MC68HC05J1A/D

HC05

MC68HC05J1A

TECHNICAL
DATA

MOTOROLA

TABLE OF CONTENTS

Paragraph **Title** **Page**

SECTION 1
GENERAL DESCRIPTION

SECTION 2
MEMORY

SECTION 3
CENTRAL PROCESSOR UNIT (CPU)

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

Paragraph **Title** **Page**

SECTION 11
MECHANICAL SPECIFICATIONS

SECTION 12
ORDERING INFORMATION

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

SECTION 1
GENERAL DESCRIPTION

The MC68HC05J1A 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 MC68HC05J1A includes 1240 bytes of user ROM and 64 bytes of user RAM.

1.1 Features

Features of the MCU include the following:

- Popular M68HC05 CPU
- Memory-Mapped Input/Output (I/O) Registers
- 1240 Bytes of User ROM Including 8 User Vector Locations
- 64 Bytes of User RAM
- 14 Bidirectional I/O Pins with the following features:
  - Software Programmable Pulldown Devices
  - 4 I/O Pins with 8 mA Current Sinking Capability
  - 4 I/O Pins with Maskable External Interrupt Capability
- Hardware Mask and Flag for External Interrupts
- Fully Static Operation with no Minimum Clock Speed
- On-Chip Oscillator with Connections for a Crystal or Ceramic Resonator or for a Resistor-Capacitor Network
- 15-Bit Multifunction Timer
- Computer Operating Properly (COP) Watchdog

- Power-Saving Stop (or Halt), Wait, and Data-Retention Modes
- Illegal Address Reset
- Internal Steering Diode Between $\overline{\text{RESET}}$ and V_{DD} pins
- 8×8 Unsigned Multiply Instruction
- 20-Pin Plastic Dual In-Line Package (PDIP)
- 20-Pin Small Outline Integrated Circuit Package (SOIC)

1.2 Mask Options

The following MC68HC05J1A mask options are available:

- On-chip oscillator connections: crystal/ceramic resonator connections or resistor-capacitor (RC) network connections
- Crystal/ceramic resonator feedback resistor: connected or not connected (available only with crystal/ceramic oscillator mask option)
- STOP instruction: enabled or disabled (converted to WAIT instruction)
- External interrupt pins: edge-triggered or edge- and level-triggered
- Port A and port B pulldown resistors: connected or not connected
- COP watchdog timer: enabled or disabled
- Port A external interrupt capability: enabled or disabled

1.3 MCU Structure

Figure 1-1 shows the structure of the MC68HC05J1A MCU.

