small Outline Integrated Circuit (SOIC) Package with 2112 Bytes of OTPROM

## 1.2 Programmable Options

The following options are programmable in the mask option register (MOR):

- Enabled or disabled COP watchdog
- Edge-triggered or edge- and level-triggered external interrupt pin
- LSB-first or MSB-First SIOP data format

## 1.3 MCU Structure

Figure 1-1 shows the structure of the MC68HC705P9 MCU.

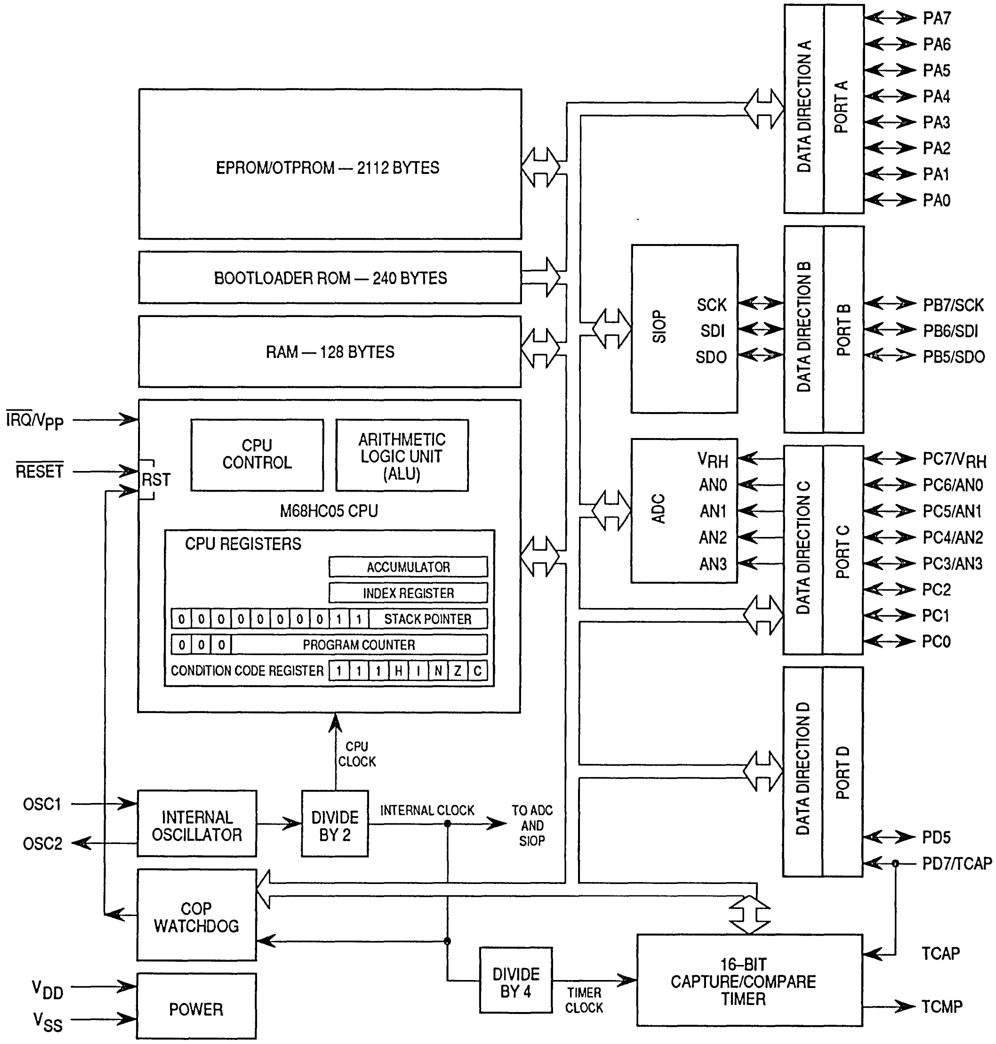


**Figure 1-1. MC68HC705P9 Block Diagram**

## 1.4 Pin Assignments

Figure 1-2 shows the MC68HC705P9 pin assignments.

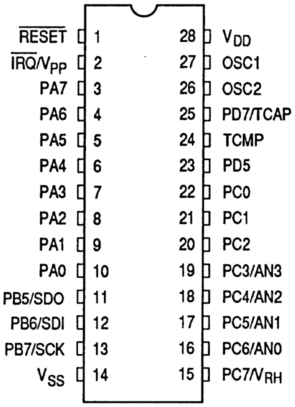


**Figure 1-2. Pin Assignments**

### 1.4.1 $V_{DD}$ and $V_{SS}$

$V_{DD}$ and $V_{SS}$ are the power supply and ground pins. The MCU operates from a single 5 V power supply.

Very fast signal transitions occur on the MCU pins, placing very high short-duration current demands on the power supply. To prevent noise problems, take special care to provide good power supply bypassing at the MCU. Place the C1 bypass capacitor as close to the MCU as possible, as Figure 1-3 shows. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.

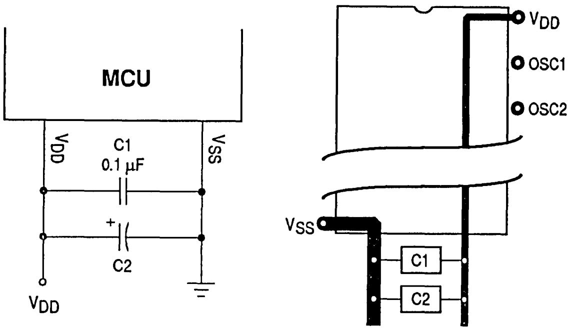


Figure 1-3. Bypassing Layout Recommendation

### 1.4.2 OSC1 and OSC2

The OSC1 and OSC2 pins are the control connections for the on-chip oscillator. The oscillator can be driven by any of the following:

- Crystal
- Ceramic resonator
- External clock signal

The MCU divides the frequency of the oscillator, $f_{OSC}$, or external clock source by two to produce the internal operating frequency, $f_{OP}$.

#### 1.4.2.1 Crystal

A crystal connected to the OSC1 and OSC2 pins can drive the on-chip oscillator. Figure 1-4 shows a typical crystal oscillator circuit for an AT-cut parallel resonant crystal. Follow the crystal supplier's recommendations, as the crystal parameters determine the external component values required to provide reliable start-up and maximum stability. The load capacitance values used in the oscillator circuit design should include all stray layout capacitances. To minimize output distortion, mount the crystal and capacitors as close as possible to the pins.

**NOTE**

Use an AT-cut crystal and not an AT-strip crystal. The MCU may overdrive an AT-strip crystal.

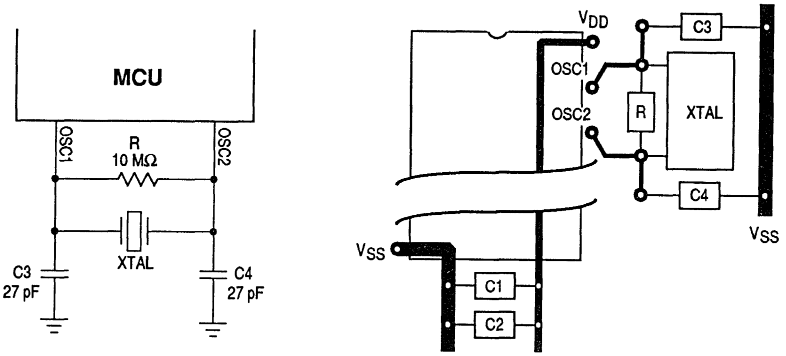


**Figure 1-4. Crystal Connections**

### 1.4.2.2 Ceramic Resonator

To reduce cost, use a ceramic resonator in place of the crystal. Figure 1-5 shows a ceramic resonator circuit. Follow the resonator manufacturer's recommendations for the values of any external components required. Loading capacitance values used in the oscillator circuit design should include all stray layout capacitances. To minimize output distortion, mount the resonator as close as possible to the pins.

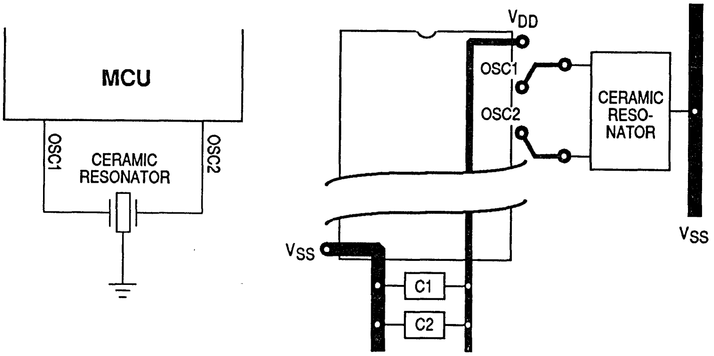


**Figure 1-5. Ceramic Resonator Connections**

**NOTE**

Because the frequency stability of ceramic resonators is not as high as that of crystal oscillators, using a ceramic resonator may degrade the performance of the A/D converter.

### 1.4.2.3 External Clock

An external clock from a CMOS-compatible device can drive the on-chip oscillator, as Figure 1-6 shows. Leave the OSC2 pin unconnected.

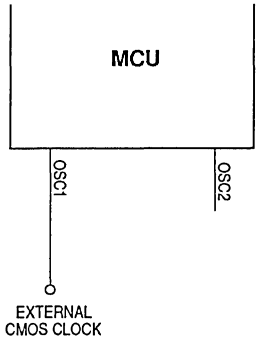


**Figure 1-6. External Clock Connections**

### 1.4.3 $\overline{\text{RESET}}$

A logic zero on the $\overline{\text{RESET}}$ pin forces the MCU to a known start-up state. (See **5.1.2 External Reset** for more information.)

### 1.4.4 $\overline{\text{IRQ}}/V_{PP}$

The $\overline{\text{IRQ}}/V_{PP}$ pin has the following functions:

- Applying asynchronous external interrupt signals (See **4.1.2 External Interrupt**.)
- Applying $V_{PP}$, the EPROM/OTPROM programming voltage (See **11.1 EPROM/OTPROM Programming**.)

### 1.4.5 PA7–PA0

PA7–PA0 are general-purpose bidirectional I/O port pins. (See **7.2 Port A**.)

### 1.4.6 PB7/SCK, PB6/SDI, and PB5/SDO

Port B is a three-pin bidirectional I/O port that shares its pins with the SIOP subsystem. (See **7.3 Port B**.)

### 1.4.7 PC7/$V_{RH}$, PC6/AN0, PC5/AN1, PC4/AN2, PC3/AN3, and PC2–PC0

Port C is an eight-bit bidirectional I/O port that shares five of its pins with the A/D converter. (See **7.4 Port C**.)

### 1.4.8 PD7/TCAP and PD5

Port D is a two-bit special-function port that shares one of its pins with the capture/compare timer. (See **7.5 Port D**.)

### 1.4.9 TCMP

TCMP is the output pin for the output compare feature of the capture/compare timer. (See **8.1.2 Output Compare**.)

# SECTION 2
# MEMORY

This section describes the organization of the on-chip memory.

## 2.1 Memory Map

The CPU can address 8 Kbytes of memory space. The ROM portion of memory holds the program instructions, fixed data, user-defined vectors, and interrupt service routines. The RAM portion of memory holds variable data. I/O registers are memory-mapped so that the CPU can access their locations in the same way that it accesses all other memory locations.

Figure 2-1 is a memory map of the MCU. Figure 2-2 is a more detailed memory map of the 32-byte I/O register section.

## 2.2 Input/Output Section

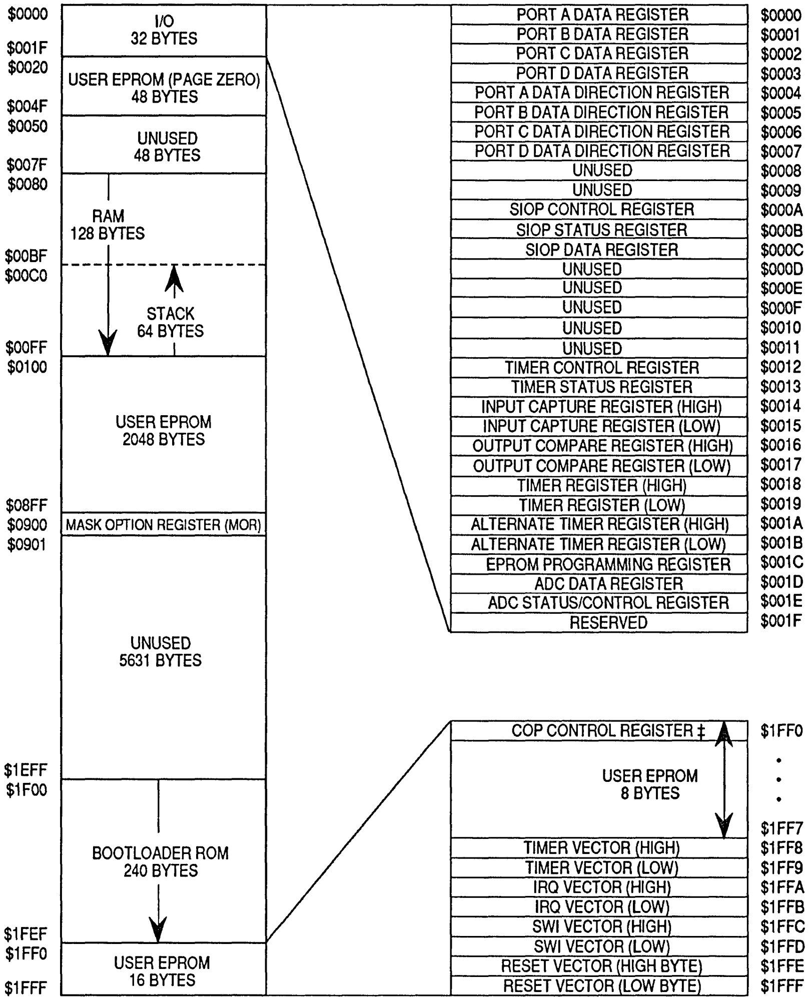
The first 32 addresses of the memory space, $0000–$001F, are the I/O section. These are the addresses of the I/O control registers, status registers, and data registers.

## 2.3 RAM

The 128 addresses from $0080–$00FF are RAM locations. The CPU uses the top 64 RAM addresses, $00C0–$00FF, as the stack. Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the CPU registers. During a subroutine call, the CPU uses two bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

### NOTE

Be careful when using nested subroutines or multiple interrupt levels. The CPU may overwrite data in the RAM during a subroutine or during the interrupt stacking operation.



| | |
|---|---|
| PORT A DATA REGISTER | $0000 |
| PORT B DATA REGISTER | $0001 |
| PORT C DATA REGISTER | $0002 |
| PORT D DATA REGISTER | $0003 |
| PORT A DATA DIRECTION REGISTER | $0004 |
| PORT B DATA DIRECTION REGISTER | $0005 |
| PORT C DATA DIRECTION REGISTER | $0006 |
| PORT D DATA DIRECTION REGISTER | $0007 |
| UNUSED | $0008 |
| UNUSED | $0009 |
| SIOP CONTROL REGISTER | $000A |
| SIOP STATUS REGISTER | $000B |
| SIOP DATA REGISTER | $000C |
| UNUSED | $000D |
| UNUSED | $000E |
| UNUSED | $000F |
| UNUSED | $0010 |
| UNUSED | $0011 |
| TIMER CONTROL REGISTER | $0012 |
| TIMER STATUS REGISTER | $0013 |
| INPUT CAPTURE REGISTER (HIGH) | $0014 |
| INPUT CAPTURE REGISTER (LOW) | $0015 |
| OUTPUT COMPARE REGISTER (HIGH) | $0016 |
| OUTPUT COMPARE REGISTER (LOW) | $0017 |
| TIMER REGISTER (HIGH) | $0018 |
| TIMER REGISTER (LOW) | $0019 |
| ALTERNATE TIMER REGISTER (HIGH) | $001A |
| ALTERNATE TIMER REGISTER (LOW) | $001B |
| EPROM PROGRAMMING REGISTER | $001C |
| ADC DATA REGISTER | $001D |
| ADC STATUS/CONTROL REGISTER | $001E |
| RESERVED | $001F |

| | |
|---|---|
| COP CONTROL REGISTER ‡ | $1FF0 |
| USER EPROM 8 BYTES | •<br>•<br>•<br>$1FF7 |
| TIMER VECTOR (HIGH) | $1FF8 |
| TIMER VECTOR (LOW) | $1FF9 |
| IRQ VECTOR (HIGH) | $1FFA |
| IRQ VECTOR (LOW) | $1FFB |
| SWI VECTOR (HIGH) | $1FFC |
| SWI VECTOR (LOW) | $1FFD |
| RESET VECTOR (HIGH BYTE) | $1FFE |
| RESET VECTOR (LOW BYTE) | $1FFF |

‡ WRITING 0 TO BIT 0 OF $1FF0 CLEARS COP TIMER. READING $1FF0 RETURNS USER EPROM DATA.

**Figure 2-1. Memory Map**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |  |
|---|---|---|---|---|---|---|---|---|---|
| $0000 | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 | PORTA |
| $0001 | PB7 | PB6 | PB5 | 0 | 0 | 0 | 0 | 0 | PORTB |
| $0002 | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 | PORTC |
| $0003 | PD7 | 0 | PD5 | 1 | 0 | 0 | 0 | 0 | PORTD |
| $0004 | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 | DDRA |
| $0005 | DDRB7 | DDRB6 | DDRB5 | 1 | 1 | 1 | 1 | 1 | DDRB |
| $0006 | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 | DDRC |
| $0007 | 0 | 0 | DDRD5 | 0 | 0 | 0 | 0 | 0 | DDRD |
| $0008 | — | — | — | — | — | — | — | — | UNUSED |
| $0009 | — | — | — | — | — | — | — | — | UNUSED |
| $000A | 0 | SPE | 0 | MSTR | 0 | 0 | 0 | 0 | SCR |
| $000B | SPIF | DCOL | 0 | 0 | 0 | 0 | 0 | 0 | SSR |
| $000C | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | SDR |
| $000D | — | — | — | — | — | — | — | — | UNUSED |
| $000E | — | — | — | — | — | — | — | — | UNUSED |
| $000F | — | — | — | — | — | — | — | — | UNUSED |
| $0010 | — | — | — | — | — | — | — | — | UNUSED |
| $0011 | — | — | — | — | — | — | — | — | UNUSED |
| $0012 | ICIE | OCIE | TOIE | 0 | 0 | 0 | IEDG | OLVL | TCR |
| $0013 | ICF | OCF | TOF | 0 | 0 | 0 | 0 | 0 | TSR |
| $0014 | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 | ICRH |
| $0015 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | ICRL |
| $0016 | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 | OCRH |
| $0017 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | OCRL |
| $0018 | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 | TRH |
| $0019 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | TRL |
| $001A | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 | ATRH |
| $001B | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | ATRL |
| $001C | 0 | 0 | 0 | 0 | 0 | LATCH | 0 | EPGM | PROG |
| $001D | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | ADDR |
| $001E | CCF | ADRC | ADON | 0 | 0 | 0 | CH1 | CH0 | ADSCR |
| $001F | — | — | — | — | — | — | — | — | RESERVED |
| $0900 | — | — | — | — | — | SIOP | IRQ | COP | MOR |
| $1FF0 | — | — | — | — | — | — | — | COPC | COPR |

**Figure 2-2. I/O Registers**

## 2.4 EPROM/OTPROM

An MCU with a quartz window has 2112 bytes of erasable, programmable ROM (EPROM). The quartz window allows EPROM erasure with ultraviolet light. In an MCU without the quartz window, the EPROM cannot be erased and serves as 2112 bytes of one-time programmable ROM (OTPROM). The following addresses are user EPROM/OTPROM locations:

- $0020–$004F
- $0100–$08FF
- $0900 (Mask option register)
- $1FF0–$1FF7
- $1FF8–$1FFF (reserved for user-defined interrupt and reset vectors)

## 2.5 Bootloader ROM

The 240 bytes at addresses $1F00–$1FEF are reserved ROM addresses that contain the instructions for the bootloader functions. (See **11.1 EPROM/OTPROM Programming**.)

# SECTION 3
# CENTRAL PROCESSOR UNIT

This section describes the CPU registers.

## 3.1 CPU Registers

Figure 3-1 shows the five CPU registers. CPU registers are not part of the memory map.

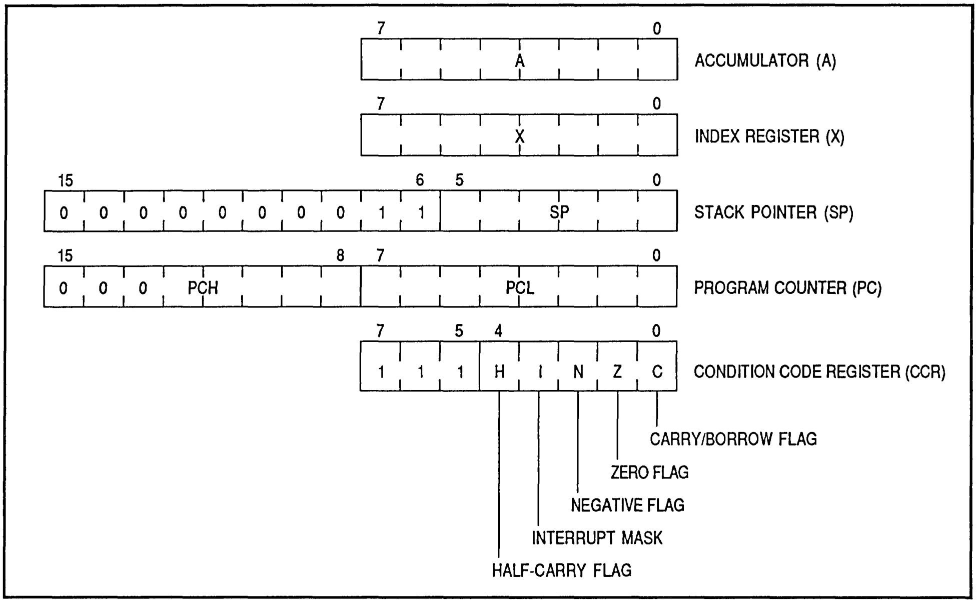


**Figure 3-1. Programming Model**

### 3.1.1 Accumulator

The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and the results of arithmetic and nonarithmetic operations.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 3-2. Accumulator**

### 3.1.2 Index Register

In the indexed addressing modes, the CPU uses the byte in the index register to determine the conditional address of the operand. (See **12.1.5 Indexed, No Offset**; **12.1.6 Indexed, 8-Bit Offset**; and **12.1.7 Indexed, 16-Bit Offset**.)

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 3-3. Index Register**

The 8-bit index register can also serve as a temporary data storage location.

### 3.1.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset or after the reset stack pointer (RSP) instruction, the stack pointer is preset to $00FF. The address in the stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

| | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 3-4. Stack Pointer**

The ten most significant bits of the stack pointer are permanently fixed at 0000000011, so the stack pointer produces addresses from $00C0 to $00FF. If subroutines and interrupts use more than 64 stack locations, the stack pointer wraps around to address $00FF and begins writing over the previously stored data. A subroutine uses two stack locations; an interrupt uses five locations.

### 3.1.4 Program Counter

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched. The three most significant bits of the program counter are ignored internally and appear as 000.

Normally, the address in the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.

| | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | LOADED WITH VECTOR FROM $1FFE AND $1FFF | | | | | | | | | | | | |

Figure 3-5. Program Counter

### 3.1.5 Condition Code Register

The condition code register is an 8-bit register whose three most significant bits are permanently fixed at 111. The condition code register contains the interrupt mask and four flags that indicate the results of the instruction just executed. The following paragraphs describe the functions of the condition code register.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | H | I | N | Z | C |
| RESET: | 1 | 1 | 1 | U | 1 | U | U | U |

U = UNAFFECTED

Figure 3-6. Condition Code Register

#### 3.1.5.1 Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between bits 3 and 4 of the accumulator during an ADD or ADC operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations.

### 3.1.5.2 Interrupt Mask

Setting the interrupt mask disables interrupts. If an interrupt request occurs while the interrupt mask is logic zero, the CPU saves the CPU registers on the stack, sets the interrupt mask, and then fetches the interrupt vector. If an interrupt request occurs while the interrupt mask is set, the interrupt request is latched. Normally, the CPU processes the latched interrupt as soon as the interrupt mask is cleared again.

A return from interrupt (RTI) instruction pulls the CPU registers from the stack, restoring the interrupt mask to its cleared state. After any reset, the interrupt mask is set and can be cleared only by a software instruction.

### 3.1.5.3 Negative Flag

The CPU sets the negative flag when an arithmetic operation, logical operation, or data manipulation produces a negative result.

### 3.1.5.4 Zero Flag

The CPU sets the zero flag when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

### 3.1.5.5 Carry/Borrow Flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow flag.

## 3.2 Arithmetic/Logic Unit (ALU)

The ALU performs the arithmetic and logical operations defined by the instruction set.

The binary arithmetic circuits decode instructions and set up the ALU for the selected operation. Most binary arithmetic is based on the addition algorithm, carrying out subtraction as negative addition. Multiplication is not performed as a discrete operation but as a chain of addition and shift operations within the ALU. The multiply instruction (MUL) requires 11 internal processor cycles to complete this chain of operations.

# SECTION 4
# INTERRUPTS

This section describes how interrupts temporarily change the normal processing sequence.

## 4.1 Interrupt Sources

The following sources can generate interrupts:

- SWI instruction
- $\overline{\text{IRQ}}$ pin
- Capture/compare timer

An interrupt temporarily stops normal program execution to process a particular event. An interrupt does not stop the operation of the instruction being executed, but takes effect when the current instruction completes its execution. Interrupt processing automatically saves the CPU registers on the stack and loads the program counter with a user-defined interrupt vector address.

### 4.1.1 Software Interrupt

The software interrupt (SWI) instruction causes a nonmaskable interrupt.

### 4.1.2 External Interrupt

An interrupt signal on the $\overline{IRQ}$ pin latches an external interrupt request. When the CPU completes its current instruction, it tests the IRQ latch. If the IRQ latch is set, the CPU then tests the I bit in the condition code register. If the I bit is clear, the CPU then begins the interrupt sequence. The CPU clears the IRQ latch during interrupt processing, so that another interrupt signal on the IRQ pin can latch another interrupt request during the interrupt service routine. As soon as the I bit is cleared during the return from interrupt, the CPU can recognize the new interrupt request. Figure 4-1 shows the $\overline{IRQ}$ pin interrupt logic.

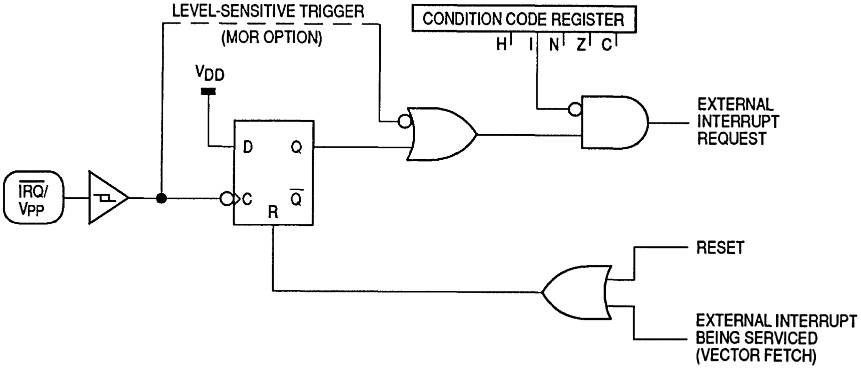


**Figure 4-1. External Interrupt Logic**

Setting the I bit in the condition code register disables external interrupts.

Interrupt triggering sensitivity of the $\overline{IRQ}$ pin is a programmable option. The $\overline{IRQ}$ pin can be negative edge-triggered or negative edge- and low level-triggered. The level-sensitive triggering option allows multiple external interrupt sources to be wired-OR to the $\overline{IRQ}$ pin. An external interrupt request is latched as long as any source is holding the $\overline{IRQ}$ pin low.

### 4.1.3 Capture/Compare Timer Interrupts

The capture/compare timer can generate the following interrupts:

- Input capture interrupt
- Output compare interrupt
- Timer overflow interrupt

Setting the I bit in the condition code register disables timer interrupts.

#### 4.1.3.1 Input Capture Interrupt

An input capture interrupt request occurs if the input capture flag, ICF, becomes set while the input capture interrupt enable bit, ICIE, is also set. ICF is in the timer status register, and ICIE is in the timer control register. (See **SECTION 8 CAPTURE/COMPARE TIMER**.)

#### 4.1.3.2 Output Compare Interrupt

An output compare interrupt request occurs if the output compare flag, OCF, becomes set while the output compare interrupt enable bit, OCIE, is also set. OCF is in the timer status register, and OCIE is in the timer control register. (See **SECTION 8 CAPTURE/COMPARE TIMER**.)

#### 4.1.3.3 Timer Overflow Interrupt

A timer overflow interrupt request occurs if the timer overflow flag, TOF, becomes set while the timer overflow interrupt enable bit, TOIE, is also set. TOF is in the timer status register, and TOIE is in the timer control register. (See **SECTION 8 CAPTURE/COMPARE TIMER**.)

## 4.2 Interrupt Processing

The CPU takes the following actions to begin servicing an interrupt:

- Stores the CPU registers on the stack in the order shown in Figure 4-2
- Sets the I bit in the condition code register to prevent further interrupts
- Loads the program counter with the contents of the appropriate interrupt vector locations:
  - — $1FFC and $1FFD (software interrupt vector)
  - — $1FFA and $1FFB (external interrupt vector)
  - — $1FF8 and $1FF9 (timer interrupt vector)

The return from interrupt (RTI) instruction causes the CPU to recover the CPU registers from the stack as shown in Figure 4-2.

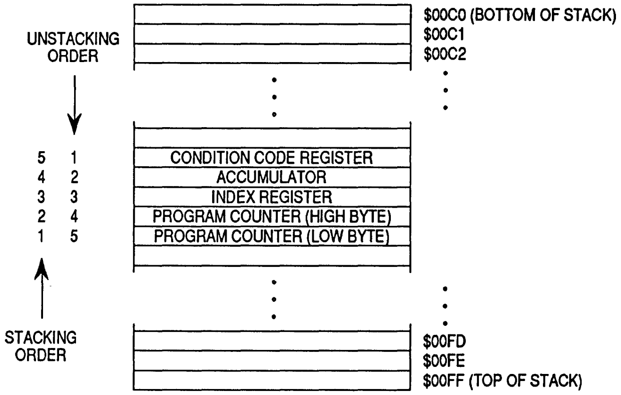


**Figure 4-2. Interrupt Stacking Order**

Table 4-1 summarizes the reset and interrupt sources and vector assignments.

## Table 4-1. Reset/Interrupt Vector Addresses

| Function | Source | Local Mask | Global Mask | Priority (1 = Highest) | Vector Address |
|---|---|---|---|---|---|
| Reset | Power-On<br>RESET Pin<br>COP Watchdog | None | None<br>None<br>None | 1<br>1<br>1 | \$1FFE–\$1FFF |
| Software Interrupt (SWI) | User Code | None | None | Same Priority As Instruction | \$1FFC–\$1FFD |
| External Interrupt | $\overline{\text{IRQ}}$ Pin | None | I Bit | 2 | \$1FFA–\$1FFB |
| Timer Interrupts | ICF Bit<br>OCF Bit<br>TOF Bit | ICIE Bit<br>OCIE Bit<br>TOIE Bit | I Bit | 3 | \$1FF8–\$1FF9 |

NOTE: The COP watchdog is programmable in the mask option register.

Figure 4-3 shows the sequence of events caused by an interrupt.

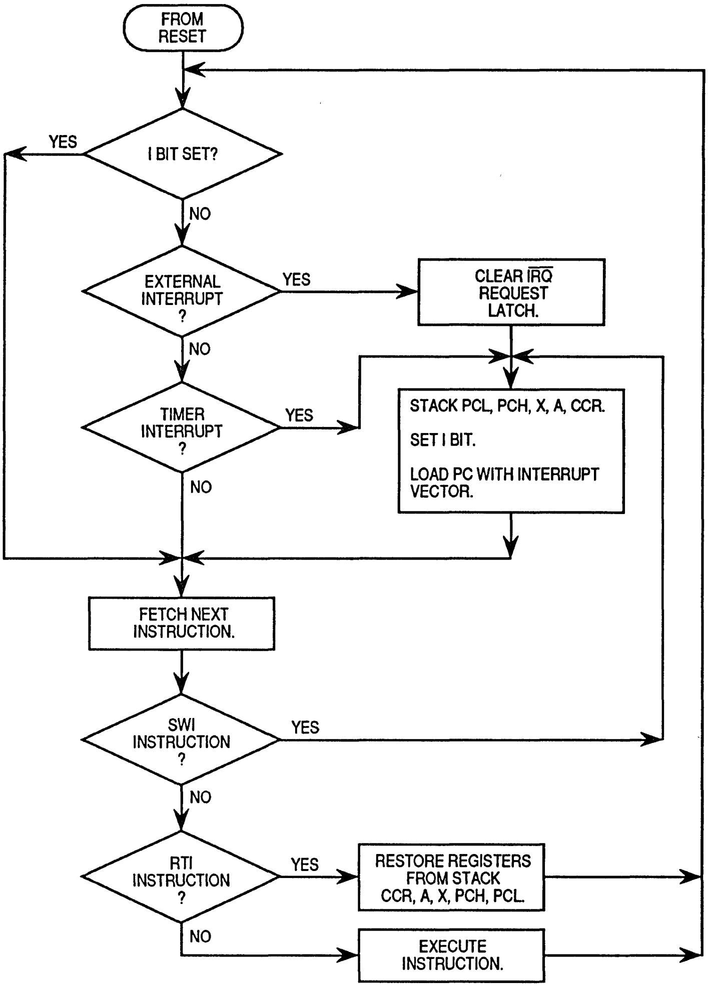


**Figure 4-3. Interrupt Flowchart**

# SECTION 5
# RESETS

This section describes how resets initialize the MCU.

## 5.1 Reset Sources

The following sources can generate resets:

- Power-on reset (POR) circuit
- $\overline{\text{RESET}}$ pin
- COP watchdog

A reset immediately stops the operation of the instruction being executed, initializes certain control bits, and loads the program counter with a user-defined reset vector address. Figure 5-1 is a block diagram of the reset sources.

### 5.1.1 Power-On Reset

A positive transition on the $V_{DD}$ pin generates a power-on reset. The power-on reset is strictly for power-up conditions and cannot be used to detect drops in power supply voltage.

A 4064 $t_{CYC}$ (internal clock cycle) delay after the oscillator becomes active allows the clock generator to stabilize. If the $\overline{\text{RESET}}$ pin is at logic zero at the end of 4064 $t_{CYC}$, the MCU remains in the reset condition until the signal on the $\overline{\text{RESET}}$ pin goes to logic one.

### 5.1.2 External Reset

A logic zero applied to the $\overline{\text{RESET}}$ pin for one and one-half $t_{CYC}$ generates an external reset. A Schmitt trigger senses the logic level at the $\overline{\text{RESET}}$ pin. (See Figure 5-1.)

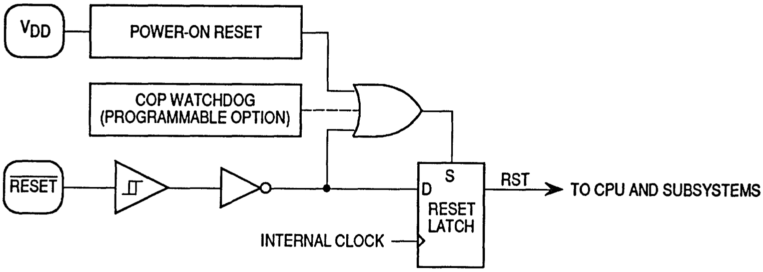


**Figure 5-1. Reset Sources**

### 5.1.3 Computer Operating Properly (COP) Watchdog Reset

A timeout of the COP watchdog generates a COP reset. The COP watchdog is part of a software error detection system and must be cleared periodically to start a new timeout period. To clear the COP watchdog and prevent a COP reset, write a logic zero to bit 0 (COPC) of the COP register at location $1FF0. The COP register, shown in Figure 5-2, is a write-only register that returns the contents of an EPROM location when read.

The COP watchdog function is programmable in the mask option register.

**COPR** — COP Register **$1FF0**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | COPC |
| RESET: | — | — | — | — | — | — | — | 0 |

**Figure 5-2. COP Register (COPR)**

COPC — COP Clear

COPC is a write-only bit. Periodically writing a logic zero to COPC prevents the COP watchdog from resetting the MCU. A reset clears the COPC bit.

## 5.2 Reset States

The following paragraphs describe how resets initialize the MCU.

### 5.2.1 CPU

A reset has the following effects on the CPU:

- Loads the stack pointer with $FF
- Sets the I bit in the condition code register, inhibiting interrupts
- Loads the program counter with the user-defined reset vector from locations $1FFE and $1FFF
- Clears the stop latch, enabling the CPU clock
- Clears the wait latch, waking the CPU from the wait mode

### 5.2.2 I/O Port Registers

A reset has the following effects on I/O port registers:

- Clears ports A, B, C, and D data direction registers so that all I/O port pins are inputs (PD7/TCAP remains an input-only pin, and TCMP remains an output-only pin.)
- Has no effect on port data registers

### 5.2.3 Capture/Compare Timer

A reset has the following effects on the capture/compare timer:

- Loads the timer counter with $FFFC
- Clears the timer control register, except for the IEDG bit, with the following results:

  - — Clears the ICIE bit, inhibiting input capture interrupts
  - — Clears the OCIE bit, inhibiting output compare interrupts
  - — Clears the TOIE bit, inhibiting timer overflow interrupts
  - — Clears OLVL, the output compare bit
  - — Clears the TCMP pin

- Has no effect on the ICF, OCF, and TOF flags in the timer status register

### 5.2.4 Serial I/O Port (SIOP)

A reset clears the SIOP control and status registers and produces the following results:

- Clears the SPE bit, disabling the SIOP
- Clears the MSTR bit, configuring the disabled SIOP for slave mode operation
- Clears the SPIF and DCOL flags

### 5.2.5 COP Watchdog

A reset clears the COP watchdog counter.

### 5.2.6 Analog-to-Digital Converter (ADC)

A reset clears the ADC status and control register, with the following results:

- Clears the ADON bit, disabling the ADC
- Clears the CCF flag
- Clears the ADRC bit, configuring the disabled ADC for operation at internal clock frequency
- Clears bits CH2–CH0, selecting channel 0 as the analog input

# SECTION 6
# LOW POWER MODES

This section describes the three low-power modes:

- Stop mode
- Wait mode
- Data-retention mode

## 6.1 Stop Mode

The STOP instruction puts the MCU in its lowest power-consumption mode and has the following effects on the MCU:

- Stops the internal oscillator, the CPU clock, and the internal clock, turning off the capture/compare timer, the COP watchdog, the SIOP, and the ADC.
- Clears the I bit in the condition code register, enabling external interrupts
- Clears the ICF, OCF, and TOF interrupt flags in the timer status register, removing any pending timer interrupts
- Clears the ICIE, OCIE, and TOIE bits in the timer control register, disabling further timer interrupts

The STOP instruction does not affect any other registers or any I/O lines.

The following events bring the MCU out of stop mode:

- An external interrupt signal on the $\overline{\text{IRQ}}$ pin — A high-to-low transition on the $\overline{\text{IRQ}}$ pin loads the program counter with the contents of locations \$1FFA and \$1FFB. The timer resumes counting from the last value before the STOP instruction.

- External reset — A logic zero on the $\overline{\text{RESET}}$ pin resets the MCU and loads the program counter with the contents of locations \$1FFE and \$1FFF. The timer begins counting from \$FFFC.

When the MCU exits stop mode, processing resumes after a stabilization delay of 4064 oscillator cycles.

An active edge on the PD7/TCAP pin during stop mode sets the ICF flag when an external interrupt brings the MCU out of stop mode. An external interrupt also latches the value in the timer registers into the input capture registers.

If a reset brings the MCU out of stop mode, then an active edge on the PD7/TCAP pin during stop mode has no effect on the ICF flag or the input capture registers.

Figure 6-1 shows the sequence of events caused by the STOP instruction.

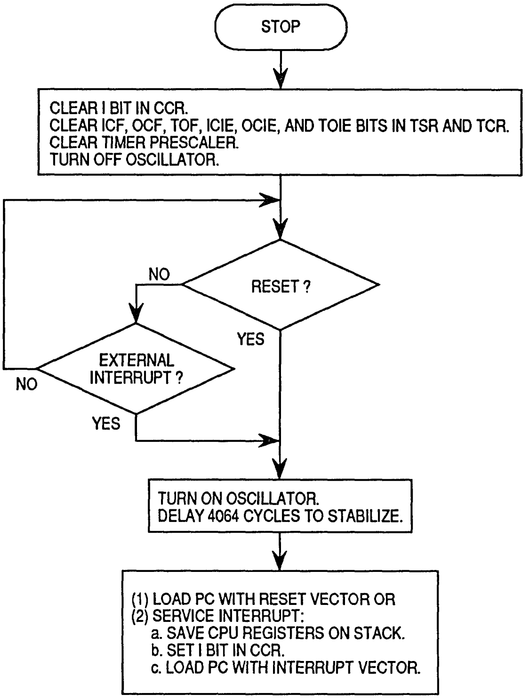

STOP
CLEAR I BIT IN CCR.
CLEAR ICF, OCF, TOF, ICIE, OCIE, AND TOIE BITS IN TSR AND TCR.
CLEAR TIMER PRESCALER.
TURN OFF OSCILLATOR.
RESET ?
NO
YES
EXTERNAL INTERRUPT ?
NO
YES
TURN ON OSCILLATOR.
DELAY 4064 CYCLES TO STABILIZE.
(1) LOAD PC WITH RESET VECTOR OR
(2) SERVICE INTERRUPT:
a. SAVE CPU REGISTERS ON STACK.
b. SET I BIT IN CCR.
c. LOAD PC WITH INTERRUPT VECTOR.

**Figure 6-1. STOP Instruction Flowchart**

## 6.2 Wait Mode

The WAIT instruction puts the MCU in an intermediate power-consumption mode and has the following effects on the MCU:

- Clears the I bit in the condition code register, enabling interrupts
- Stops the CPU clock, but allows the internal clock to drive the capture/compare timer, the COP watchdog, and the ADC

The WAIT instruction does not affect any other registers or any I/O lines.

The following conditions restart the CPU clock and bring the MCU out of wait mode:

- External interrupt — A high-to-low transition on the $\overline{\text{IRQ}}$ pin loads the program counter with the contents of locations $1FFA and $1FFB.

- Timer interrupt — Input capture, output compare, and timer overflow interrupt requests load the program counter with the contents of locations $1FF8 and $1FF9.

- COP watchdog reset — A timeout of the COP watchdog resets the MCU and loads the program counter with the contents of locations $1FFE and $1FFF. Software can enable timer interrupts so that the MCU can periodically exit wait mode to reset the COP watchdog.

- External reset — A logic zero on the $\overline{\text{RESET}}$ pin resets the MCU and loads the program counter with the contents of locations $1FFE and $1FFF.

Figure 6-2 shows the sequence of events caused by the WAIT instruction.

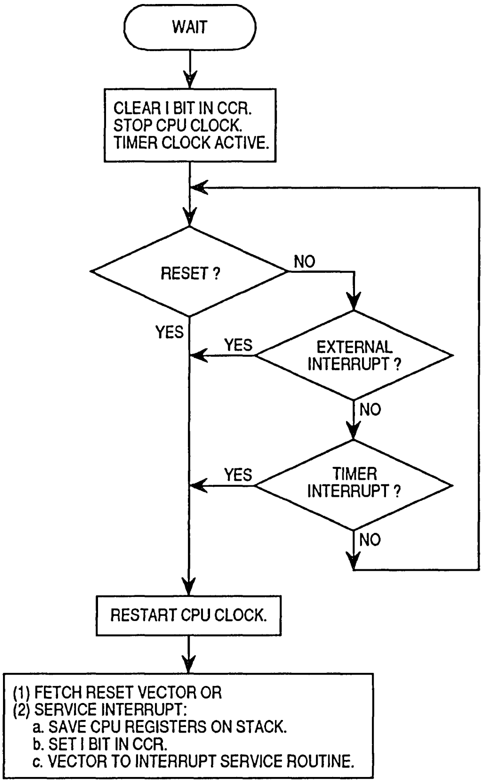


Figure 6-2. WAIT Instruction Flowchart

The following figure shows the effect of the STOP and WAIT instructions on the CPU clock and the timer clock.

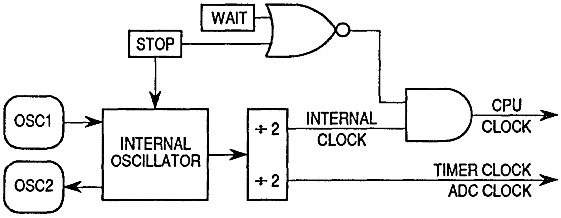


**Figure 6-3. STOP/WAIT Clock Logic**

## 6.3 Data-Retention Mode

In data-retention mode, the MCU retains RAM contents and CPU register contents at $V_{DD}$ voltages as low as 2.0 Vdc. The data-retention feature allows the MCU to remain in a low power-consumption state during which it retains data, but the CPU cannot execute instructions. To put the MCU in data-retention mode:

1. Drive the $\overline{\text{RESET}}$ pin to logic zero.
2. Lower the $V_{DD}$ voltage. The $\overline{\text{RESET}}$ pin must remain low continuously during data-retention mode.

To take the MCU out of data-retention mode:

1. Return $V_{DD}$ to normal operating voltage.
2. Return the $\overline{\text{RESET}}$ pin to logic one.

# SECTION 7
# PARALLEL I/O

This section describes the four parallel I/O ports.

## 7.1 I/O Port Function

The 20 bidirectional I/O pins and one input-only pin form four parallel I/O ports. Each I/O pin is programmable as an input or an output in the four data direction registers. Reset configures all I/O pins as inputs.

**NOTE**

Connect any unused inputs and I/O pins to an appropriate logic level, either $V_{DD}$ or $V_{SS}$. Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.

## 7.2 Port A

Port A is an 8-bit general-purpose bidirectional I/O port.

### 7.2.1 Port A Data Register (PORTA)

The port A data register, shown in Figure 7-1, contains a data latch for each of the eight port A pins.

**PORTA** — Port A Data Register **$0000**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 7-1. Port A Data Register (PORTA)**

PA7–PA0 — Port A Data Bits

These read/write bits are software-programmable. Data direction of each bit is under the control of the corresponding bit in data direction register A.

### 7.2.2 Data Direction Register A (DDRA)

Data direction register A, shown in Figure 7-2, determines whether each port A pin is an input or an output. Writing a logic one to a DDRA bit enables the output buffer for the corresponding port A pin; a logic zero disables the output buffer. A reset clears all DDRA bits, configuring all port A pins as inputs.

**DDRA** — Data Direction Register A **$0004**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-2. Data Direction Register A (DDRA)**

DDRA7–DDRA0 — Port A Data Direction Bits

These read/write bits control port A data direction. A reset clears all DDRA bits.

1 = Corresponding port A pin configured as output

0 = Corresponding port A pin configured as input

### NOTE

Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.

Figure 7-3 shows the port A I/O logic.

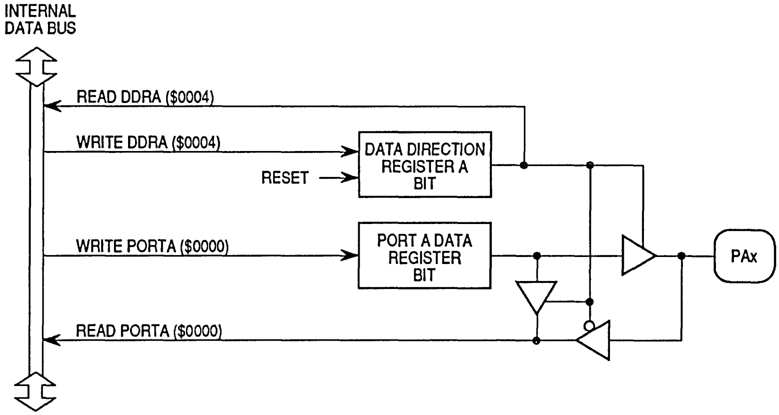


**Figure 7-3. Port A I/O Circuit**

When a port A pin is programmed as an output, reading the port bit reads the value of the data latch and not the voltage on the pin. When a port A pin is programmed as an input, reading the port bit reads the voltage level on the pin. The data latch can always be written, regardless of the state of its DDRA bit. Table 7-1 summarizes the operation of the port A pins.

**Table 7-1. Port A Pin Functions**

| DDRA Bit | PORTA Bit | I/O Pin Mode | Accesses to DDRA | Accesses to PORTA | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, hi-Z | DDRA7–0 | Pin | NOTE 2 |
| 1 | X | Output | DDRA7–0 | PA7–0 | PA7–0 |

NOTES:
1. X = don't care.
2. Writing affects data register, but does not affect input.

## 7.3 Port B

Port B is a 3-bit general-purpose bidirectional I/O port that shares its pins with the serial I/O port (SIOP).

When the SIOP is enabled, PB6/SDI is the SIOP data input pin, and PB5/SDO is the SIOP data output pin. With the SIOP in master mode, the PB7/SCK pin is the SIOP clock output. In slave mode, PB7/SCK is the SIOP clock input.

### 7.3.1 Port B Data Register (PORTB)

The port B data register, shown in Figure 7-4, contains a data latch for each of the three port B pins.

**PORTB** — Port B Data Register **$0001**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | PB7 | PB6 | PB5 | 0 | 0 | 0 | 0 | 0 |
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 7-4. Port B Data Register (PORTB)**

PB7–PB5 — Port B Data Bits

These read/write bits are software-programmable. Data direction of each bit is under the control of the corresponding bit in data direction register B.

Bits 4–0 — Not used

Bits 4–0 always read as logic zeros. Writes to these bits have no effect.

### 7.3.2 Data Direction Register B (DDRB)

Data direction register B determines whether pins PB7/SCK, PB6/SDI, and PB5/SDO are inputs or outputs. (See Figure 7-5.) Writing a logic one to a DDRB bit enables the output buffer for the corresponding port B pin; a logic zero disables the output buffer. A reset clears bits DDRB7–DDRB5, configuring all three port B pins as inputs.

**DDRB** — Data Direction Register B **$0005**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | DDRB7 | DDRB6 | DDRB5 | 0 | 0 | 0 | 0 | 0 |
| RESET: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 7-5. Data Direction Register B (DDRB)**

DDRB7–DDRB5 — Port B Data Direction Bits

These read/write bits control port B data direction.

1 = Corresponding port B pin configured as output

0 = Corresponding port B pin configured as input

Bits 4–0 — Not used

Bits 4–0 always read as logic ones.

**NOTE**

Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1.

Figure 7-6 shows the port B I/O logic.

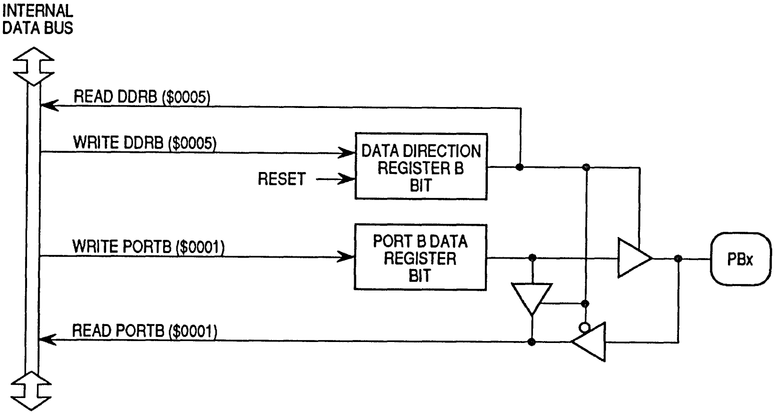


**Figure 7-6. Port B I/O Circuit**

When a port B pin is programmed as an output, reading the port bit reads the value of the data latch and not the voltage on the pin. When a port B pin is programmed as an input, reading the port bit reads the voltage level on the pin. The data latch can always be written, regardless of the state of its DDRB bit. Table 7-2 summarizes the operation of the port B pins.

**Table 7-2. Port B Pin Functions**

| DDRB Bit | PORTB Bit | I/O Pin Mode | Accesses to DDRB | Accesses to PORTB | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, hi-Z | DDRB7–5 | Pin | NOTE 2 |
| 1 | X | Output | DDRB7–5 | PB7–5 | PB7–5 |

NOTES:
1. X = don't care.
2. Writing affects data register, but does not affect input.

## 7.4 Port C

Port C is an 8-bit general-purpose bidirectional I/O port that shares five of its pins with the A/D converter (ADC).

When the ADC is turned on, the PC7/$V_{RH}$ pin is the positive reference voltage pin for the ADC. Software selects one of pins PC6/AN0, PC5/AN1, PC4/AN2, or PC3/AN3 as an analog input. The port C data register reads normally while the ADC is on, except that the bit corresponding to the currently selected ADC input pin reads as logic zero.

### 7.4.1 Port C Data Register (PORTC)

The port C data register, shown in Figure 7-7, contains a data latch for each of the eight port C pins.

**PORTC** — Port C Data Register **$0002**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
|  | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| RESET: | UNAFFECTED BY RESET |  |  |  |  |  |  |  |

**Figure 7-7. Port C Data Register (PORTC)**

PC7–PC0 — Port C Data Bits

These read/write bits are software-programmable. Data direction of each bit is under the control of the corresponding bit in data direction register C.

### 7.4.2 Data Direction Register C (DDRC)

Data direction register C determines whether each port C pin is an input or an output. (See Figure 7-8.) Writing a logic one to a DDRC bit enables the output buffer for the corresponding port C pin; a logic zero disables the output buffer. A reset clears all DDRC bits, configuring all port C pins as inputs.

**DDRC** — Data Direction Register C **$0006**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-8. Data Direction Register C (DDRC)**

DDRC7–DDRC0 — Port C Data Direction Bits

These read/write bits control port C data direction.

1 = Corresponding port C pin configured as output

0 = Corresponding port C pin configured as input

**NOTE**

Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.

Figure 7-9 shows the port C I/O logic.

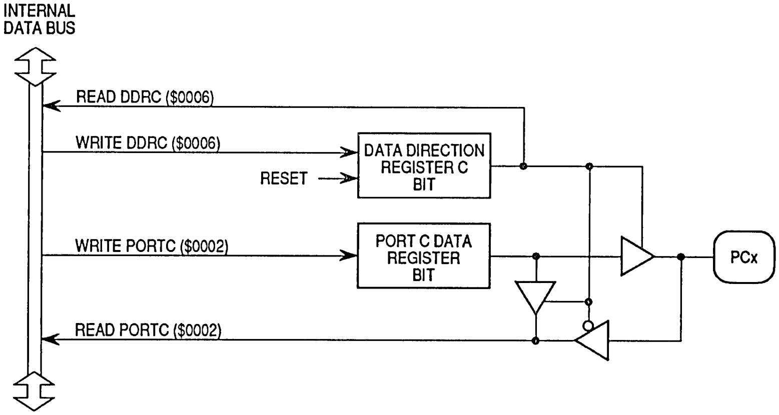


**Figure 7-9. Port C I/O Circuit**

When a port C pin is programmed as an output, reading the port bit reads the value of the data latch and not the voltage on the pin. When a port C pin is programmed as an input, reading the port bit reads the voltage level on the pin. The data latch can always be written, regardless of the state of its DDRC bit. Table 7-3 summarizes the operation of the port C pins.

**Table 7-3. Port C Pin Functions**

| DDRC Bit | PORTC Bit | I/O Pin Mode | Accesses to DDRC | Accesses to PORTC | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, hi-Z | DDRC7–0 | Pin | NOTE 2 |
| 1 | X | Output | DDRC7–0 | PC7–0 | PC7–0 |

NOTES:
1. X = don't care.
2. Writing affects data register, but does not affect input.

## 7.5 Port D

Port D is a 2-bit port with one general-purpose bidirectional pin and one general-purpose input-only pin. Port D shares the input-only pin, PD7/TCAP, with the capture/compare timer. The timer uses PD7/TCAP as the input capture pin. The PD7/TCAP pin can always serve as a general-purpose input pin, even if input capture interrupts are enabled.

### 7.5.1 Port D Data Register (PORTD)

The port D data register, shown in Figure 7-10, contains data latches for the two port D pins.

**PORTD** — Port D Data Register **$0003**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | PD7 | 0 | PD5 | 1 | 0 | 0 | 0 | 0 |
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 7-10. Port D Data Register (PORTD)**

PD7 — Port D Data Bit 7

PD7 is a read-only bit. Pin PD7/TCAP is an input-only pin.

PD5 — Port D Data Bit 5

This read/write bit is software-programmable. Data direction is under the control of the DDRD5 bit in data direction register D.

### 7.5.2 Data Direction Register D (DDRD)

Data direction register D determines whether pin PD5 is an input or an output. (See Figure 7-11.) Writing a logic one to DDRD5 enables the output buffer for the PD5 pin; a logic zero disables the output buffer. A reset initializes bit DDRD5 to logic zero, configuring pin PD5 as an input. DDR7 is fixed at logic zero, configuring pin PD7/TCAP as input-only.

**DDRD** — Data Direction Register D **$0007**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
|  | 0 | 0 | DDRD5 | 0 | 0 | 0 | 0 | 0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-11. Data Direction Register D (DDRD)**

DDRD5 — Port D Data Direction Bit 5

This read/write bit controls the data direction of pin PD5.

1 = PD5 pin configured as output

0 = PD5 pin configured as input

**NOTE**

Avoid glitches on pin PD5 by writing to bit PD5 in the port D data register before changing bit DDRD5 from 0 to 1.

Figure 7-12 shows the port D I/O logic.

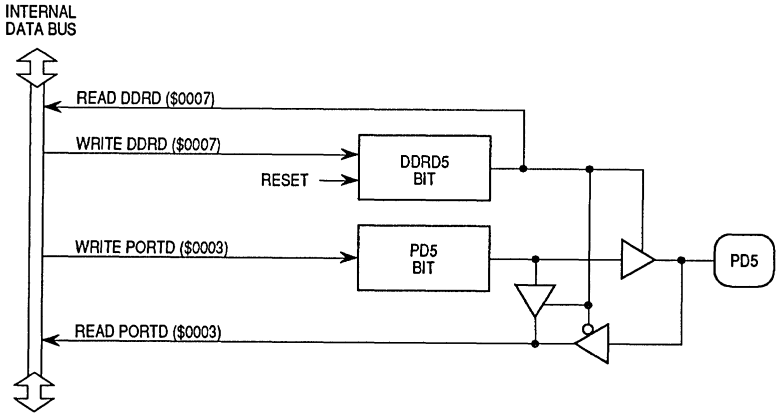


**Figure 7-12. Port D I/O Circuit**

When the PD5 pin is programmed as an output, reading the port bit reads the value of the PD5 data latch and not the voltage on the pin. When the PD5 pin is programmed as an input, reading bit PD5 reads the voltage level on pin PD5. The PD5 data latch can always be written regardless of the state of the DDRD5 bit. PD7/TCAP is an input-only pin, and reading bit PD7 reads the voltage level on the pin. Table 7-4 summarizes the operation of pin PD5.

**Table 7-4. PD5 Pin Functions**

| DDRD5 Bit | PD5 Bit | I/O Pin Mode | Accesses to DDRD | Accesses to PORTD | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, hi-Z | DDRD5 | Pin | NOTE 2 |
| 1 | X | Output | DDRD5 | PD5 | PD5 |

NOTES:
1. X = don't care.
2. Writing affects data register, but does not affect input.

# SECTION 8
# CAPTURE/COMPARE TIMER

This section describes the operation of the 16-bit capture/compare timer. Figure 8-1 shows the organization of the capture/compare timer subsystem.

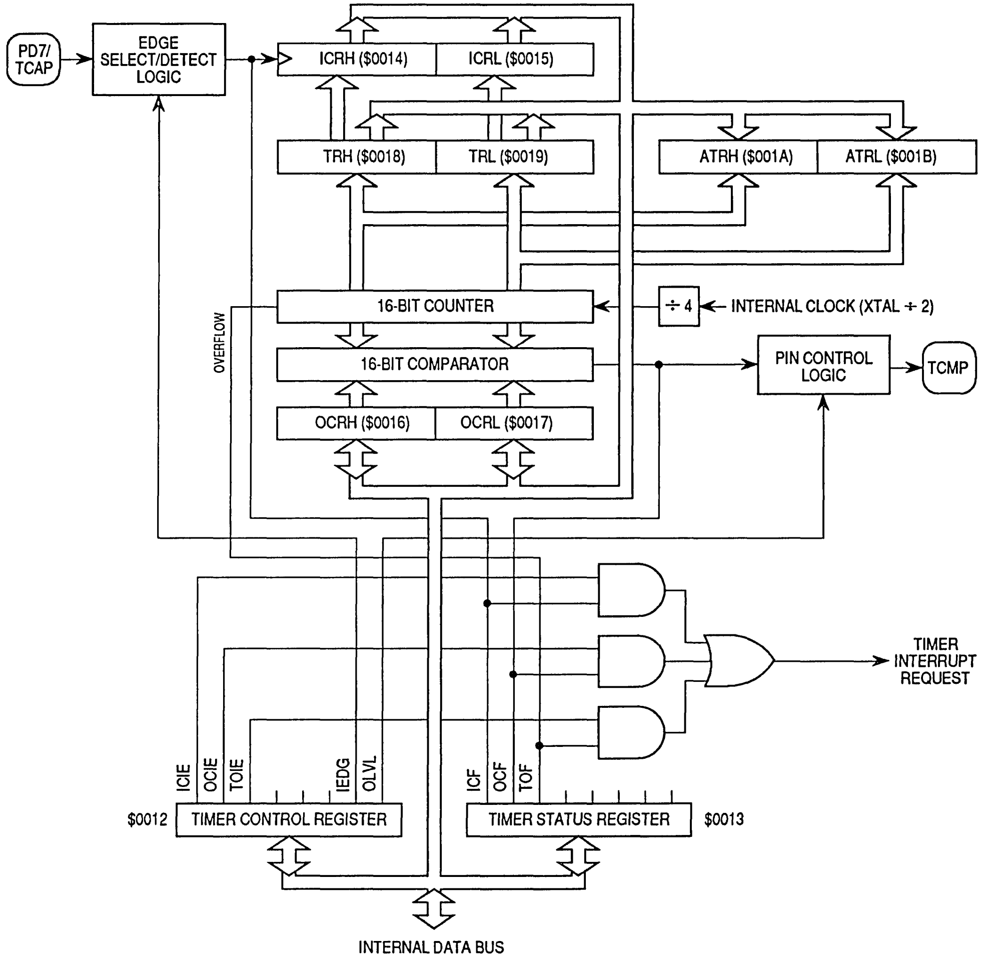


**Figure 8-1. Timer Block Diagram**

## 8.1 Timer Operation

The core of the capture/compare timer is a 16-bit free-running counter. The counter provides the timing reference for the input capture and output compare functions. The input capture and output compare functions provide a means to latch the times at which external events occur, to measure input waveforms, and to generate output waveforms and timing delays. Software can read the value in the 16-bit free-running counter at any time without affecting the counter sequence.

Because of the 16-bit timer architecture, the I/O registers for the input capture and output compare functions are pairs of 8-bit registers.

Because the counter is 16 bits long and preceded by a fixed divide-by-four prescaler, the counter rolls over every 262,144 internal clock cycles. Timer resolution with a 4-MHz crystal is 2 µs.

### 8.1.1 Input Capture

The input capture function is a means to record the time at which an external event occurs. When the input capture circuitry detects an active edge on the PD7/TCAP pin, it latches the contents of the timer registers into the input capture registers. The polarity of the active edge is programmable.

Latching values into the input capture registers at successive edges of the same polarity measures the period of the input signal on the PD7/TCAP pin. Latching the counter values at successive edges of opposite polarity measures the pulse width of the signal. Figure 8-2 shows the logic of the input capture function.

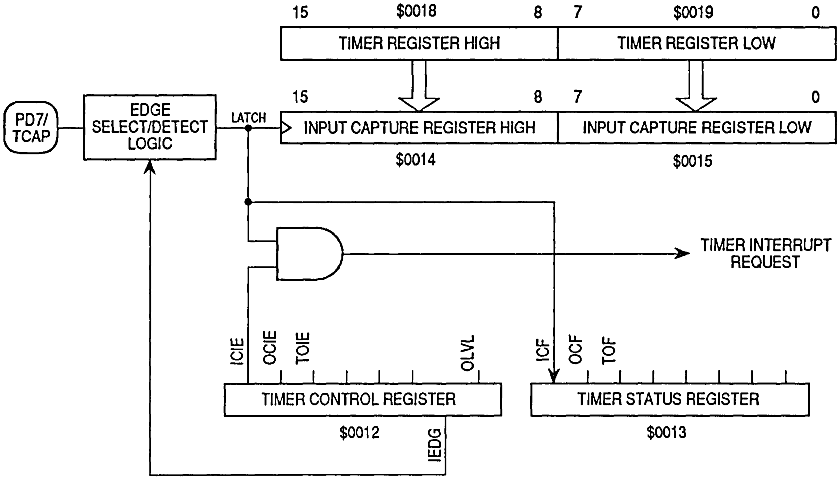


**Figure 8-2. Input Capture Operation**

### 8.1.2 Output Compare

The output compare function is a means of generating an output signal when the 16-bit counter reaches a selected value. Software writes the selected value into the output compare registers. On every fourth internal clock cycle the output compare circuitry compares the value of the counter to the value written in the output compare registers. When a match occurs, the timer transfers the programmable output level bit (OLVL) from the timer control register to the TCMP pin.

The programmer can use the output compare register to measure time periods, to generate timing delays, or to generate a pulse of specific duration or a pulse train of specific frequency and duty cycle on the TCMP pin. Figure 8-3 shows the logic of the output compare function.

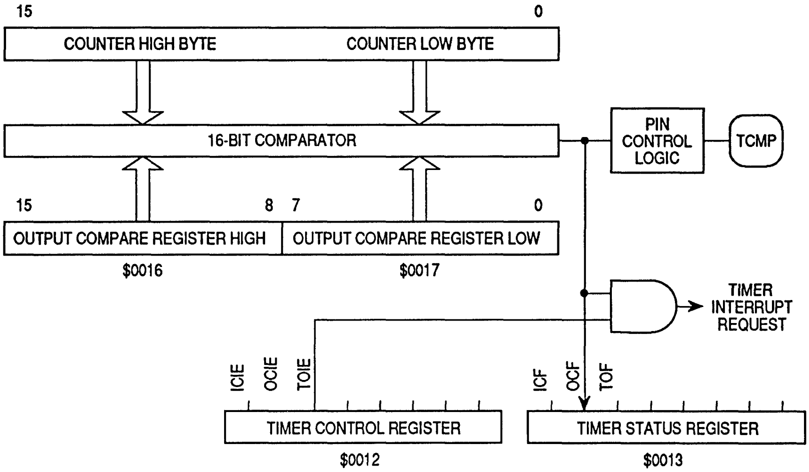


**Figure 8-3. Output Compare Operation**

## 8.2 Timer I/O Registers

The following registers control and monitor the operation of the timer:

- Timer control register (TCR)
- Timer status register (TSR)
- Timer registers (TRH and TRL)
- Alternate timer registers (ATRH and ATRL)
- Input capture registers (ICRH and ICRL)
- Output compare registers (OCRH and OCRL)

### 8.2.1 Timer Control Register (TCR)

The timer control register, shown in Figure 8-4, performs the following functions:

- Enables input capture interrupts
- Enables output compare interrupts
- Enables timer overflow interrupts
- Controls the active edge polarity of the TCAP signal
- Controls the active level of the TCMP output

**TCR** — Timer Control Register **$0012**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | ICIE | OCIE | TOIE | 0 | 0 | 0 | IEDG | OLVL |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | U | 0 |

U = UNAFFECTED

**Figure 8-4. Timer Control Register (TCR)**

ICIE — Input Capture Interrupt Enable

This read/write bit enables interrupts caused by active signal on the TCAP pin. Resets clear the ICIE bit.

1 = Input capture interrupts enabled
0 = Input capture interrupts disabled

OCIE — Output Compare Interrupt Enable

This read/write bit enables interrupts caused by an active signal on the TCMP pin. Resets clear the OCIE bit.

1 = Output compare interrupts enabled
0 = Output compare interrupts disabled

TOIE — Timer Overflow Interrupt Enable

This read/write bit enables interrupts caused by a timer overflow. Resets clear the TOIE bit.

1 = Timer overflow interrupts enabled
0 = Timer overflow interrupts disabled

IEDG — Input Edge

The state of this read/write bit determines whether a positive or negative transition on the TCAP pin triggers a transfer of the contents of the timer register to the input capture register. Resets have no effect on the IEDG bit.

1 = Positive edge (low to high transition) triggers input capture
0 = Negative edge (high to low transition) triggers input capture

OLVL — Output Level

The state of this read/write bit determines whether a logic one or a logic zero appears on the TCMP pin when a successful output compare occurs. Resets clear the OLVL bit.

1 = TCMP goes high on output compare
0 = TCMP goes low on output compare

### 8.2.2 Timer Status Register (TSR)

The timer status register, shown in Figure 8-5, contains flags for the following events:

- An active signal on the TCAP pin, transferring the contents of the timer registers to the input capture registers
- A match between the 16-bit counter and the output compare registers, transferring the OLVL bit to the TCMP pin
- A timer rollover from $FFFF to $0000

**TSR** — Timer Status Register **$0012**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | ICF | OCF | TOF | 0 | 0 | 0 | 0 | 0 |
| RESET: | U | U | U | 0 | 0 | 0 | 0 | 0 |

U = UNAFFECTED

**Figure 8-5. Timer Status Register (TSR)**

ICF — Input Capture Flag

The ICF bit is automatically set when an edge of the selected polarity occurs on the TCAP pin. Clear the ICF bit by reading the timer status register with ICF set, and then reading the low byte ($0015) of the input capture registers. Resets have no effect on ICF.

OCF — Output Compare Flag

The OCF bit is automatically set when the value of the timer registers matches the contents of the output compare registers. Clear the OCF bit by reading the timer status register with OCF set, and then accessing the low byte ($0017) of the output compare registers. Resets have no effect on OCF.

TOF — Timer Overflow Flag

The TOF bit is automatically set when the 16-bit counter rolls over from $FFFF to $0000. Clear the TOF bit by reading the timer status register with TOF set, and then accessing the low byte ($0019) of the timer registers. Resets have no effect on TOF.

### 8.2.3 Timer Registers (TRH and TRL)

The timer registers, shown in Figure 8-6, contain the current high and low bytes of the 16-bit counter. Reading TRH before reading TRL causes TRL to be latched until TRL is read. Reading TRL after reading the timer status register clears the timer overflow flag (TOF). Writing to the timer registers has no effect.

**TRH and TRL** — Timer Registers High/Low **$0018 and $0019**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $0018 | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0019 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

Reset initializes the timer registers to $FFFC.

**Figure 8-6. Timer Registers (TRH and TRL)**

Reading TRH returns the current value of the high byte of the counter and causes the low byte to be latched into a buffer, as shown in Figure 8-7. The buffer value remains fixed even if the high byte is read more than once. Reading TRL reads the transparent low byte buffer and completes the read sequence of the timer registers.

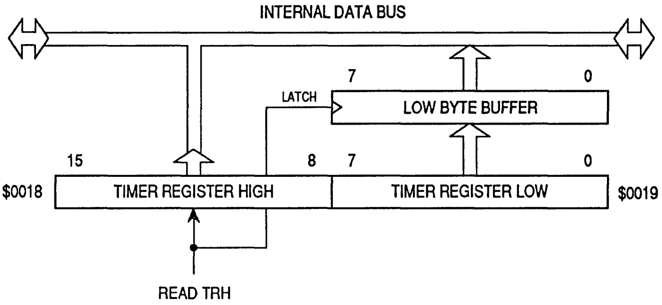


**Figure 8-7. Timer Register Reads**

**NOTE**

To prevent interrupts from occurring between readings of TRH and TRL, set the interrupt flag in the condition code register before reading TRH, and clear the flag after reading TRL.

### 8.2.4 Alternate Timer Registers (ATRH and ATRL)

The alternate timer registers, shown in Figure 8-8, contain the current high and low bytes of the 16-bit counter. Reading ATRH before reading ATRL causes ATRL to be latched until ATRL is read. Reading does not affect the timer overflow flag (TOF). Writing to the alternate timer registers has no effect.

**ATRH and ATRL** — Alternate Timer Registers High/Low **$001A and $001B**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $001A | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $001B | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

Reset initializes the alternate timer registers to $FFFC.

**Figure 8-8. AlternateTimer Registers (ATRH and ATRL)**

Reading ATRH returns the current value of the high byte of the counter and causes the low byte to be latched into a buffer, as shown in Figure 8-9. The buffer value remains fixed even if the high byte is read more than once. Reading ATRL reads the transparent low byte buffer and completes the read sequence of the alternate timer registers.

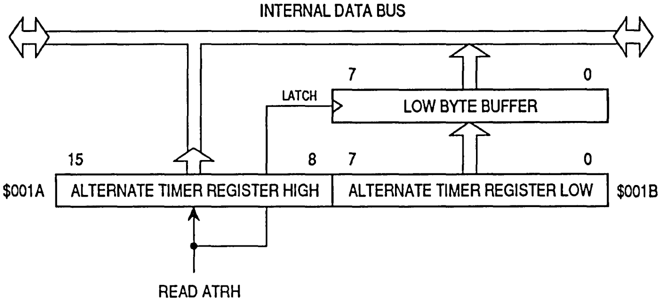


Figure 8-9. Alternate Timer Register Reads

NOTE

To prevent interrupts from occurring between readings of ATRH and ATRL, set the interrupt flag in the condition code register before reading ATRH, and clear the flag after reading ATRL.

### 8.2.5 Input Capture Registers (ICRH and ICRL)

When a selected edge occurs on the TCAP pin, the current high and low bytes of the 16-bit counter are latched into the input capture registers. Reading ICRH before reading ICRL inhibits further captures until ICRL is read. Reading ICRL after reading the timer status register clears the input capture flag (ICF). Writing to the input capture registers has no effect.

**ICRH and ICRL** — Input Capture Registers High/Low **$0014 and $0015**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $0014 | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0015 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

Reset does not affect the input capture registers.

Figure 8-10. Input Capture Registers (ICRH/ICRL)

**NOTE**

To prevent interrupts from occurring between readings of ICRH and ICRL, set the interrupt flag in the condition code register before reading ICRH, and clear the flag after reading ICRL.

### 8.2.6 Output Compare Registers (OCRH and OCRL)

When the value of the 16-bit counter matches the value in the output compare registers, the planned TCMP pin action takes place. Writing to OCRH before writing to OCRL inhibits timer compares until OCRL is written. Reading or writing to OCRL after reading the timer status register clears the output compare flag (OCF).

**OCRH and OCRL** — Output Compare Registers High/Low **$0016 and $0017**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $0016 | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0017 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

Reset does not affect the output compare registers.

**Figure 8-11. Output Compare Registers (OCRH and OCRL)**

To prevent OCF from being set between the time it is read and the time the output compare registers are updated, use the following procedure:

1. Disable interrupts by setting the I bit in the condition code register.
2. Write to OCRH. Compares are now inhibited until OCRL is written.
3. Clear bit OCF by reading the timer status register (TSR).
4. Enable the output compare function by writing to OCRL.
5. Enable interrupts by clearing the I bit in the condition code register.

# SECTION 9
# SERIAL I/O PORT (SIOP)

This section describes the operation of the serial I/O port (SIOP). Figure 9-1 shows the structure of the SIOP subsystem.

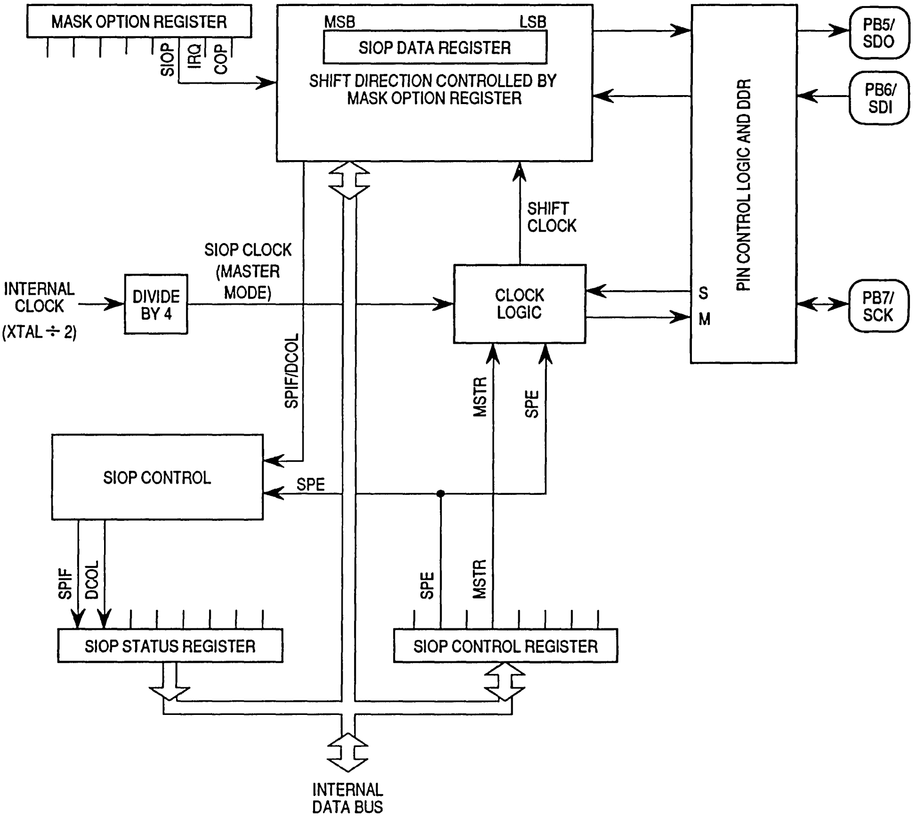


**Figure 9-1. SIOP Block Diagram**

## 9.1 SIOP Operation

The SIOP enables high-speed synchronous serial data transfer between the MCU and peripheral devices. Shift registers used with the SIOP can increase the number of parallel I/O pins controlled by the MCU. More powerful peripherals such as analog-to-digital converters and real-time clocks are also compatible with the SIOP. A programmable option allows the SIOP to transfer data MSB first or LSB first.

### 9.1.1 SIOP Pin Functions

The SIOP uses the three port B I/O pins. Setting the SPE bit in the SIOP control register enables the SIOP. When the SPE bit is set, the PB7/SCK, PB6/SDI, and PB5/SDO pins are dedicated to SIOP functions. When the SPE bit is clear, the PB7/SCK, PB6/SDI, and PB5/SDO pins are bidirectional port B I/O pins.

Setting the MSTR bit in the SIOP control register configures the SIOP for master mode. In master mode, the PB7/SCK pin is the SIOP clock output. PB6/SDI is the serial data input, and PB5/SDO is the serial data output. The master MCU initiates and controls the transfer of data to and from one or more slave peripheral devices. In master mode, a transmission is initiated by writing to the SIOP data register. Data written to the SIOP data register is parallel-loaded and shifted out serially to the slave device(s).

### 9.1.2 SIOP Clock

The PB7/SCK pin synchronizes the movement of data into and out of the MCU through the PB6/SDI and PB5/SDO pins. In master mode, the PB7/SCK pin is an output. The transmission rate for master mode is one-fourth of the internal clock rate.

In slave mode, the PB7/SCK pin is an input. The maximum SIOP clock rate for slave mode is the maximum internal clock rate divided by four. There is no minimum SIOP clock frequency for slave mode.

Figure 9-2 shows the timing relationships between the SIOP clock, data input, and data output. The state of the SIOP clock between transmissions is a logic one. The first falling edge on the PB7/SCK pin signals the beginning of a transmission, and data appears at the PB5/SDO pin. Data is captured at the PB6/SDI pin on the rising edge of the SIOP clock, and the transmission ends on the eighth rising edge of the SIOP clock.

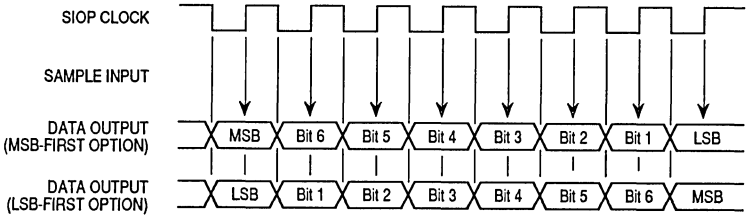


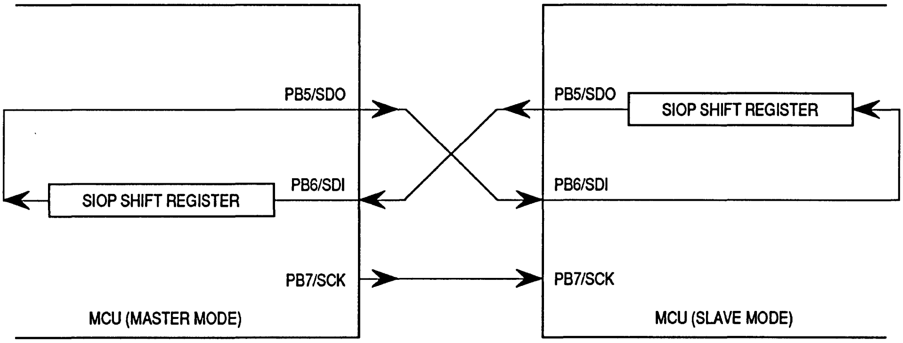
**Figure 9-2. SIOP Data/Clock Timing**

Valid SDI data must be present for an SDI setup time before the rising edge of the SIOP clock and must remain valid for an SDI hold time after the rising edge of the SIOP clock. (See **13.9 SIOP Timing ($V_{DD}$ = 5.0 Vdc)** and **13.10 SIOP Timing ($V_{DD}$ = 3.3 Vdc)**.)

Between transfers, the state of the PB5/SDO pin reflects the value of the last bit shifted out on the previous transmission. On the first falling edge on the PB7/SCK pin, the first data bit to be shifted out appears at the PB5/SDO pin.

### 9.1.3 Data Movement

Connecting the SIOP data register of a master MCU with the SIOP of a slave MCU forms a 16-bit circular shift register. During an SIOP transfer, the master shifts out the contents of its SIOP data register on its PB5/SDO pin. At the same time, the slave MCU shifts out the contents of its SIOP data register on its PB5/SDO pin. Figure 9-3 shows how the master and slave exchange the contents of their data registers.



NOTE: Both MCUs shown are programmed for MSB-first data format.

**Figure 9-3. Master/Slave SIOP Shift Register Operation**

## 9.2 SIOP I/O Registers

The following registers control and monitor SIOP operation:

- SIOP control register (SCR)
- SIOP status register (SSR)
- SIOP data register (SDR)

### 9.2.1 SIOP Control Register (SCR)

The read/write SIOP control register, shown in Figure 9-4, contains two bits. One bit enables the SIOP, and the other configures the SIOP for master mode or for slave mode.

**SCR** — SIOP Control Register **$000A**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
|  | 0 | SPE | 0 | MSTR | 0 | 0 | 0 | 0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-4. SIOP Control Register (SCR)**

SPE — SIOP Enable

This read/write bit enables the SIOP. Setting the SPE bit enables the SIOP. Clearing the SPE bit returns port B to its normal I/O function. After clearing the SPE bit, be sure to initialize the port B data direction register for the intended port B I/O use. Resets clear SPE.

1 = SIOP enabled
0 = SIOP disabled

MSTR — Master Mode Select

This read/write bit configures the SIOP for master mode. Setting MSTR initializes the PB7/SCK pin as the SIOP clock output. Clearing MSTR initializes the PB7/SCK pin as the SIOP clock input. Resets clear MSTR.

1 = Master mode selected
0 = Slave mode selected

### 9.2.2 SIOP Status Register (SSR)

The read-only SIOP status register, shown in Figure 9-5, contains two bits. One bit indicates that a SIOP transfer is complete, and the other indicates that an invalid access of the SIOP data register occurred while a transfer was in progress.

**SSR** — SIOP Status Register **$000B**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
|  | SPIF | DCOL | 0 | 0 | 0 | 0 | 0 | 0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-5. SIOP Status Register (SCR)**

SPIF — SIOP Interrupt Flag

This clearable, read-only bit is set automatically at the end of a serial transfer. Clear the SPIF bit by reading the SIOP status register with SPIF set, and then reading or writing the SIOP data register. Resets clear the SPIF bit.

1 = Serial transfer complete
0 = Serial transfer not complete

DCOL — Data Collision Flag

This clearable, read-only bit is set if the SIOP data register is read or written during a serial transfer. Clear the DCOL bit by reading the SIOP status register with the SPIF bit set, and then reading or writing the SIOP data register. Resets clear the DCOL bit.

1 = Invalid access of SIOP data register
0 = No invalid access of SIOP data register

### 9.2.3 SIOP Data Register (SDR)

The SIOP data register, shown in Figure 9-6, is both the transmit data register and the receive data register. To read or write the SIOP data register, the SPE bit in the SIOP control register must be set.

**SDR** — SIOP Data Register **$000C**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 9-6. SIOP Data Register (SDR)**

With the SIOP configured for master mode, writing to the SIOP data register initiates a serial transfer. This register is not buffered. Writing to the SIOP data register overwrites the previous contents. Reading or writing to the SIOP data register while a transmission is in progress can cause invalid data to be transmitted or received.

# SECTION 10
# ANALOG-TO-DIGITAL CONVERTER (ADC)

This section describes the four-channel, 8-bit analog-to-digital converter.

## 10.1 ADC Operation

The A/D conversion process is ratiometric, using two reference voltages, $V_{RH}$ and $V_{SS}$. Conversion accuracy is guaranteed only if $V_{RH}$ is equal to $V_{DD}$.

A multiplexer selects one of four analog input channels (AN0, AN1, AN2, or AN3) for sampling. A comparator successively compares the output of an internal D/A converter to the sampled analog input. Control logic changes the D/A converter input one bit at a time, starting with the MSB, until the D/A converter output matches the sampled analog input. The conversion is monotonic and has no missing codes.

An analog input voltage equal to $V_{RH}$ converts to digital $FF; an input voltage greater than $V_{RH}$ converts to $FF with no overflow. An analog input voltage equal to $V_{SS}$ converts to digital $00. For ratiometric conversion, the source of each analog input should use $V_{RH}$ as the supply voltage and be referenced to $V_{SS}$.

Pins PC3/AN3, PC4/AN2, PC5/AN1, and PC6/AN0 are the four inputs to the multiplexer. Each channel of conversion takes 32 internal clock cycles, and the clock frequency must be equal to or greater than 1 MHz. If the internal clock frequency is less than 1 MHz, the internal RC oscillator (nominally 1.5 MHz) must be used for the ADC conversion clock. Make this selection by setting the ADRC bit to logic one in the ADC status and control register.

## 10.2 ADC I/O Registers

The following registers control and monitor operation of the ADC:

- ADC status and control register (ADSCR)
- ADC data register (ADDR)

### 10.2.1 ADC Status and Control Register (ADSCR)

The ADC status and control register, shown in Figure 10-1, contains a conversion complete flag and four writable control bits. Writing to ADSCR clears the conversion complete flag and starts a new conversion sequence.

**ADSCR** — ADC Status and Control Register **$001E**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | CCF | ADRC | ADON | 0 | 0 | CH2 | CH1 | CH0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-1. ADC Status and Control Register (ADSCR)**

CCF — Conversion Complete Flag

This read-only bit is automatically set when an analog-to-digital conversion is complete, and a new result can be read from the ADC data register. Clear the CCF bit by writing to the ADC status and control register or by reading the ADC data register. Resets clear the CCF bit.

1 = Conversion complete
0 = Conversion not complete

ADRC — ADC RC (Oscillator)

This read/write bit turns on the internal RC oscillator to drive the ADC. If the internal clock frequency ($f_{OP}$) is less than 1 MHz, ADRC must be set. When the RC oscillator is turned on, it requires a time, $t_{ADRC}$, to stabilize, and results can be inaccurate during this time. Resets clear the ADRC bit.

1 = Internal RC oscillator drives ADC
0 = Internal clock drives ADC

When the internal RC oscillator is being used as the ADC clock, two limitations apply:

- Because of the frequency tolerance of the RC oscillator and its asynchronism with the internal clock, the conversion complete flag must be used to determine when a conversion sequence is complete.
- The conversion process runs at the nominal 1.5-MHz rate, but the conversion results must be transferred to the ADC data register synchronously with the internal clock; therefore, the conversion process is limited to a maximum of one channel every internal clock cycle.

ADON — ADC On

This read/write bit turns on the ADC. When the ADC is on, it requires a time, $t_{ADON}$, for the current sources to stabilize. During this time, results can be inaccurate. Resets clear the ADON bit.

1 = ADC turned on
0 = ADC turned off

Bits 4–2 — Not used

Bits 4–2 always read as logic zeros.

CH2–CH0 — Channel 2–Channel 0

These read/write bits select one of eight ADC input channels as shown in Table 10-1. Channels 0–3 are the port C input pins, PC3/AN3, PC4/AN2, PC5/AN1, and PC6/AN0. Channels 4–6 can be used for reference measurements. Channel 7 is reserved for factory testing.

**Table 10-1. ADC Input Channel Selection**

| CH[2:1:0] | Channel | Signal |
|---|---|---|
| 000 | 0 | AN0 Port C Bit 6 |
| 001 | 1 | AN1 Port C Bit 5 |
| 010 | 2 | AN2 Port C Bit 4 |
| 011 | 3 | AN3 Port C Bit 3 |
| 100 | 4 | $V_{RH}$ Port C Bit 7 |
| 101 | 5 | $(V_{RH} + V_{SS}) \div 2$ |
| 110 | 6 | $V_{SS}$ |
| 111 | 7 | Reserved |

To prevent excess power dissipation, do not use a port C pin as an analog input and a digital input at the same time.

Using one of the port C pins as the ADC input does not affect the ability to use the remaining port C pins as digital inputs.

Reading a port C pin that is selected as an analog input returns a logic zero.

### 10.2.2 ADC Data Register (ADDR)

The ADC data register is a read-only register that contains the result of the most recent analog-to-digital conversion.

**ADDR** — ADC Data Register **$001D**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| RESET: | UNAFFECTED BY RESET | | | | | | | |

**Figure 10-1. ADC Data Register (ADDR)**

# SECTION 11
# EPROM/OTPROM

This section describes how to program the 2113-byte EPROM/OTPROM.

## 11.1 EPROM/OTPROM Programming

There are two ways to program the EPROM/OTPROM:

- Manipulating the control bits in the EPROM programming register to program the EPROM/OTPROM on a byte-by-byte basis
- Activating the bootloader ROM to download the contents of an external memory device to the on-chip EPROM/OTPROM

### 11.1.1 EPROM Programming Register (EPROG)

The EPROM programming register, shown in Figure 11-1, contains the control bits for programming the EPROM/OTPROM.

**EPROG** — EPROM Programming Register **$001C**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | LATCH | 0 | EPGM |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-1. EPROM Programming Register (EPROG)**

LATCH — EPROM Bus Latch

This read/write bit latches the address and data buses for EPROM/OTPROM programming. Clearing the LATCH bit automatically clears the EPGM bit. EPROM/OTPROM data cannot be read while the LATCH bit is set. Resets clear the LATCH bit.

1 = Address and data buses configured for EPROM/OTPROM programming
0 = Address and data buses configured for normal operation

EPGM — EPROM Programming

This read/write bit applies the voltage from the $\overline{\text{IRQ}}/V_{PP}$ pin to the EPROM/OTPROM. To write the EPGM bit, the LATCH bit must already be set. Clearing the LATCH bit also clears the EPGM bit. Resets clear the EPGM bit.

1 = EPROM/OTPROM programming power switched on
0 = EPROM/OTPROM programming power switched off

**NOTE**

Writing logic ones to both the LATCH and EPGM bits with a single instruction sets LATCH and clears EPGM. LATCH must be set first by a separate instruction.

Bits 7–3 and Bit 1— Reserved

Bits 7–3 and bit 1 are factory test bits that always read as logic zeros.

Take the following steps to program a byte of EPROM/OTPROM:

1. Apply 16.5 V to the $\overline{\text{IRQ}}/V_{PP}$ pin.
2. Set the LATCH bit.
3. Write to any EPROM/OTPROM address.
4. Set the EPGM bit for a time, $t_{EPGM}$, to apply the programming voltage.
5. Clear the LATCH bit.

### 11.1.2 Bootloader ROM

The bootloader ROM, located at addresses \$1F00–\$1FEF, contains routines for copying an external EPROM to the on-chip EPROM/OTPROM.

The bootloader copies to the following MC68HC705P9 EPROM/OTPROM addresses:

- \$0020–\$004F
- \$0100–\$0900
- \$1FF0–\$1FFF

The addresses of the code in the external EPROM must match the MC68HC705P9 addresses. The bootloader ignores all other addresses.

Figure 11-2 shows the circuit used to download to the on-chip EPROM/OTPROM from a 2764 EPROM. The bootloader circuit includes an external 12-bit counter to address the external EPROM. Operation is fastest when unused external EPROM addresses contain \$00. The bootloader function begins when a rising edge occurs on the $\overline{\text{RESET}}$ pin while the $\overline{\text{IRQ}}/V_{PP}$ pin is at $V_{PP}$, and the PD7/TCAP pin is at logic one.

The logical states of the PC4/AN2 and PC3/AN3 pins select the bootloader function, as Table 11-1 shows.

**Table 11-1. Bootloader Function Selection**

| PC4/AN2 | PC3/AN3 | Function |
|---|---|---|
| 1 | 1 | Program and Verify |
| 1 | 0 | Verify Only |

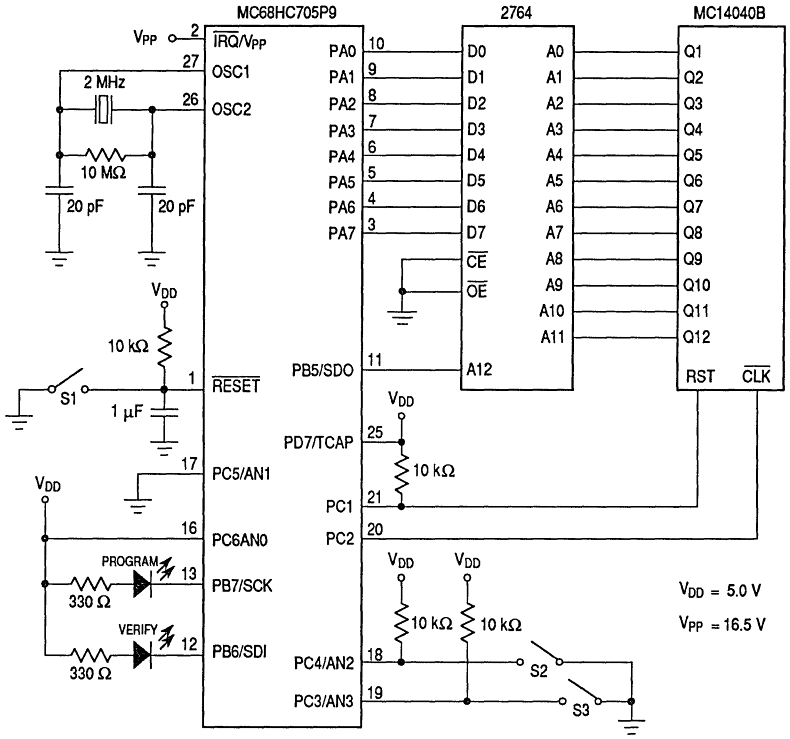


**Figure 11-2. Bootloader Circuit**

Complete the following steps to bootload the MCU:

1. Turn off all power to the circuit.
2. Install the EPROM containing the code to be downloaded.
3. Install the MCU.
4. Select the bootloader function:
    a. Open switches S2 and S3 to select the program and verify function.
    b. Open only switch S2 to select only the verify function.

5. Close switch S1.

6. Turn on the $V_{DD}$ power supply.

**CAUTION**

Turn on the $V_{DD}$ power supply **before** turning on the $V_{PP}$ power supply.

7. Turn on the $V_{PP}$ power supply.

8. Open switch S1. The bootloader code begins to execute. If the PROGRAM function is selected, the PROGRAM LED turns on during programming. If the VERIFY function is selected, the VERIFY LED turns on when verification is successful. The PROGRAM and VERIFY functions take about 10 seconds.

9. Close switch S1.

10. Turn off the $V_{PP}$ power supply.

**CAUTION**

Turn off the $V_{PP}$ power supply **before** turning off the $V_{DD}$ power supply.

11. Turn off the $V_{DD}$ power supply.

## 11.2 EPROM Erasing

The erased state of an EPROM bit is zero. Erase the EPROM by exposing it to 15 Ws/cm$^2$ of ultraviolet light with a wavelength of 2537 angstroms. Position the ultraviolet light source one inch from the EPROM. Do not use a shortwave filter.

Cerdip packages have a transparent window for erasing the EPROM with ultraviolet light. In the windowless PDIP and SOIC packages, the 2113 EPROM bytes function as one-time programmable ROM (OTPROM)

## 11.3 Mask Option Register (MOR)

The MOR is an EPROM/OTPROM byte that is programmable only with the bootloader function. The MOR controls the following options:

- LSB-first or MSB-first SIOP data transfer
- Edge-triggered or edge- and level-triggered external interrupt pin
- Enabled or disabled COP watchdog

**MOR** — Mask Option Register **$0900**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SIOP | IRQ | COPE |
| RESET: | UNAFFECTED BY RESET | | | | | | | |
| ERASED: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-3. Mask Option Register (MOR)**

SIOP — Serial I/O Port

The SIOP bit controls the shift direction into and out of the SIOP shift register.

1 = SIOP data transferred LSB first (bit 0 first)

0 = SIOP data transferred MSB first (bit 7 first)

IRQ — Interrupt Request

The LEVEL bit makes the external interrupt function of the $\overline{\text{IRQ}}/V_{PP}$ pin level-triggered as well as edge-triggered.

1 = $\overline{\text{IRQ}}/V_{PP}$ pin negative edge-triggered and low-level-triggered

0 = $\overline{\text{IRQ}}/V_{PP}$ pin negative edge-triggered only

COPE — COP Enable

COPE enables the COP watchdog. In applications that have wait cycles longer than the COP watchdog timeout period, the COP watchdog can be disabled by not programming the COPE bit to logic one.

1 = COP watchdog enabled

0 = COP watchdog disabled

# SECTION 12
# INSTRUCTION SET

This section describes the addressing modes and the types of instructions.

## 12.1 Addressing Modes

The CPU uses eight addressing modes for flexibility in accessing data. These addressing modes define the manner in which the CPU finds the data required to execute an instruction. The eight addressing modes are as follows:

- Inherent
- Immediate
- Direct
- Extended
- Indexed, no offset
- Indexed, 8-bit offset
- Indexed, 16-bit offset
- Relative

### 12.1.1 Inherent

Inherent instructions are those that have no operand, such as return from interrupt (RTI) and stop (STOP). Some of the inherent instructions act on data in the CPU registers, such as set carry flag (SEC) and increment accumulator (INCA). Inherent instructions require no memory address and are one byte long. Table 12-1 lists the instructions that use inherent addressing.

**Table 12-1. Inherent Addressing Instructions**

| Instruction | Mnemonic |
|---|---|
| Arithmetic Shift Left | ASLA, ASLX |
| Arithmetic Shift Right | ASRA, ASRX |
| Clear Carry Bit | CLC |
| Clear Interrupt Mask | CLI |
| Clear | CLRA, CLRX |
| Complement | COMA, COMX |
| Decrement | DECA, DECX |
| Increment | INCA, INCX |
| Logical Shift Left | LSLA, LSLX |
| Logical Shift Right | LSRA, LSRX |
| Multiply Index Register by Accumulator (Unsigned) | MUL |
| Negate | NEGA, NEGX |
| No Operation | NOP |
| Rotate Left through Carry | ROLA, ROLX |
| Rotate Right through Carry | RORA, RORX |
| Reset Stack Pointer | RSP |
| Return from Interrupt | RTI |
| Return from Subroutine | RTS |
| Set Carry Bit | SEC |
| Set Interrupt Mask | SEI |
| Enable $\overline{\text{IRQ}}$ and Stop Oscillator | STOP |
| Software Interrupt | SWI |
| Transfer Accumulator to Index Register | TAX |
| Test for Negative or Zero | TSTA, TSTX |
| Transfer Index Register to Accumulator | TXA |
| Enable Interrupts and Halt CPU | WAIT |

### 12.1.2 Immediate

Immediate instructions are those that contain a value to be used in an operation with the value in the accumulator or index register. Immediate instructions require no memory address and are two bytes long. The opcode is the first byte and the immediate data value is the second byte. Table 12-2 lists the instructions that use immediate addressing.

**Table 12-2. Immediate Addressing Instructions**

| Instruction | Mnemonic |
|---|---|
| Add Memory and Carry to Accumulator | ADC |
| Add Memory to Accumulator | ADD |
| Logical AND Memory with Accumulator | AND |
| Bit Test Memory with Accumulator (Logical Compare) | BIT |
| Arithmetic Compare Accumulator with Memory | CMP |
| Arithmetic Compare Index Register with Memory | CPX |
| Exclusive OR Memory with Accumulator | EOR |
| Load Accumulator from Memory | LDA |
| Load Index Register from Memory | LDX |
| Logical Inclusive OR Memory with Accumulator | ORA |
| Subtract Memory and Carry from Accumulator | SBC |
| Subtract Memory from Accumulator | SUB |

### 12.1.3 Direct

Direct instructions can access any of the first 256 memory addresses with two bytes. The first byte is the opcode, and the second is the low byte of the operand address. In direct addressing, the CPU automatically uses $00 as the high byte of the operand address. BRSET and BRCLR are three-byte instructions that use direct addressing to access the operand and relative addressing to specify a branch destination. Table 12-3 lists the instructions that use direct addressing.

**Table 12-3. Direct Addressing Instructions**

| Instruction | Mnemonic |
|---|---|
| Add Memory and Carry to Accumulator | ADC |
| Add Memory to Accumulator | ADD |
| Logical AND Memory with Accumulator | AND |
| Arithmetic Shift Left | ASL |
| Arithmetic Shift Right | ASR |
| Clear Bit | BCLR |
| Bit Test Memory with Accumulator (Logical Compare) | BIT |
| Branch if Bit Clear | BRCLR |
| Branch if Bit Set | BRSET |
| Set Bit | BSET |
| Clear | CLR |
| Arithmetic Compare Accumulator with Memory | CMP |
| Complement | COM |
| Arithmetic Compare Index Register with Memory | CPX |
| Decrement | DEC |
| Exclusive OR Memory with Accumulator | EOR |
| Increment | INC |
| Jump | JMP |
| Jump to Subroutine | JSR |
| Load Accumulator from Memory | LDA |
| Load Index Register from Memory | LDX |
| Logical Shift Left | LSL |
| Logical Shift Right | LSR |
| Negate | NEG |
| Logical Inclusive OR Memory with Accumulator | ORA |
| Rotate Left through Carry | ROL |
| Rotate Right through Carry | ROR |
| Subtract Memory and Carry from Accumulator | SBC |
| Store Accumulator in Memory | STA |
| Store Index Register in Memory | STX |
| Subtract Memory from Accumulator | SUB |
| Test for Negative or Zero | TST |

### 12.1.4 Extended

Extended instructions use only three bytes to access any address in memory. The first byte is the opcode; the second and third bytes are the high and low bytes of the operand address.

When using the Motorola assembler, the programmer does not need to specify whether an instruction is direct or extended. The assembler automatically selects the shortest form of the instruction. Table 12-4 lists the instructions that use extended addressing.

**Table 12-4. Extended Addressing Instructions**

| Instruction | Mnemonic |
|---|---|
| Add Memory and Carry to Accumulator | ADC |
| Add Memory to Accumulator | ADD |
| Logical AND Memory with Accumulator | AND |
| Bit Test Memory with Accumulator (Logical Compare) | BIT |
| Arithmetic Compare Accumulator with Memory | CMP |
| Arithmetic Compare Index Register with Memory | CPX |
| Exclusive OR Memory with Accumulator | EOR |
| Jump | JMP |
| Jump to Subroutine | JSR |
| Load Accumulator from Memory | LDA |
| Load Index Register from Memory | LDX |
| Logical Inclusive OR Memory with Accumulator | ORA |
| Subtract Memory and Carry from Accumulator | SBC |
| Store Accumulator in Memory | STA |
| Store Index Register in Memory | STX |
| Subtract Memory from Accumulator | SUB |

### 12.1.5 Indexed, No Offset

Indexed instructions with no offset are one-byte instructions that can access data with variable addresses within the first 256 memory locations. The index register contains the low byte of the conditional address of the operand. The CPU automatically uses $00 as the high byte, so these instructions can address locations $0000–$00FF.

Indexed, no offset instructions are often used to move a pointer through a table or to hold the address of a frequently used RAM or I/O location. Table 12-5 lists the instructions that use indexed, no offset addressing.

### 12.1.6 Indexed, 8-Bit Offset

Indexed, 8-bit offset instructions are two-byte instructions that can access data with variable addresses within the first 511 memory locations. The CPU adds the unsigned byte in the index register to the unsigned byte following the opcode. The sum is the conditional address of the operand. These instructions can access locations $0000–$01FE.

Indexed, 8-bit offset instructions are useful for selecting the kth element in an n-element table. The table can begin anywhere within the first 256 memory locations and could extend as far as location 510 ($01FE). The k value would typically be in the index register, and the address of the beginning of the table would be in the byte following the opcode. Table 12-5 lists the instructions that use indexed, 8-bit offset addressing.

### 12.1.7 Indexed, 16-Bit Offset

Indexed, 16-bit offset instructions are three-byte instructions that can access data with variable addresses at any location in memory. The CPU adds the unsigned byte in the index register to the two unsigned bytes following the opcode. The sum is the conditional address of the operand. The first byte after the opcode is the high byte of the 16-bit offset; the second byte is the low byte of the offset. These instructions can address any location in memory.

Indexed, 16-bit offset instructions are useful for selecting the kth element in an n-element table anywhere in memory.

As with direct and extended addressing, the Motorola assembler determines the shortest form of indexed addressing. Table 12-5 lists the instructions that use indexed, 16-bit offset addressing.

## Table 12-5. Indexed Addressing Instructions

| Instruction | Mnemonic | No Offset | 8-Bit Offset | 16-Bit Offset |
|---|---|---|---|---|
| Add Memory and Carry to Accumulator | ADC | √ | √ | √ |
| Add Memory to Accumulator | ADD | √ | √ | √ |
| Logical AND Memory with Accumulator | AND | √ | √ | √ |
| Arithmetic Shift Left | ASL | √ | √ | |
| Arithmetic Shift Right | ASR | √ | √ | |
| Bit Test Memory with Accumulator (Logical Compare) | BIT | √ | √ | √ |
| Clear | CLR | √ | √ | |
| Arithmetic Compare Accumulator with Memory | CMP | √ | √ | √ |
| Complement | COM | √ | √ | |
| Arithmetic Compare Index Register with Memory | CPX | √ | √ | √ |
| Decrement | DEC | √ | √ | |
| Exclusive OR Memory with Accumulator | EOR | √ | √ | √ |
| Increment | INC | √ | √ | |
| Jump | JMP | √ | √ | √ |
| Jump to Subroutine | JSR | √ | √ | √ |
| Load Accumulator from Memory | LDA | √ | √ | √ |
| Load Index Register from Memory | LDX | √ | √ | √ |
| Logical Shift Left | LSL | √ | √ | |
| Logical Shift Right | LSR | √ | √ | |
| Negate | NEG | √ | √ | |
| Logical Inclusive OR Memory with Accumulator | ORA | √ | √ | √ |
| Rotate Left through Carry | ROL | √ | √ | |
| Rotate Right through Carry | ROR | √ | √ | |
| Subtract Memory and Carry from Accumulator | SBC | √ | √ | √ |
| Store Accumulator in Memory | STA | √ | √ | √ |
| Store Index Register in Memory | STX | √ | √ | √ |
| Subtract Memory from Accumulator | SUB | √ | √ | √ |
| Test for Negative or Zero | TST | √ | √ | |

### 12.1.8 Relative

Relative addressing is only for branch instructions and bit test and branch instructions. If the branch condition is true, the CPU finds the conditional branch destination by adding the signed byte following the opcode to the contents of the program counter. If the branch condition is not true, the CPU goes to the next instruction. The offset is a signed, two's complement byte that gives a branching range of –127 to +128 bytes from the address of the next location after the branch instruction.

When using the Motorola assembler, the programmer does not need to calculate the offset, because the assembler determines the proper offset and verifies that it is within the span of the branch. Table 12-6 lists the instructions that use relative addressing.

**Table 12-6. Relative Addressing Instructions**

| Instruction | Mnemonic |
|---|---|
| Branch if Carry Clear | BCC |
| Branch if Carry Set | BCS |
| Branch if Equal | BEQ |
| Branch if Half-Carry Clear | BHCC |
| Branch if Half-Carry Set | BHCS |
| Branch if Higher | BHI |
| Branch if Higher or Same | BHS |
| Branch if Interrupt Line High | BIH |
| Branch if Interrupt Line Low | BIL |
| Branch if Lower | BLO |
| Branch if Lower or Same | BLS |
| Branch if Interrupt Mask Clear | BMC |
| Branch if Minus | BMI |
| Branch if Interrupt Mask Set | BMS |
| Branch if Not Equal | BNE |
| Branch if Plus | BPL |
| Branch Always | BRA |
| Branch if Bit Clear | BRCLR |
| Branch if Bit Set | BRSET |
| Branch Never | BRN |
| Branch to Subroutine | BSR |

## 12.2 Instruction Types

The MCU instructions fall into the following five categories:

- Register/memory
- Read-modify-write
- Jump/branch
- Bit manipulation
- Control

### 12.2.1 Register/Memory Instructions

Most of these instructions use two operands. One operand is in either the accumulator or the index register. The CPU finds the other operand in memory. Register/memory instructions use all the addressing modes except relative.

Table 12-7 lists the register/memory instructions.

**Table 12-7. Register/Memory Instructions**

| Instruction | Mnemonic |
|---|---|
| Load Accumulator from Memory | LDA |
| Load Index Register from Memory | LDX |
| Store Accumulator in Memory | STA |
| Store Index Register in Memory | STX |
| Add Memory to Accumulator | ADD |
| Add Memory and Carry to Accumulator | ADC |
| Subtract Memory from Accumulator | SUB |
| Subtract Memory and Carry from Accumulator | SBC |
| Logical AND Memory with Accumulator | AND |
| Logical Inclusive OR Memory with Accumulator | ORA |
| Arithmetic Compare Accumulator with Memory | CMP |
| Arithmetic Compare Index Register with Memory | CPX |
| Bit Test Memory with Accumulator (Logical Compare) | BIT |
| Multiply Index Register by Accumulator (Unsigned) | MUL |

### 12.2.2 Read-Modify-Write Instructions

These instructions read a memory location or a register, modify its contents, and write the modified value back to the memory location or to the register. The test for negative or zero (TST) instruction is an exception to the read-modify-write sequence because it does not write a replacement value. Read-modify-write instructions use the following addressing modes:

- Inherent
- Direct
- Indexed, no offset
- Indexed, 8-bit offset

Table 12-8 lists the read-modify-write instructions.

**Table 12-8. Read-Modify-Write Instructions**

| Instruction | Mnemonic |
|---|---|
| Increment | INC |
| Decrement | DEC |
| Clear | CLR |
| Complement | COM |
| Negate (Two's Complement) | NEG |
| Rotate Left through Carry | ROL |
| Rotate Right through Carry | ROR |
| Logical Shift Left | LSL |
| Logical Shift Right | LSR |
| Arithmetic Shift Right | ASR |
| Test for Negative or Zero | TST |

### 12.2.3 Jump/Branch Instructions

Jump instructions allow the CPU to interrupt the normal sequence of the program counter. The unconditional jump (JMP) and jump to subroutine (JSR) instructions have no register operand. Jump instructions use the following addressing modes:

- Direct
- Extended
- Indexed, no offset

- Indexed, 8-bit offset
- Indexed, 16-bit offset

Branch instructions allow the CPU to interrupt the normal sequence of the program counter when a test condition is met. If the test condition is not met, the branch is not performed. All branch instructions use relative addressing.

Bit test and branch instructions cause a branch based on the state of any readable bit in the first 256 memory locations. These three-byte instructions use a combination of direct addressing and relative addressing. The direct address of the byte to be tested is in the byte following the opcode. The third byte is the signed offset byte. The CPU finds the conditional branch destination by adding the third byte to the program counter if the specified bit tests true. The bit to be tested and its condition (set or clear) is part of the opcode. The span of branching is from –128 to +127 from the address of the next location after the branch instruction. The CPU also transfers the tested bit to the carry/borrow bit of the condition code register. Table 12-9 lists the jump and branch instructions.

**Table 12-9. Jump and Branch Instructions**

| Instruction | Mnemonic |
|---|---|
| Branch Always | BRA |
| Branch Never | BRN |
| Branch if Bit Clear | BRCLR |
| Branch if Bit Set | BRSET |
| Branch if Higher | BHI |
| Branch if Lower or Same | BLS |
| Branch if Carry Clear | BCC |
| Branch if Higher or Same | BHS |
| Branch if Carry Set | BCS |
| Branch if Lower | BLO |
| Branch if Not Equal | BND |
| Branch if Equal | BEQ |
| Branch if Half-Carry Bit Clear | BHCC |
| Branch if Half-Carry Bit Set | BHCS |
| Branch if Plus | BPL |
| Branch if Minus | BMI |
| Branch if Interrupt Mask Clear | BMC |
| Branch if Interrupt Mask Set | BMS |
| Branch if Interrupt Line Low | BIL |
| Branch if Interrupt Line High | BIH |
| Branch to Subroutine | BSR |
| Jump | JMP |
| Jump to Subroutine | JSR |

### 12.2.4 Bit Manipulation Instructions

The CPU can set or clear any writable bit in the first 256 bytes of memory. Port registers, port data direction registers, timer registers, and on-chip RAM locations are in the first 256 bytes of memory. The CPU can also test and branch based on the state of any bit in any of the first 256 memory locations. Bit manipulation instructions use direct addressing. Table 12-10 lists these instructions.

**Table 12-10. Bit Manipulation Instructions**

| Instruction | Mnemonic |
|---|---|
| Set Bit | BSET |
| Clear Bit | BCLR |
| Branch if Bit Clear | BRCLR |
| Branch if Bit Set | BRSET |

### 12.2.5 Control Instructions

These register reference instructions control CPU operation during program execution. Control instructions, listed in Table 12-11, use inherent addressing.

**Table 12-11. Control Instructions**

| Instruction | Mnemonic |
|---|---|
| Transfer Accumulator to Index Register | TAX |
| Transfer Index Register to Accumulator | TXA |
| Set Carry Bit | SEC |
| Clear Carry Bit | CLC |
| Set Interrupt Mask | SEI |
| Clear Interrupt Mask | CLI |
| Software Interrupt | SWI |
| Return from Subroutine | RTI |
| Reset Stack Pointer | RSP |
| No Operation | NOP |
| Stop | STOP |
| Wait | WAIT |

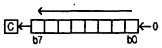
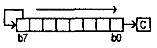
## 12.3 Instruction Set Summary

Table 12-12 shows all MC68HC705P9 instructions in all possible addressing modes. The table shows the operand construction and the execution time in internal clock cycles ($t_{CYC}$) of each instruction. One internal clock cycle equals two oscillator input cycles. The following legend summarizes the symbols and abbreviations used in Table 12-12.

### Abbreviations and Symbols

| | | | |
|---|---|---|---|
| A | Accumulator | PCH | Program counter high byte |
| C | Carry/borrow flag | PCL | Program counter low byte |
| CCR | Condition code register | REL | Relative addressing |
| dd | Address of operand in direct addressing | rel | Offset byte for relative addressing |
| dd rr | Address (dd) of operand and offset (rr) of branch instruction for bit test instructions | rr | Offset byte of branch instruction |
| DIR | Direct addressing | SP | Stack pointer |
| ee ff | High (ee) and low (ff) bytes of offset in indexed, 16-bit offset addressing | X | Index register |
| EXT | Extended addressing | Z | Zero flag |
| ff | Offset byte in indexed, 8-bit offset addressing | • | AND |
| H | Half-carry flag | – | Not affected |
| hh ll | High (hh) and low (ll) bytes of operand address in extended addressing | ? | If |
| I | Interrupt mask | — | NOT |
| ii | Operand byte for immediate addressing | ( ) | Contents of |
| IMM | Immediate addressing | ← | Is loaded with |
| INH | Inherent addressing | : | Concatenated with |
| IX | Indexed, no offset addressing | × | Multiplication |
| IX1 | Indexed, 8-bit offset addressing | –( ) | Negation (two's complement) |
| IX2 | Indexed, 16-bit offset addressing | + | Inclusive OR |
| M | Any memory location (1 byte) | ↕ | Set if true; clear if not true |
| N | Negative flag | ⊕ | Exclusive OR |
| n | Any bit (7,6,5 . . . 0) | + | Addition |
| opr | Operand byte | – | Subtraction |
| PC | Program counter | | |

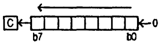
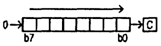
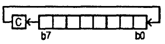
# Table 12-12. Instruction Set (Sheet 1 of 4)

| Source Form(s) | Operation | Description | Addressing Mode for Operand | Machine Coding (hexadecimal) | | Cycles | Condition Code | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Opcode | Operand | | H | I | N | Z | C |
| ADC opr | Add with carry | A ← (A) + (M) + C | IMM | A9 | ii | 2 | ↕ | – | ↕ | ↕ | ↕ |
| | | | DIR | B9 | dd | 3 | | | | | |
| | | | EXT | C9 | hh ll | 4 | | | | | |
| | | | IX2 | D9 | ee ff | 5 | | | | | |
| | | | IX1 | E9 | ff | 4 | | | | | |
| | | | IX | F9 | | 3 | | | | | |
| ADD opr | Add without carry | A ← (A) + (M) | IMM | AB | ii | 2 | ↕ | – | ↕ | ↕ | ↕ |
| | | | DIR | BB | dd | 3 | | | | | |
| | | | EXT | CB | hh ll | 4 | | | | | |
| | | | IX2 | DB | ee ff | 5 | | | | | |
| | | | IX1 | EB | ff | 4 | | | | | |
| | | | IX | FB | | 3 | | | | | |
| AND opr | Logical AND | A ← (A) • (M) | IMM | A4 | ii | 2 | – | – | ↕ | ↕ | – |
| | | | DIR | B4 | dd | 3 | | | | | |
| | | | EXT | C4 | hh ll | 4 | | | | | |
| | | | IX2 | D4 | ee ff | 5 | | | | | |
| | | | IX1 | E4 | ff | 4 | | | | | |
| | | | IX | F4 | | 3 | | | | | |
| ASL opr | Arithmetic shift left |  | DIR | 38 | dd | 5 | – | – | ↕ | ↕ | ↕ |
| ASLA | | | INH | 48 | | 3 | | | | | |
| ASLX | | | INH | 58 | | 3 | | | | | |
| ASL opr | | | IX1 | 68 | ff | 6 | | | | | |
| ASL opr | | | IX | 78 | | 5 | | | | | |
| ASR opr | Arithmetic shift right |  | DIR | 37 | dd | 5 | – | – | ↕ | ↕ | ↕ |
| ASRA | | | INH | 47 | | 3 | | | | | |
| ASRX | | | INH | 57 | | 3 | | | | | |
| ASR opr | | | IX1 | 67 | ff | 6 | | | | | |
| ASR opr | | | IX | 77 | | 5 | | | | | |
| BCC rel | Branch if carry bit clear | ? C = 0 | REL | 24 | rr | 3 | – | – | – | – | – |
| BCLR n opr | Clear bit n | Mn ← 0 | DIR (b0) | 11 | dd | 5 | – | – | – | – | – |
| | | | DIR (b1) | 13 | dd | 5 | | | | | |
| | | | DIR (b2) | 15 | dd | 5 | | | | | |
| | | | DIR (b3) | 17 | dd | 5 | | | | | |
| | | | DIR (b4) | 19 | dd | 5 | | | | | |
| | | | DIR (b5) | 1B | dd | 5 | | | | | |
| | | | DIR (b6) | 1D | dd | 5 | | | | | |
| | | | DIR (b7) | 1F | dd | 5 | | | | | |
| BCS rel | Branch if carry bit set | ? C = 1 | REL | 25 | rr | 3 | – | – | – | – | – |
| BEQ rel | Branch if equal | ? Z = 1 | REL | 27 | rr | 3 | – | – | – | – | – |
| BHCC rel | Branch if half carry bit clear | ? H = 0 | REL | 28 | rr | 3 | – | – | – | – | – |
| BHCS rel | Branch if half carry bit set | ? H = 1 | REL | 29 | rr | 3 | – | – | – | – | – |
| BHI rel | Branch if higher | ? C + Z = 0 | REL | 22 | rr | 3 | – | – | – | – | – |
| BHS rel | Branch if higher or same | ? C = 0 | REL | 24 | rr | 3 | – | – | – | – | – |
| BIH rel | Branch if $\overline{IRQ}$ pin high | ? $\overline{IRQ}$ = 1 | REL | 2F | rr | 3 | – | – | – | – | – |
| BIL rel | Branch if $\overline{IRQ}$ pin low | ? $\overline{IRQ}$ = 0 | REL | 2E | rr | 3 | – | – | – | – | – |
| BIT rel | Bit test accumulator contents with memory contents | (A) • (M) | IMM | A5 | ii | 2 | – | – | ↕ | ↕ | – |
| | | | DIR | B5 | dd | 3 | | | | | |
| | | | EXT | C5 | hh ll | 4 | | | | | |
| | | | IX2 | D5 | ee ff | 5 | | | | | |
| | | | IX1 | E5 | ff | 4 | | | | | |
| | | | IX | F5 | | 3 | | | | | |
| BLO rel | Branch if lower | ? C = 1 | REL | 25 | rr | 3 | – | – | – | – | – |
| BLS rel | Branch if lower or same | ? C + Z = 1 | REL | 23 | rr | 3 | – | – | – | – | – |
| BMC rel | Branch if interrupt mask clear | ? I = 0 | REL | 2C | rr | 3 | – | – | – | – | – |
| BMI rel | Branch if minus | ? N = 1 | REL | 2B | rr | 3 | – | – | – | – | – |
| BMS rel | Branch if interrupt mask set | ? I = 0 | REL | 2D | rr | 3 | – | – | – | – | – |
| BNE rel | Branch if not equal | ? Z = 0 | REL | 26 | rr | 3 | – | – | – | – | – |
| BPL rel | Branch if plus | ? N = 0 | REL | 2A | rr | 3 | – | – | – | – | – |

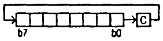
# Table 12-12. Instruction Set (Sheet 2 of 4)

| Source Form(s) | Operation | Description | Addressing Mode for Operand | Machine Coding (hexadecimal) | | Cycles | Condition Code | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Opcode | Operand | | H | I | N | Z | C |
| BRA rel | Branch always | ? 1 = 1 | REL | 20 | rr | 3 | – | – | – | – | – |
| BRCLR n opr rel | Branch if bit n clear | ? Mn = 0 | DIR (b0) | 01 | dd rr | 5 | – | – | – | – | ↕ |
| | | | DIR (b1) | 03 | dd rr | 5 | | | | | |
| | | | DIR (b2) | 05 | dd rr | 5 | | | | | |
| | | | DIR (b3) | 07 | dd rr | 5 | | | | | |
| | | | DIR (b4) | 09 | dd rr | 5 | | | | | |
| | | | DIR (b5) | 0B | dd rr | 5 | | | | | |
| | | | DIR (b6) | 0D | dd rr | 5 | | | | | |
| | | | DIR (b7) | 0F | dd rr | 5 | | | | | |
| BRN rel | Branch never | ? 1 = 0 | REL | 21 | rr | 3 | – | – | – | – | – |
| BRSET n opr rel | Branch if bit n set | ? Mn = 1 | DIR (b0) | 00 | dd rr | 5 | – | – | – | – | ↕ |
| | | | DIR (b1) | 02 | dd rr | 5 | | | | | |
| | | | DIR (b2) | 04 | dd rr | 5 | | | | | |
| | | | DIR (b3) | 06 | dd rr | 5 | | | | | |
| | | | DIR (b4) | 08 | dd rr | 5 | | | | | |
| | | | DIR (b5) | 0A | dd rr | 5 | | | | | |
| | | | DIR(b6) | 0C | dd rr | 5 | | | | | |
| | | | DIR (b7) | 0E | dd rr | 5 | | | | | |
| BSET n opr | Set bit n | Mn ← 1 | DIR (b0) | 10 | dd | 5 | – | – | – | – | – |
| | | | DIR (b1) | 12 | dd | 5 | | | | | |
| | | | DIR (b2) | 14 | dd | 5 | | | | | |
| | | | DIR (b3) | 16 | dd | 5 | | | | | |
| | | | DIR (b4) | 18 | dd | 5 | | | | | |
| | | | DIR (b5) | 1A | dd | 5 | | | | | |
| | | | DIR (b6) | 1C | dd | 5 | | | | | |
| | | | DIR (b7) | 1E | dd | 5 | | | | | |
| BSR rel | Branch to subroutine | PC ← (PC) + 2; push (PCL)<br>SP ← (SP) – 1; push (PCH)<br>SP ← (SP) – 1<br>PC ← (PC) + rel | REL | AD | rr | 6 | – | – | – | – | – |
| CLC | Clear carry bit | C ← 0 | INH | 98 | | 2 | – | – | – | – | 0 |
| CLI | Clear interrupt mask | I ← 0 | INH | 9A | | 2 | – | 0 | – | – | – |
| CLR opr | Clear register | M ← $00 | DIR | 3F | dd | 5 | – | – | 0 | 1 | – |
| CLRA | | A ← $00 | INH | 4F | | 3 | | | | | |
| CLRX | | X ← $00 | INH | 5F | | 3 | | | | | |
| CLR opr | | M ← $00 | IX1 | 6F | ff | 6 | | | | | |
| CLR opr | | M ← $00 | IX | 7F | | 5 | | | | | |
| CMP opr | Compare accumulator contents with memory contents | (A) – (M) | IMM | A1 | ii | 2 | – | – | ↕ | ↕ | ↕ |
| | | | DIR | B1 | dd | 3 | | | | | |
| | | | EXT | C1 | hh ll | 4 | | | | | |
| | | | IX2 | D1 | ee ff | 5 | | | | | |
| | | | IX1 | E1 | ff | 4 | | | | | |
| | | | IX | F1 | | 3 | | | | | |
| COM opr | Complement register contents | M ← M̄ = $FF – (M) | DIR | 33 | dd | 5 | – | – | ↕ | ↕ | 1 |
| COMA | (ones complement) | A ← Ā = $FF – (A) | INH | 43 | | 3 | | | | | |
| COMX | | X ← X̄ = $FF – (X) | INH | 53 | | 3 | | | | | |
| COM opr | | M ← M̄ = $FF – (M) | IX1 | 63 | ff | 6 | | | | | |
| COM opr | | M ← M̄ = $FF – (M) | IX | 73 | | 5 | | | | | |
| CPX opr | Compare index register contents with memory contents | (X) – (M) | IMM | A3 | ii | 2 | – | – | ↕ | ↕ | ↕ |
| | | | DIR | B3 | dd | 3 | | | | | |
| | | | EXT | C3 | hh ll | 4 | | | | | |
| | | | IX2 | D3 | ee ff | 5 | | | | | |
| | | | IX1 | E3 | ff | 4 | | | | | |
| | | | IX | F3 | | 3 | | | | | |
| DEC opr | Decrement register contents | M ← (M) – 1 | DIR | 3A | dd | 5 | – | – | ↕ | ↕ | – |
| DECA | | A ← (A) – 1 | INH | 4A | | 3 | | | | | |
| DECX | | X ← (X) – 1 | INH | 5A | | 3 | | | | | |
| DEC opr | | M ←(M) – 1 | IX1 | 6A | ff | 6 | | | | | |
| DEC opr | | M ←(M) – 1 | IX | 7A | | 5 | | | | | |

## Table 12-12. Instruction Set (Sheet 3 of 4)

| Source Form(s) | Operation | Description | Addressing Mode for Operand | Machine Coding (hexadecimal) | | Cycles | Condition Code | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Opcode | Operand | | H | I | N | Z | C |
| EOR opr | Exclusive OR accumulator contents with memory contents | A ← (A) ⊕ (M) | IMM | A8 | ii | 2 | – | – | ↕ | ↕ | – |
| | | | DIR | B8 | dd | 3 | | | | | |
| | | | EXT | C8 | hh ll | 4 | | | | | |
| | | | IX2 | D8 | ee ff | 5 | | | | | |
| | | | IX1 | E8 | ff | 4 | | | | | |
| | | | IX | F8 | | 3 | | | | | |
| INC opr | Increment memory or register contents | M ← (M) + 1 | DIR | 3C | dd | 5 | – | – | ↕ | ↕ | – |
| INCA | | A ← (A) + 1 | INH | 4C | | 3 | | | | | |
| INCX | | X ← (X) + 1 | INH | 5C | | 3 | | | | | |
| INC opr | | M ← (M) + 1 | IX1 | 6C | ff | 6 | | | | | |
| INC opr | | M ← (M) + 1 | IX | 7C | | 5 | | | | | |
| JMP opr | Unconditional jump | PC ← jump address | DIR | BC | dd | 2 | – | – | – | – | – |
| | | | EXT | CC | hh ll | 3 | | | | | |
| | | | IX2 | DC | ee ff | 4 | | | | | |
| | | | IX1 | EC | ff | 3 | | | | | |
| | | | IX | FC | | 2 | | | | | |
| JSR opr | Jump to subroutine | PC ← (PC) + n (n = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – 1<br>Push (PCH); SP ← (SP) – 1<br>PC ← conditional address | DIR | BD | dd | 5 | – | – | – | – | – |
| | | | EXT | CD | hh ll | 6 | | | | | |
| | | | IX2 | DD | ee ff | 7 | | | | | |
| | | | IX1 | ED | ff | 6 | | | | | |
| | | | IX | FD | | 5 | | | | | |
| LDA opr | Load accumulator with memory contents | A ← (M) | IMM | A6 | ii | 2 | – | – | ↕ | ↕ | – |
| | | | DIR | B6 | dd | 3 | | | | | |
| | | | EXT | C6 | hh ll | 4 | | | | | |
| | | | IX2 | D6 | ee ff | 5 | | | | | |
| | | | IX1 | E6 | ff | 4 | | | | | |
| | | | IX | F6 | | 3 | | | | | |
| LDX opr | Load index register with memory contents | X ← (M) | IMM | AE | ii | 2 | – | – | ↕ | ↕ | – |
| | | | DIR | BE | dd | 3 | | | | | |
| | | | EXT | CE | hh ll | 4 | | | | | |
| | | | IX2 | DE | ee ff | 5 | | | | | |
| | | | IX1 | EE | ff | 4 | | | | | |
| | | | IX | FE | | 3 | | | | | |
| LSL opr | Logical shift left |  | DIR | 38 | dd | 5 | – | – | ↕ | ↕ | ↕ |
| LSLA | | | INH | 48 | | 3 | | | | | |
| LSLX | | | INH | 58 | | 3 | | | | | |
| LSL opr | | | IX1 | 68 | ff | 6 | | | | | |
| LSL opr | | | IX | 78 | | 5 | | | | | |
| LSR opr | Logical shift right |  | DIR | 34 | dd | 5 | – | – | 0 | ↕ | ↕ |
| LSRA | | | INH | 44 | | 3 | | | | | |
| LSRX | | | INH | 54 | | 3 | | | | | |
| LSR opr | | | IX1 | 64 | ff | 6 | | | | | |
| LSR opr | | | IX | 74 | | 5 | | | | | |
| MUL | Unsigned multiply | X : A ← (X) × (A) | INH | 42 | | 11 | 0 | – | – | – | 0 |
| NEG opr | Negate memory or register contents (twos complement) | M ← –(M) = $00 – (M) | DIR | 30 | dd | 5 | – | – | ↕ | ↕ | ↕ |
| NEGA | | A ← –(A) = $00 – (A) | INH | 40 | | 3 | | | | | |
| NEGX | | X ← –(X) = $00 – (X) | INH | 50 | | 3 | | | | | |
| NEG opr | | M ← –(M) = $00 – (M) | IX1 | 60 | ff | 6 | | | | | |
| NEG opr | | M ← –(M) = $00 – (M) | IX | 70 | | 5 | | | | | |
| NOP | No operation | | INH | 9D | | 2 | – | – | – | – | – |
| ORA opr | Inclusive OR accumulator contents with memory contents | A ← (A) + (M) | IMM | AA | ii | 2 | – | – | ↕ | ↕ | – |
| | | | DIR | BA | dd | 3 | | | | | |
| | | | EXT | CA | hh ll | 4 | | | | | |
| | | | IX2 | DA | ee ff | 5 | | | | | |
| | | | IX1 | EA | ff | 4 | | | | | |
| | | | IX | FA | | 3 | | | | | |
| ROL opr | Rotate left through carry |  | DIR | 39 | dd | 5 | – | – | ↕ | ↕ | ↕ |
| ROLA | | | INH | 49 | | 3 | | | | | |
| ROLX | | | INH | 59 | | 3 | | | | | |
| ROL opr | | | IX1 | 69 | ff | 6 | | | | | |
| ROL opr | | | IX | 79 | | 5 | | | | | |

# Table 12-12. Instruction Set (Sheet 4 of 4)

| Source Form(s) | Operation | Description | Addressing Mode for Operand | Machine Coding (hexadecimal) | | Cycles | Condition Code | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Opcode | Operand | | H | I | N | Z | C |
| ROR opr | Rotate right through carry |  | DIR | 36 | dd | 5 | – | – | ↕ | ↕ | ↕ |
| RORA | | | INH | 46 | | 3 | | | | | |
| RORX | | | INH | 56 | | 3 | | | | | |
| ROR opr | | | IX1 | 66 | ff | 6 | | | | | |
| ROR opr | | | IX | 76 | | 5 | | | | | |
| RSP | Reset stack pointer | SP ← $00FF | INH | 9C | | 2 | – | – | – | – | – |
| RTI | Return from interrupt | SP ← (SP) + 1; pull (CCR) | INH | 80 | | 9 | From Stack | | | | |
| | | SP ← (SP) + 1; pull (A) | | | | | ↕ | ↕ | ↕ | ↕ | ↕ |
| | | SP ← (SP) + 1; pull (X) | | | | | | | | | |
| | | SP ← (SP) + 1; pull (PCH) | | | | | | | | | |
| | | SP ← (SP) + 1; pull (PCL) | | | | | | | | | |
| RTS | Return from subroutine | SP ← (SP) + 1; pull (PCH)<br>SP ← (SP) + 1; pull (PCL) | INH | 81 | | 6 | – | – | – | – | – |
| SBC opr | Subtract memory contents and carry bit from accumulator contents | A ← (A) – (M) – C | IMM | A2 | ii | 2 | – | – | ↕ | ↕ | ↕ |
| | | | DIR | B2 | dd | 3 | | | | | |
| | | | EXT | C2 | hh ll | 4 | | | | | |
| | | | IX2 | D2 | ee ff | 5 | | | | | |
| | | | IX1 | E2 | ff | 4 | | | | | |
| | | | IX | F2 | | 3 | | | | | |
| SEC | Set carry bit | C ← 1 | INH | 99 | | 2 | – | – | – | – | 1 |
| SEI | Set interrupt mask | I ← 1 | INH | 9B | | 2 | – | 1 | – | – | – |
| STA opr | Store accumulator contents in memory | M ← (A) | DIR | B7 | dd | 4 | – | – | ↕ | ↕ | – |
| | | | EXT | C7 | hh ll | 5 | | | | | |
| | | | IX2 | D7 | ee ff | 6 | | | | | |
| | | | IX1 | E7 | ff | 5 | | | | | |
| | | | IX | F7 | | 4 | | | | | |
| STOP | Enable $\overline{\text{IRQ}}$; stop oscillator | | INH | 8E | | 2 | – | 0 | – | – | – |
| STX opr | Store index register contents in memory | M ← (X) | DIR | BF | dd | 4 | – | – | ↕ | ↕ | – |
| | | | EXT | CF | hh ll | 5 | | | | | |
| | | | IX2 | DF | ee ff | 6 | | | | | |
| | | | IX1 | EF | ff | 5 | | | | | |
| | | | IX | FF | | 4 | | | | | |
| SUB opr | Subtract memory contents from accumulator contents | A ← (A) – (M) | IMM | A0 | ii | 2 | – | – | ↕ | ↕ | ↕ |
| | | | DIR | B0 | dd | 3 | | | | | |
| | | | EXT | C0 | hh ll | 4 | | | | | |
| | | | IX2 | D0 | ee ff | 5 | | | | | |
| | | | IX1 | E0 | ff | 4 | | | | | |
| | | | IX | F0 | | 3 | | | | | |
| SWI | Software interrupt | PC ← (PC) + 1; push (PCL)<br>SP ← (SP) – 1; push (PCH)<br>SP ← (SP) – 1; push (X)<br>SP ← (SP) – 1; push (A)<br>SP ← (SP) – 1; push (CCR)<br>SP ← (SP) – 1; I ← 1<br>PCH ← Interrupt vector hi byte<br>PCL ← Int. vector low byte | INH | 83 | | 10 | – | 1 | – | – | – |
| TAX | Transfer accumulator contents to index register | X ← (A) | INH | 97 | | 2 | – | – | – | – | – |
| TST opr | Test memory, accumulator, or index register contents for negative or zero | (M) – $00 | DIR | 3D | dd | 4 | – | – | ↕ | ↕ | – |
| TSTA | | | INH | 4D | | 3 | | | | | |
| TSTX | | | INH | 5D | | 3 | | | | | |
| TST opr | | | IX1 | 6D | ff | 5 | | | | | |
| TST opr | | | IX | 7D | | 4 | | | | | |
| TXA | Transfer index register contents to accumulator | A ← (X) | INH | 9F | | 2 | – | – | – | – | – |
| WAIT | Enable interrupts; halt CPU | | INH | 8F | | 2 | – | 0 | – | – | – |

## 12.4 Opcode Map

Table 12-13 is an opcode map of the M68HC05 instruction set.

### Table 12-13. Opcode Map

| | Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DIR | DIR | REL | DIR | INH | INH | IX1 | IX | INH | INH | IMM | DIR | EXT | IX2 | IX1 | IX | |
| HI<br>LO | 0<br>0000 | 1<br>0001 | 2<br>0010 | 3<br>0011 | 4<br>0100 | 5<br>0101 | 6<br>0110 | 7<br>0111 | 8<br>1000 | 9<br>1001 | A<br>1010 | B<br>1011 | C<br>1100 | D<br>1101 | E<br>1110 | F<br>1111 | HI<br>LO |
| 0<br>0000 | 5<br>BRSET0<br>3 DIR | 5<br>BSET0<br>2 DIR | 3<br>BRA<br>2 REL | 5<br>NEG<br>2 DIR | 3<br>NEGA<br>1 INH | | 6<br>NEG<br>2 IX1 | 5<br>NEG<br>1 IX | 9<br>RTI<br>1 INH | | 2<br>SUB<br>2 IMM | 3<br>SUB<br>2 DIR | 4<br>SUB<br>3 EXT | 5<br>SUB<br>3 IX2 | 4<br>SUB<br>2 IX1 | 3<br>SUB<br>1 IX | 0<br>0000 |
| 1<br>0001 | 5<br>BRCLR0<br>3 DIR | 5<br>BCLR0<br>2 DIR | 3<br>BRN<br>2 REL | | | | | | 6<br>RTS<br>1 INH | | 2<br>CMP<br>2 IMM | 3<br>CMP<br>2 DIR | 4<br>CMP<br>3 EXT | 5<br>CMP<br>3 IX2 | 4<br>CMP<br>2 IX1 | 3<br>CMP<br>1 IX | 1<br>0001 |
| 2<br>0010 | 5<br>BRSET1<br>3 DIR | 5<br>BSET1<br>2 DIR | 3<br>BHI<br>2 REL | | 11<br>MUL<br>1 INH | | | | | | 2<br>SBC<br>2 IMM | 3<br>SBC<br>2 DIR | 4<br>SBC<br>3 EXT | 5<br>SBC<br>3 IX2 | 4<br>SBC<br>2 IX1 | 3<br>SBC<br>1 IX | 2<br>0010 |
| 3<br>0011 | 5<br>BRCLR1<br>3 DIR | 5<br>BSCLR1<br>2 DIR | 3<br>BLS<br>2 REL | 5<br>COM<br>2 DIR | 3<br>COMA<br>1 INH | 3<br>COMX<br>1 INH | 6<br>COM<br>2 IX1 | 5<br>COM<br>1 IX | 10<br>SWI<br>1 INH | | 2<br>CPX<br>2 IMM | 3<br>CPX<br>2 DIR | 4<br>CPX<br>3 EXT | 5<br>CPX<br>3 IX2 | 4<br>CPX<br>2 IX1 | 3<br>CPX<br>1 IX | 3<br>0011 |
| 4<br>0100 | 5<br>BRSET2<br>3 DIR | 5<br>BSET2<br>2 DIR | 3<br>BCC<br>2 REL | 5<br>LSR<br>2 DIR | 3<br>LSRA<br>1 INH | 3<br>LSRX<br>1 INH | 6<br>LSR<br>2 IX1 | 5<br>LSR<br>1 IX | | | 2<br>AND<br>2 IMM | 3<br>AND<br>2 DIR | 4<br>AND<br>3 EXT | 5<br>AND<br>3 IX2 | 4<br>AND<br>2 IX1 | 3<br>AND<br>1 IX | 4<br>0100 |
| 5<br>0101 | 5<br>BRCLR2<br>3 DIR | 5<br>BCLR2<br>2 DIR | 3<br>BCS<br>2 REL | | | | | | | | 2<br>BIT<br>2 IMM | 3<br>BIT<br>2 DIR | 4<br>BIT<br>3 EXT | 5<br>BIT<br>3 IX2 | 4<br>BIT<br>2 IX1 | 3<br>BIT<br>1 IX | 5<br>0101 |
| 6<br>0110 | 5<br>BRSET3<br>3 DIR | 5<br>BSET3<br>2 DIR | 3<br>BNE<br>2 REL | 5<br>ROR<br>2 DIR | 3<br>RORA<br>1 INH | 3<br>RORX<br>1 INH | 6<br>ROR<br>2 IX1 | 5<br>ROR<br>1 IX | | | 2<br>LDA<br>2 IMM | 3<br>LDA<br>2 DIR | 4<br>LDA<br>3 EXT | 5<br>LDA<br>3 IX2 | 4<br>LDA<br>2 IX1 | 3<br>LDA<br>1 IX | 6<br>0110 |
| 7<br>0111 | 5<br>BRCLR3<br>3 DIR | 5<br>BCLR3<br>2 DIR | 3<br>BEQ<br>2 REL | 5<br>ASR<br>2 DIR | 3<br>ASRA<br>1 INH | 3<br>ASRX<br>1 INH | 6<br>ASR<br>2 IX1 | 5<br>ASR<br>1 IX | | 2<br>TAX<br>1 INH | | 4<br>STA<br>2 DIR | 5<br>STA<br>3 EXT | 6<br>STA<br>3 IX2 | 5<br>STA<br>2 IX1 | 4<br>STA<br>1 IX | 7<br>0111 |
| 8<br>1000 | 5<br>BRSET4<br>3 DIR | 5<br>BSET4<br>2 DIR | 3<br>BHCC<br>2 REL | 5<br>LSL<br>2 DIR | 3<br>LSLA<br>1 INH | 3<br>LSLX<br>1 INH | 6<br>LSL<br>2 IX1 | 5<br>LSL<br>1 IX | | 2<br>CLC<br>1 INH | 2<br>EOR<br>2 IMM | 3<br>EOR<br>2 DIR | 4<br>EOR<br>3 EXT | 5<br>EOR<br>3 IX2 | 4<br>EOR<br>2 IX1 | 3<br>EOR<br>1 IX | 8<br>1000 |
| 9<br>1001 | 5<br>BRCLR4<br>3 DIR | 5<br>BCLR4<br>2 DIR | 3<br>BHCS<br>2 REL | 5<br>ROL<br>2 DIR | 3<br>ROLA<br>1 INH | 3<br>ROLX<br>1 INH | 6<br>ROL | 5<br>ROL<br>1 IX | | 2<br>SEC<br>1 INH | 2<br>ADC<br>2 IMM | 3<br>ADC<br>2 DIR | 4<br>ADC<br>3 EXT | 5<br>ADC<br>3 IX2 | 4<br>ADC<br>2 IX1 | 3<br>ADC<br>1 IX | 9<br>1001 |
| A<br>1010 | 5<br>BRSET5<br>3 DIR | 5<br>BSET5<br>2 DIR | 3<br>BPL<br>2 REL | 5<br>DEC<br>2 DIR | 3<br>DECA<br>1 INH | 3<br>DECX<br>1 INH | 6<br>DEC<br>2 INH | 5<br>DEC<br>1 IX | | 2<br>CLI<br>1 INH | 2<br>ORA<br>2 IMM | 3<br>ORA<br>2 DIR | 4<br>ORA<br>3 EXT | 5<br>ORA<br>3 IX2 | 4<br>ORA<br>2 IX1 | 3<br>ORA<br>1 IX | A<br>1010 |
| B<br>1011 | 5<br>BRCLR5<br>3 DIR | 5<br>BCLR5<br>2 DIR | 3<br>BMI<br>2 REL | | | | | | | 2<br>SEI<br>1 INH | 2<br>ADD<br>2 IMM | 3<br>ADD<br>2 DIR | 4<br>ADD<br>3 EXT | 5<br>ADD<br>3 IX2 | 4<br>ADD<br>2 IX1 | 3<br>ADD<br>1 IX | B<br>1011 |
| C<br>1100 | 5<br>BRSET6<br>3 DIR | 5<br>BSET6<br>2 DIR | 3<br>BMC<br>2 REL | 5<br>INC<br>2 DIR | 3<br>INCA<br>1 INH | 3<br>INCX<br>1 INH | 6<br>INC<br>2 INH | 5<br>INC<br>1 IX | | 2<br>RSP<br>1 INH | | 2<br>JMP<br>2 DIR | 3<br>JMP<br>3 EXT | 4<br>JMP<br>3 IX2 | 3<br>JMP<br>2 IX1 | 2<br>JMP<br>1 IX | C<br>1100 |
| D<br>1101 | 5<br>BRCLR6<br>3 DIR | 5<br>BCLR6<br>2 DIR | 3<br>BMS<br>2 REL | 4<br>TST<br>2 DIR | 3<br>TSTA<br>1 INH | 3<br>TSTX<br>1 INH | 6<br>TST<br>2 INH | 4<br>TST<br>1 IX | | 2<br>NOP<br>1 INH | 6<br>BSR<br>2 REL | 5<br>JSR<br>2 DIR | 6<br>JSR<br>3 EXT | 7<br>JSR<br>3 IX2 | 6<br>JSR<br>2 IX1 | 5<br>JSR<br>1 IX | D<br>1101 |
| E<br>1110 | 5<br>BRSET7<br>3 DIR | 5<br>BSET7<br>2 DIR | 3<br>BIL<br>2 REL | | | | | | 2<br>STOP<br>1 INH | | 2<br>LDX<br>2 IMM | 3<br>LDX<br>2 DIR | 4<br>LDX<br>3 EXT | 5<br>LDX<br>3 IX2 | 4<br>LDX<br>2 IX1 | 3<br>LDX<br>1 IX | E<br>1110 |
| F<br>1111 | 5<br>BRCLR7<br>3 DIR | 5<br>BCLR7<br>2 DIR | 3<br>BIH<br>2 REL | 5<br>CLR<br>2 DIR | 3<br>CLRA<br>1 INH | 3<br>CLRX<br>1 INH | 6<br>CLR<br>2 INH | 5<br>CLR<br>1 IX | 2<br>WAIT<br>1 INH | 2<br>TXA<br>1 INH | | 4<br>STX<br>2 DIR | 5<br>STX<br>3 EXT | 6<br>STX<br>3 IX2 | 5<br>STX<br>2 IX1 | 4<br>STX<br>1 IX | F<br>1111 |

**ABBREVIATIONS FOR ADDRESSING MODES**

| | | | |
|---|---|---|---|
| INH | Inherent | REL | Relative |
| IMM | Immediate | IX | Indexed, No Offset |
| DIR | Direct | IX1 | Indexed, 8-Bit Offset |
| EXT | Extended | IX2 | Indexed, 16-Bit Offset |

**LEGEND**

| | | | |
|---|---|---|---|
| | F<br>1111 | | High Byte of Opcode in Hexadecimal<br>High Byte of Opcode in Binary |
| Number of Cycles<br>Opcode Mnemonic<br>Number of Bytes/Addressing Mode | 3<br>SUB<br>1 IX | 0<br>0000 | Low Byte of Opcode in Hexadecimal<br>Low Byte of Opcode in Binary |

# SECTION 13
# ELECTRICAL SPECIFICATIONS

This section contains MCU electrical specifications and timing information.

## 13.1 Maximum Ratings

The MCU contains circuitry that protects the inputs against damage from high static voltages; however, do not apply voltages higher than those shown in Table 13-1. Keep $V_{IN}$ and $V_{OUT}$ within the range $V_{SS} \leq (V_{IN}$ or $V_{OUT}) \leq V_{DD}$. Connect unused inputs to the appropriate logic level, either $V_{SS}$ or $V_{DD}$.

### Table 13-1. Maximum Ratings

| Rating | Symbol | Value | Unit |
|---|---|---|---|
| Supply Voltage | $V_{DD}$ | –0.3 to +7.0 | V |
| Input Voltage | $V_{IN}$ | $V_{SS}$ – 0.3 to $V_{DD}$ + 0.3 | V |
| EPROM Programming Voltage ($\overline{IRQ}$/$V_{PP}$ Pin) | $V_{PP}$ | 16.75 | V |
| Current Drain per Pin (Excluding $V_{DD}$ and $V_{SS}$) | I | 25 | mA |
| Operating Temperature Range | $T_A$ |  | °C |
| MC68HC705P9P, DW, S (Standard) |  | 0 to +70 |  |
| MC68HC705P9CP, CDW, CS (Extended) |  | –40 to +85 |  |
| MC68HC705P9VP, VDW, VS (Automotive) |  | –40 to +105 |  |
| MC68HC705P9MP, MDW, MS (Automotive) |  | –40 to +125 |  |
| Storage Temperature Range | $T_{STG}$ | –65 to +150 | °C |

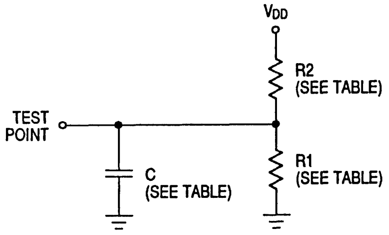
NOTES:

1. P = Plastic dual in-line package (PDIP)
2. DW = Small outline integrated circuit (SOIC)
3. S = Ceramic dual in-line package (Cerdip)
4. C = Extended temperature range (–40 to +85 °C)
5. V = Automotive temperature range (–40 to +105 °C)
6. M = Automotive temperature range (–40 to +125 °C)

## 13.2 Thermal Characteristics

### Table 13-2. Thermal Resistance

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Thermal Resistance | $R_{\theta JA}$ |  | °C/W |
| Plastic |  | 60 |  |
| SOIC |  | 60 |  |



$V_{DD}$ = 4.5 V

| Pins | R1 | R2 | C |
|---|---|---|---|
| PA7–PA0<br>PB7–PB5<br>PC7–PC0 | 3.26 kΩ | 2.38 kΩ | 50 pF |

Figure 13-1. Test Load

### 13.3 Power Considerations

The average chip junction temperature, $T_J$, in °C can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \qquad (1)$$

where:

$T_A$ = Ambient temperature in °C
$\theta_{JA}$ = Package thermal resistance, junction to ambient in °C/W
$P_D = P_{INT} + P_{I/O}$
$P_{INT} = I_{CC} \times V_{CC}$, watts — chip internal power
$P_{I/O}$ = Power dissipation on input and output pins — user-determined

For most applications $P_{I/O} \ll P_{INT}$ and can be neglected.
The following is an approximate relationship between $P_D$ and $T_J$ (if $P_{I/O}$ is neglected):

$$P_D = K \div (T_J + 273\ ^\circ C) \qquad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_A \times (T_A + 273\ ^\circ C) + R_{\theta JA} \times P_D \qquad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring $P_D$ (at equilibrium) for a known $T_A$. Using this value of K, the values of $P_D$ and $T_J$ can be obtained by solving equations (1) and (2) iteratively for any value of $T_A$.

## 13.4 DC Electrical Characteristics ($V_{DD}$ = 5.0 Vdc)

### Table 13-3. DC Electrical Characteristics ($V_{DD}$ = 5.0 Vdc)

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Output Voltage<br>$I_{LOAD}$ = 10.0 µA<br>$I_{LOAD}$ = –10.0 µA | <br>$V_{OL}$<br>$V_{OH}$ | <br>—<br>$V_{DD}$ – 0.1 | <br>—<br>— | <br>0.1<br>— | <br>V<br>V |
| Output High Voltage ($I_{LOAD}$ = –0.8 mA)<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, TCMP | $V_{OH}$ | $V_{DD}$ – 0.8 | — | — | V |
| Output Low Voltage ($I_{LOAD}$ = 1.6 mA)<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, TCMP | $V_{OL}$ | — | — | 0.4 | V |
| Input High Voltage<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, PD7/TCAP, $\overline{\text{IRQ}}$/$V_{PP}$, $\overline{\text{RESET}}$, OSC1 | $V_{IH}$ | 0.7 × $V_{DD}$ | — | $V_{DD}$ | V |
| Input Low Voltage<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, PD7/TCAP, $\overline{\text{IRQ}}$/$V_{PP}$, $\overline{\text{RESET}}$, OSC1 | $V_{IL}$ | $V_{SS}$ | — | 0.2 × $V_{DD}$ | V |
| Supply Current (NOTES 3–6)<br>RUN<br>WAIT (A/D Converter On)<br>WAIT (A/D Converter Off)<br>STOP<br>25 °C<br>0 to 70 °C (Standard)<br>–40 to 125 °C | $I_{DD}$ | <br>—<br>—<br>—<br><br>—<br>—<br>— | <br>4.7<br>2.1<br>1.3<br><br>2<br>—<br>— | <br>6.5<br>2.9<br>1.9<br><br>30<br>50<br>100 | <br>mA<br>mA<br>mA<br><br>µA<br>µA<br>µA |
| I/O Ports Hi-Z Leakage Current<br>PA7–PA0, PB7–PB5, PC7–PC0, PD5 | $I_{IL}$ | — | — | ±10 | µA |
| A/D Ports Hi-Z Leakage Current | $I_{OZ}$ | — | — | ±1 | µA |
| Input Current<br>$\overline{\text{RESET}}$,$\overline{\text{IRQ}}$/$V_{PP}$, OSC1,PD7/TCAP | $I_{IN}$ | — | — | ±1 | µA |
| Capacitance<br>Ports (As Input or Output)<br>$\overline{\text{RESET}}$, $\overline{\text{IRQ}}$/$V_{PP}$ | <br>$C_{OUT}$<br>$C_{IN}$ | <br>—<br>— | <br>—<br>— | <br>12<br>8 | <br>pF<br>pF |
| Programming Voltage | $V_{PP}$ | 16.25 | 16.5 | 16.75 | V |
| Programming Current | $I_{PP}$ | — | 5 | 10 | mA |
| Programming Time per Byte | $t_{EPGM}$ | 4 | — | — | ms |

NOTES:

1. $V_{DD}$ = 5.0 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. Typical values at midpoint of voltage range, 25 °C only.
3. RUN (operating) $I_{DD}$ and WAIT $I_{DD}$ measured using external square wave clock source ($f_{OSC}$ = 4.2 MHz); all inputs 0.2 V from rail; no dc loads; less than 50 pF on all outputs; $C_L$ = 20 pF on OSC2.
4. WAIT $I_{DD}$ and STOP $I_{DD}$: all ports configured as inputs; $V_{IL}$ = 0.2 V; $V_{IH}$ = $V_{DD}$ – 0.2 V.
5. STOP $I_{DD}$ measured with OSC1 = $V_{SS}$.
6. WAIT $I_{DD}$ is affected linearly by the OSC2 capacitance.

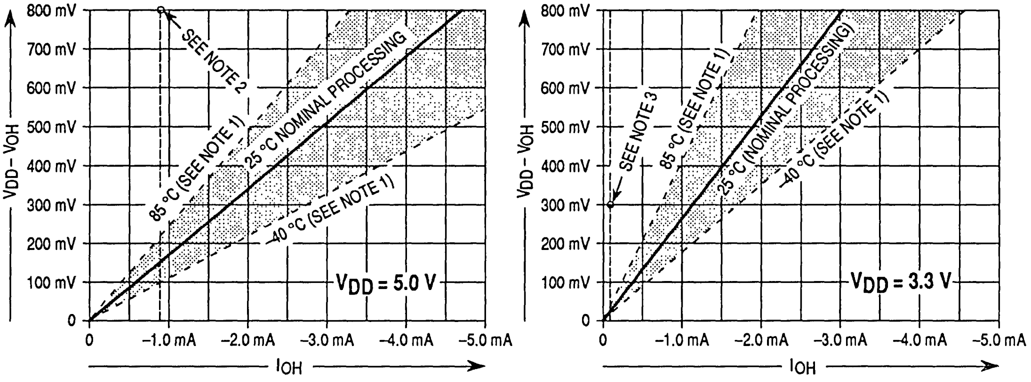
## 13.5 DC Electrical Characteristics ($V_{DD}$ = 3.3 Vdc)

### Table 13-4. DC Electrical Characteristics ($V_{DD}$ = 3.3 Vdc)

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Output Voltage ($I_{LOAD} \leq$ 10.0 µA) | $V_{OL}$<br>$V_{OH}$ | —<br>$V_{DD}$ – 0.1 | —<br>— | 0.1<br>— | <br>V |
| Output High Voltage ($I_{LOAD}$ = –0.2 mA)<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, TCMP | $V_{OH}$ | $V_{DD}$ – 0.3 | — | — | V |
| Output Low Voltage ($I_{LOAD}$ = 0.4 mA)<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, TCMP | $V_{OL}$ | — | — | 0.3 | V |
| Input High Voltage<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, PD7/TCAP, $\overline{\text{IRQ}}$/$V_{PP}$, $\overline{\text{RESET}}$, OSC1 | $V_{IH}$ | 0.7 × $V_{DD}$ | — | $V_{DD}$ | V |
| Input Low Voltage<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5, PD7/TCAP, $\overline{\text{IRQ}}$/$V_{PP}$, $\overline{\text{RESET}}$, OSC1 | $V_{IL}$ | $V_{SS}$ | — | 0.2 × $V_{DD}$ | V |
| Data-Retention Mode Supply Voltage | $V_{RM}$ | 2.0 | — | — | V |
| Supply Current (NOTES 3–6)<br>RUN<br>WAIT(A/D Converter On)<br>WAIT (A/D Converter Off)<br>STOP<br>25 °C<br>0 to 70 °C (Standard)<br>–40 to +125 °C | $I_{DD}$ | <br>—<br>—<br><br>—<br>—<br>—<br>— | <br>1.6<br>0.9<br>0.4<br><br>1.0<br>—<br>— | <br>2.3<br>1.3<br>0.6<br><br>20<br>40<br>50 | <br>mA<br>mA<br>mA<br><br>µA<br>µA<br>µA |
| I/O Ports Hi-Z Leakage Current<br>PA7–PA0, PB7/SCK–PB5/SDO, PC7/$V_{RH}$–PC3/AN3,<br>PC2–PC0, PD5 | $I_{IL}$ | — | — | ±10 | µA |
| Input Current<br>$\overline{\text{RESET}}$, $\overline{\text{IRQ}}$/$V_{PP}$, OSC1, PD5, PD7/TCAP | $I_{IN}$ | — | — | ±1 | µA |
| Capacitance<br>Ports (As Input or Output)<br>$\overline{\text{RESET}}$, $\overline{\text{IRQ}}$/$V_{PP}$, PD5, PD7/TCAP | <br>$C_{OUT}$<br>$C_{IN}$ | <br>—<br>— | <br>—<br>— | <br>12<br>8 | <br>pF<br>pF |

NOTES:

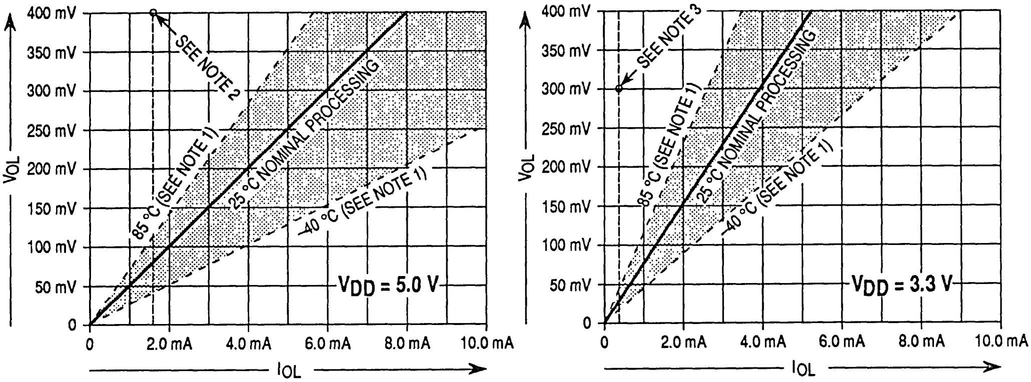
1. $V_{DD}$ = 3.3 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$ unless otherwise noted.
2. Typical values at midpoint of voltage range, 25 °C only.
3. RUN (operating) $I_{DD}$ and WAIT $I_{DD}$ measured using external square wave clock source ($f_{OSC}$ = 2.1 MHz); all inputs 0.2 V from rail; no dc loads; less than 50 pF on all outputs; $C_L$ = 20 pF on OSC2.
4. WAIT $I_{DD}$ and STOP $I_{DD}$: all ports configured as inputs; $V_{IL}$ = 0.2 V, $V_{IH}$ = $V_{DD}$ – 0.2 V.
5. STOP $I_{DD}$ measured with OSC1 = $V_{SS}$.
6. WAIT $I_{DD}$ is affected linearly by the OSC2 capacitance.



NOTES:

1. Shaded area indicates variation in driver characteristics due to changes in temperature and for normal processing tolerances. Within the limited range of values shown, V vs I curves are approximately straight lines.
2. At $V_{DD}$ = 5.0 V, devices are specified and tested for ($V_{DD}$ – $V_{OH}$) ≤ 800 mV @ $I_{OH}$= –0.8 mA.
3. At $V_{DD}$ = 3.3 V, devices are specified and tested for ($V_{DD}$ – $V_{OH}$) ≤ 300 mV @ $I_{OH}$= –0.2 mA.

Figure 13-2. Typical High-Side Driver Characteristics



NOTES:

1. Shaded area indicates variation in driver characteristics due to changes in temperature and for normal processing tolerances. Within the limited range of values shown, V vs I curves are approximately straight lines.
2. At $V_{DD}$ = 5.0 V, devices are specified and tested for $V_{OL}$ ≤ 400 mV @ $I_{OL}$ = 1.6 mA.
3. At $V_{DD}$ = 3.3 V, devices are specified and tested for $V_{OL}$ ≤ 300 mV @ $I_{OL}$ = 0.4 mA.

Figure 13-3. Typical Low-Side Driver Characteristics

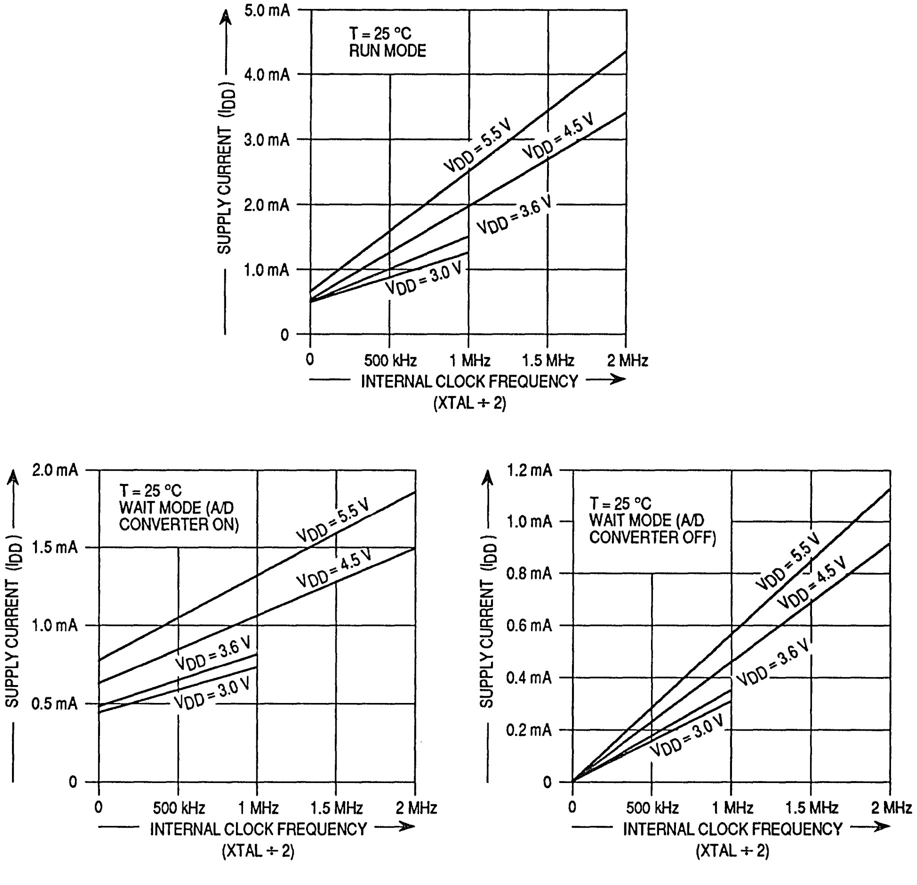


**Figure 13-4. Typical Supply Current vs Internal Clock Frequency**

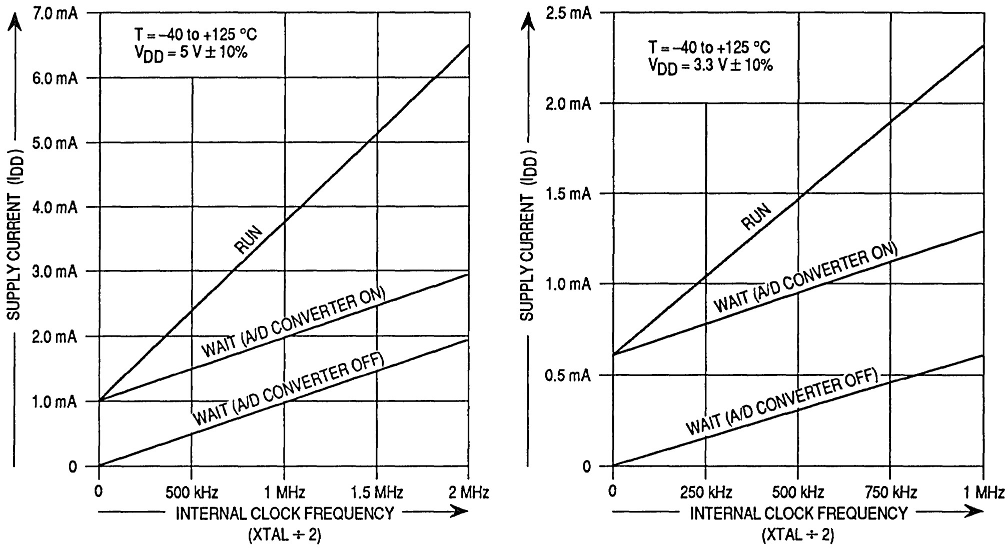


**Figure 13-5. Maximum Supply Current vs Internal Clock Frequency**

## 13.6 A/D Converter Characteristics

### Table 13-5. A/D Converter Characteristics

| Characteristic | Min | Max | Unit |
|---|---|---|---|
| Resolution | 8 | 8 | Bit |
| Absolute Accuracy (4.0 > $V_{RH}$ > $V_{DD}$) (NOTE 2) | — | ±1−1/2 | LSB |
| Conversion Range ($V_{RH}$ Pin) | $V_{SS}$ | $V_{DD}$ | V |
| Conversion Time (Includes Sampling Time)<br>External Clock (XTAL)<br>Internal RC Oscillator (ADRC = 1) | <br>32<br>32 | <br>32<br>32 | <br>$t_{AD}$<br>µs |
| Monotonicity | Inherent (Within Total Error) | | |
| Zero Input Reading ($V_{IN}$ = 0 V) | 00 | 01 | Hex |
| Full-Scale Reading ($V_{IN}$ = $V_{RH}$) | FF | FF | Hex |
| Sample Acquisition Time (NOTE 3)<br>External Clock (XTAL)<br>Internal RC Oscillator (ADRC) = 1 | <br>12<br>— | <br>12<br>12 | <br>$t_{AD}$<br>µs |
| Input Capacitance<br>PC6/AN0, PC5/AN1, PC4/AN2, PC3/AN3 | <br>— | <br>12 | <br>pF |
| Analog Input Voltage | $V_{SS}$ | $V_{RH}$ | V |
| Input Leakage (NOTE 5)<br>PC6/AN0, PC5/AN1, PC4/AN2, PC3/AN3<br>$V_{RH}$ | <br>—<br>— | <br>±1<br>±1 | <br>µA<br>µA |

NOTES:

1. $V_{DD}$ = 5.0 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. A/D accuracy may decrease proportionately as $V_{RH}$ is reduced below 4.0 V.
3. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.
4. $t_{AD}$ = $t_{CYC}$ if clock source is MCU.
5. The external system error caused by input leakage current is approximately equal to the product of R source and input current.

## 13.7 Control Timing ($V_{DD}$ = 5.0 Vdc)

### Table 13-6. Control Timing ($V_{DD}$ = 5.0 Vdc)

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Oscillator Frequency | $f_{OSC}$ | | | |
| Crystal Option | | — | 4.2 | MHz |
| External Clock Option | | dc | 4.2 | MHz |
| Internal Operating Frequency | $f_{OP}$ | | | |
| Crystal ($f_{OSC}$ ÷ 2) | | — | 2.1 | MHz |
| External Clock ($f_{OSC}$ ÷ 2) | | dc | 2.1 | MHz |
| Cycle Time | $t_{CYC}$ | 480 | – | ns |
| Crystal Oscillator Startup Time | $t_{OXOV}$ | — | 100 | ms |
| STOP Recovery Startup Time (Crystal Oscillator) | $t_{ILCH}$ | — | 100 | ms |
| $\overline{\text{RESET}}$ Pulse Width | $t_{RL}$ | 1.5 | — | $t_{CYC}$ |
| Timer Resolution (NOTE 2) | $t_{RESL}$ | 4.0 | — | $t_{CYC}$ |
| Interrupt Pulse Width Low (Edge-Triggered) | $t_{ILIH}$ | 125 | — | ns |
| Interrupt Pulse Period | $t_{ILIL}$ | (NOTE 3) | — | $t_{CYC}$ |
| OSC1 Pulse Width | $t_{OH}$, $t_{OL}$ | 90 | — | ns |
| RC Oscillator Stabilization Time | $t_{RCON}$ | — | 5 | µs |
| $V_{DD}$ Slew Rate | | | | V/µs |
| Rising | $S_{VDDR}$ | — | 0.05 | |
| Falling | $S_{VDDR}$ | — | 0.1 | |
| A/D On Current Stabilization Time | $t_{ADON}$ | — | 100 | µs |

NOTES:

1. $V_{DD}$ = 5.0 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. Since a 2-bit prescaler in the timer must count four internal cycles ($t_{CYC}$), this is the limiting minimum factor in determining the timer resolution.
3. The minimum period $t_{ILIL}$ should not be less than the number of cycle times it takes to execute the interrupt service routine plus 19 $t_{CYC}$.

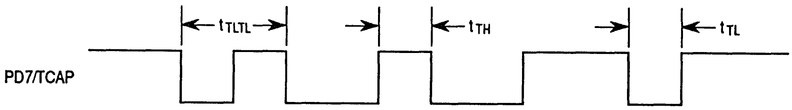
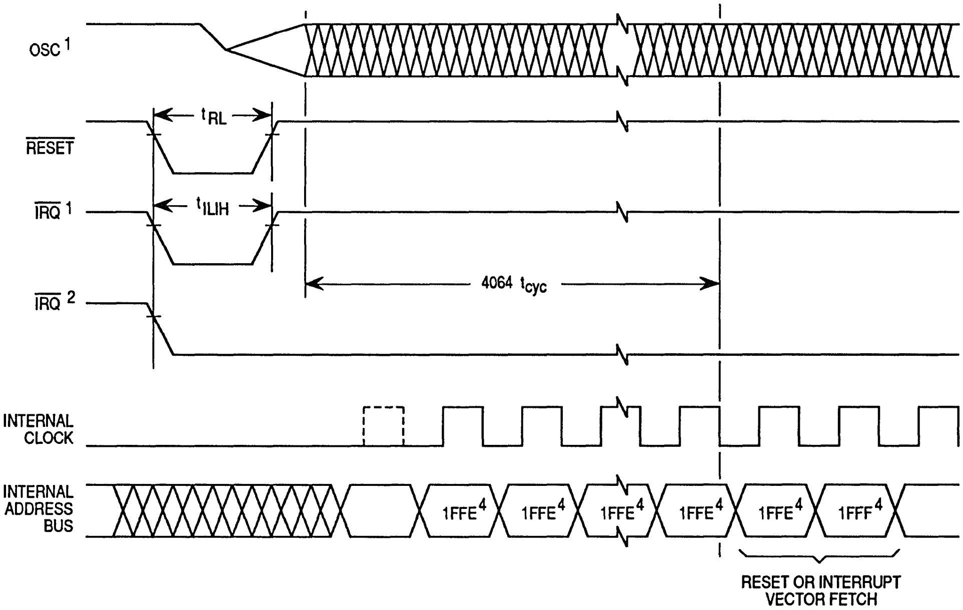


**Figure 13-6. TCAP Timing**



NOTES:

1. Represents the internal clocking of OSC1 pin.
2. External interrupt edge-triggered mask option.
3. External interrupt edge- and level-triggered mask option.
4. Reset vector shown for timing example.

**Figure 13-7. STOP Recovery Timing**

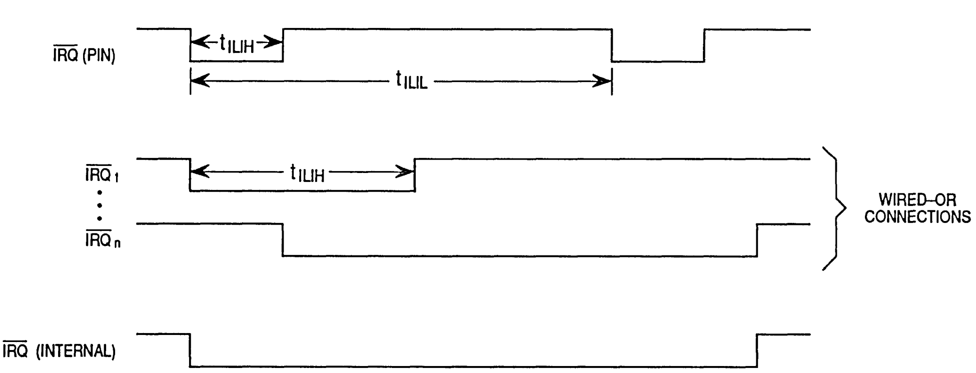


Figure 13-8. External Interrupt Timing

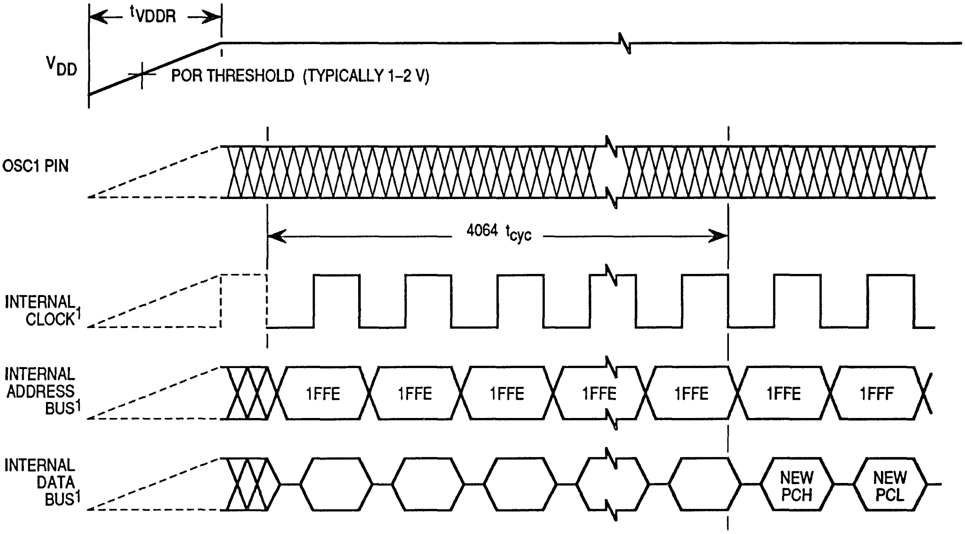
## 13.8 Control Timing ($V_{DD}$ = 3.3 Vdc)

Table 13-7. Control Timing ($V_{DD}$ = 3.3 Vdc)

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Oscillator Frequency<br>Crystal Option<br>External Clock Option | $f_{OSC}$ | <br>–<br>dc | <br>2.0<br>2.0 | <br>MHz<br>MHz |
| Internal Operating Frequency<br>Crystal ($f_{OSC}$ ÷ 2)<br>External Clock ($f_{OSC}$ ÷ 2) | $f_{OP}$ | <br>–<br>dc | <br>1.0<br>1.0 | <br>MHz<br>MHz |
| Cycle Time | $t_{CYC}$ | 1 | – | ms |
| Crystal Oscillator Startup Time | $t_{OXOV}$ | – | 100 | ms |
| STOP Recovery Startup Time (Crystal Oscillator) | $t_{ILCH}$ | – | 100 | ms |
| RESET Pulse Width | $t_{RL}$ | 1.5 | – | $t_{CYC}$ |
| Timer Resolution (NOTE 2) | $t_{RESL}$ | 4.0 | – | $t_{CYC}$ |
| Interrupt Pulse Width Low (Edge-Triggered) | $t_{ILIH}$ | 250 | – | ns |
| Interrupt Pulse Period | $t_{ILIL}$ | (NOTE 3) | – | $t_{CYC}$ |
| OSC1 Pulse Width | $t_{OH}$, $t_{OL}$ | 200 | – | ns |

NOTES:

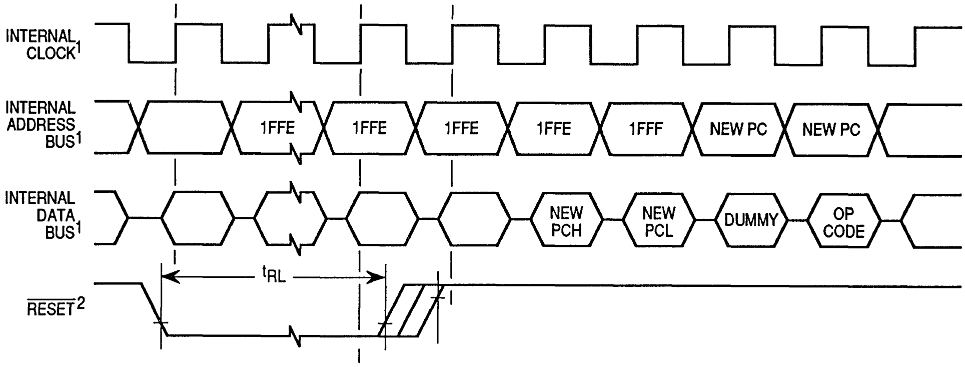
1. $V_{DD}$ = 3.3 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. Because a 2-bit prescaler in the timer must count four internal cycles ($t_{CYC}$), this is the limiting minimum factor in determining the timer resolution.
3. The minimum period $t_{ILIL}$ should not be less than the number of cycle times it takes to execute the interrupt service routine plus 19 $t_{CYC}$.



NOTES:

1. Internal clock, internal address bus, and internal data bus are not available externally.

**Figure 13-9. Power-On Reset Timing**



NOTES:

1. Internal clock, internal address bus, and internal data bus signals are not available externally.
2. Next rising edge of internal clock after rising edge of $\overline{\text{RESET}}$ initiates reset sequence.

**Figure 13-10. External Reset Timing**

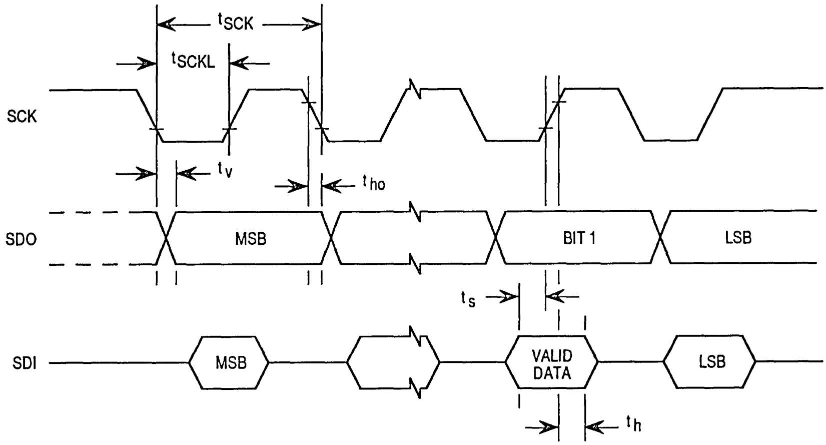
## 13.9 SIOP Timing (V$_{DD}$ = 5.0 Vdc)

**Table 13-8. SIOP Timing (V$_{DD}$ = 5.0 Vdc)**

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Frequency of Operation<br>Master<br>Slave | <br>$f_{SIOP(M)}$<br>$f_{SIOP(S)}$ | <br>0.25<br>dc | <br>0.25<br>525 | <br>$f_{OP}$<br>kHz |
| Cycle time<br>Master<br>Slave | <br>$t_{SCK(M)}$<br>$t_{SCK(S)}$ | <br>4.0<br>— | <br>4.0<br>1920 | <br>$t_{CYC}$<br>ns |
| Clock (SCK) Low Time ($f_{OP}$ = 2.1 MHz) | $t_{SCKL}$ | 932 | — | ns |
| SDO Data Valid Time | $t_V$ | — | 200 | ns |
| SDO Hold Time | $t_{HO}$ | 0 | — | ns |
| SDI Setup Time | $t_S$ | 100 | — | ns |
| SDI Hold Time | $t_H$ | 100 | — | ns |

NOTES:

1. $V_{DD}$ = 5.0 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. $f_{OP} = f_{OSC} \div 2$ = 2.1 MHz maximum; $t_{CYC} = 1 \div f_{OP}$.
3. In master mode, SCK is generated by dividing the internal clock ($f_{OP}$) by 4.



NOTES:

1. This diagram applies to both the master and slave modes of the SIOP.
2. Bit order is shown for MSB first option.

**Figure 13-11. SIOP Timing**

## 13.10 SIOP Timing ($V_{DD}$ = 3.3 Vdc)

Table 13-9. SIOP Timing ($V_{DD}$ = 3.3 Vdc)

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Frequency of operation<br>Master<br>Slave | <br>$f_{SIOP(M)}$<br>$f_{SIOP(S)}$ | <br>0.25<br>dc | <br>0.25<br>250 | <br>$f_{OP}$<br>kHz |
| Cycle time<br>Master<br>Slave | <br>$t_{SCK(M)}$<br>$t_{SCK(S)}$ | <br>4.0<br>– | <br>4.0<br>4000 | <br>$t_{CYC}$<br>ns |
| Clock (SCK) Low Time ($f_{OP}$ = 1.0 MHz) | $t_{SCKL}$ | 1980 | – | ns |
| SDO Data Valid Time | $t_V$ | – | 400 | ns |
| SDO Hold Time | $t_{HO}$ | 0 | – | ns |
| SDI Setup Time | $t_S$ | 200 | – | ns |
| SDI Hold Time | $t_H$ | 200 | – | ns |

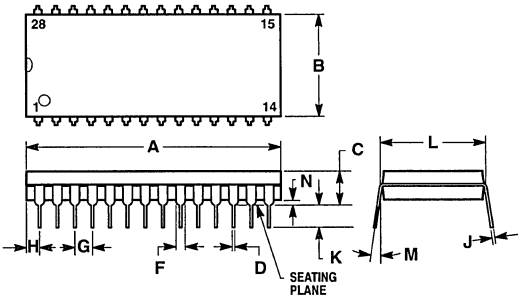
NOTES:

1. $V_{DD}$ = 3.3 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. $f_{OP}$ = 1.0 MHz maximum.

# SECTION 14
# MECHANICAL SPECIFICATIONS

This section gives the dimensions of the plastic dual in-line package (PDIP), ceramic dual in-line (cerdip), and small outline integrated circuit (SOIC) packages.

## 14.1 PDIP

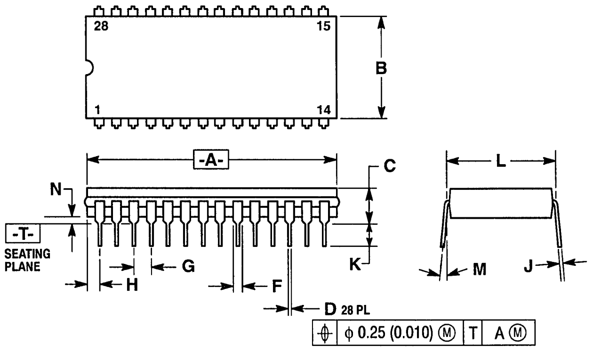


NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.
4. 710-01 OBSOLETE, NEW STANDARD 710-02.

| | MILLIMETERS | | INCHES | |
|---|---|---|---|---|
| DIM | MIN | MAX | MIN | MAX |
| A | 36.45 | 37.21 | 1.435 | 1.465 |
| B | 13.72 | 14.22 | 0.540 | 0.560 |
| C | 3.94 | 5.08 | 0.155 | 0.200 |
| D | 0.36 | 0.56 | 0.014 | 0.022 |
| F | 1.02 | 1.52 | 0.040 | 0.060 |
| G | 2.54 BSC | | 0.100 BSC | |
| H | 1.65 | 2.16 | 0.065 | 0.085 |
| J | 0.20 | 0.38 | 0.008 | 0.015 |
| K | 2.92 | 3.43 | 0.115 | 0.135 |
| L | 15.24 BSC | | 0.600 BSC | |
| M | 0° | 15° | 0° | 15° |
| N | 0.51 | 1.02 | 0.020 | 0.040 |

**Figure 14-1. MC68HC705P9P (Case 710-02)**

## 14.2 Cerdip

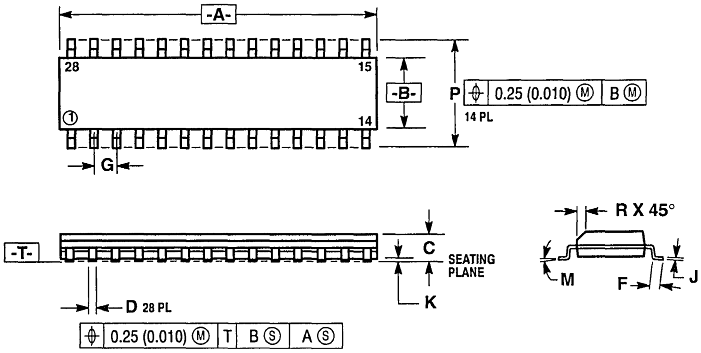


NOTES:
1. DIM A AND B INCLUDES MENISCUS.
2. DIM -L- TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1982.
4. CONTROLLING DIM: INCH.
5. 733-03 OBSOLETE, NEW STANDARD 733-04.

| | MILLIMETERS | | INCHES | |
|---|---|---|---|---|
| DIM | MIN | MAX | MIN | MAX |
| A | 36.45 | 37.84 | 1.435 | 1.490 |
| B | 12.70 | 15.36 | 0.500 | 0.605 |
| C | 4.06 | 5.84 | 0.160 | 0.230 |
| D | 0.38 | 0.55 | 0.015 | 0.022 |
| F | 1.27 | 1.65 | 0.050 | 0.065 |
| G | 2.54 BSC | | 0.100 BSC | |
| J | 0.20 | 0.30 | 0.008 | 0.012 |
| K | 3.18 | 4.06 | 0.125 | 0.160 |
| L | 15.24 BSC | | 0.600 BSC | |
| M | 0° | 15° | 0° | 15° |
| N | 0.51 | 1.27 | 0.020 | 0.050 |

**Figure 14-2. MC68HC705P9S (Case 733-04)**

## 14.3 SOIC



NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.
5. 751F-01 AND -02 OBSOLETE, NEW STANDARD 751F-03.

| | MILLIMETERS | | INCHES | |
|---|---|---|---|---|
| DIM | MIN | MAX | MIN | MAX |
| A | 17.80 | 18.05 | 0.701 | 0.711 |
| B | 7.40 | 7.60 | 0.292 | 0.299 |
| C | 2.35 | 2.65 | 0.093 | 0.104 |
| D | 0.35 | 0.49 | 0.014 | 0.019 |
| F | 0.41 | 0.90 | 0.016 | 0.035 |
| G | 1.27 BSC | | 0.050 BSC | |
| J | 0.229 | 0.317 | 0.0090 | 0.0125 |
| K | 0.127 | 0.292 | 0.0050 | 0.0115 |
| M | 0° | 8° | 0° | 8° |
| P | 10.05 | 10.55 | 0.395 | 0.415 |
| R | 0.25 | 0.75 | 0.010 | 0.029 |

**Figure 14-3. MC68HC709DW (Case 751F-02)**

**Literature Distribution Centers:**

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong.

MOTOROLA

1ATX26859–1 PRINTED IN USA 8/92 IMPERIAL LITHO 86801 18,000 MCU YGACAA