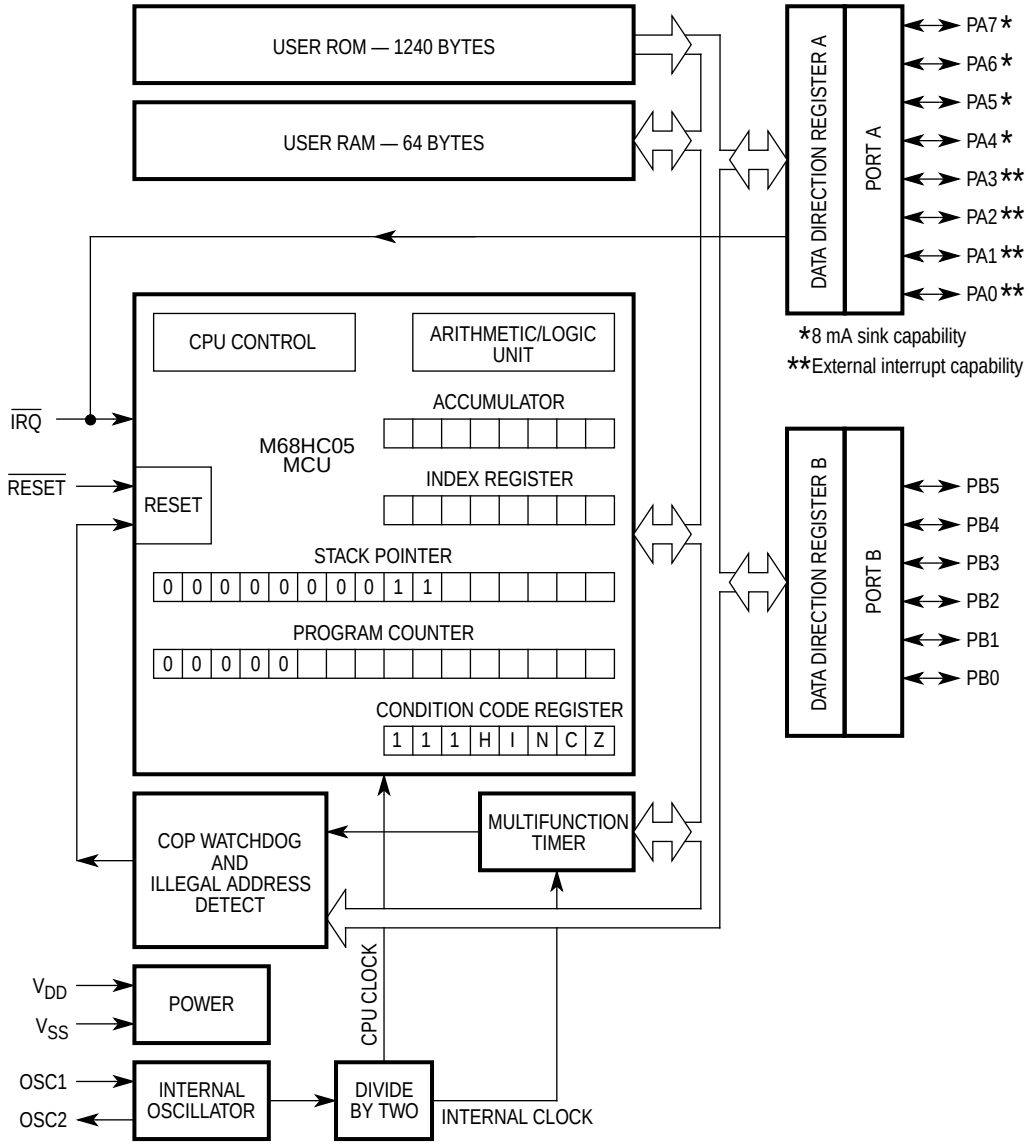


Figure 1-1. MC68HC05J1A Block Diagram

1.4 Pin Assignments

Figure 1-2 shows the MC68HC05J1A pin assignments.

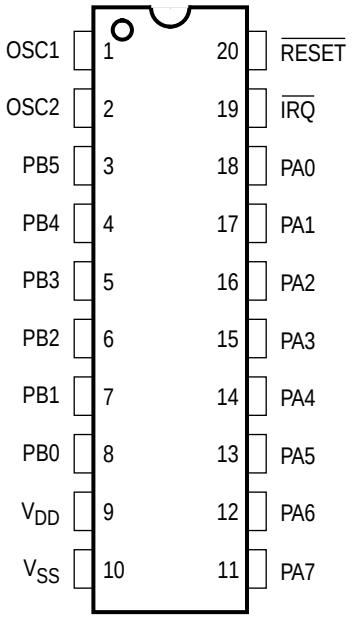


Figure 1-2. Pin Assignments

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 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 bypass capacitors 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.

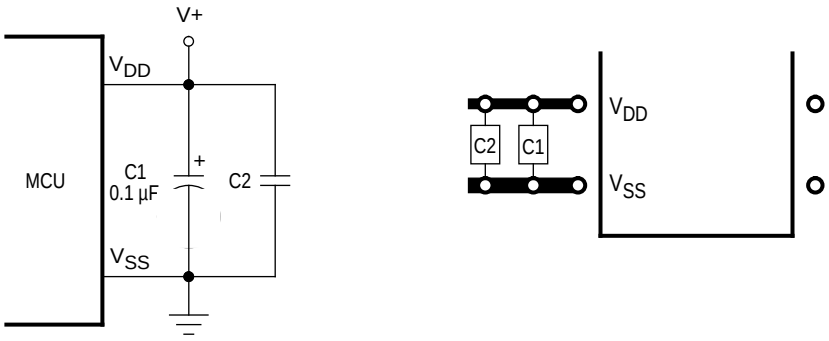


Figure 1-3. Bypassing Layout Recommendation

1.4.2 OSC1 and OSC2

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

- Crystal
- Ceramic resonator
- Resistor-capacitor network
- External clock signal

The frequency of the internal oscillator is f_{OSC}. The MCU divides the internal oscillator output by two to produce the internal clock with a frequency of f_{OP}.

An internal feedback resistor between the OSC1 and OSC2 pins is available as a mask option. The feedback resistor mask option is available only when the crystal/ceramic resonator mask option is also selected.

1.4.2.1 Crystal

With the crystal/ceramic resonator mask option, a crystal connected to the OSC1 and OSC2 pins can drive the on-chip oscillator. Figure 1-4 and Figure 1-5 show 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.

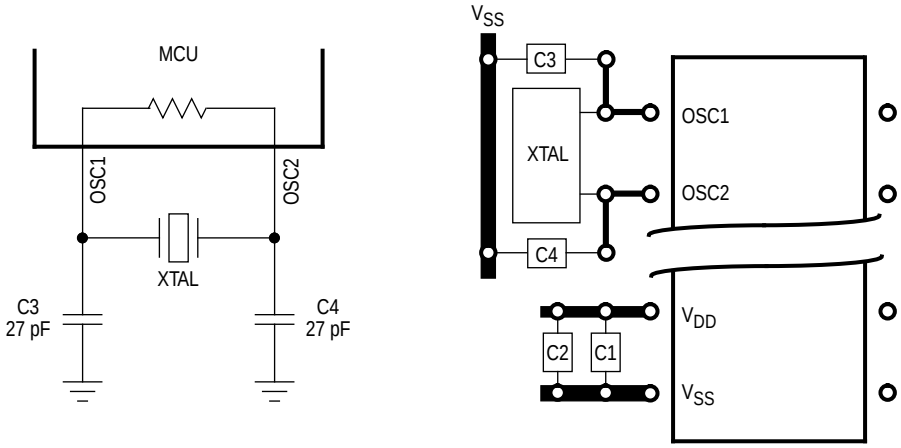


Figure 1-4. Crystal Connections with Feedback Resistor Mask Option

NOTE

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

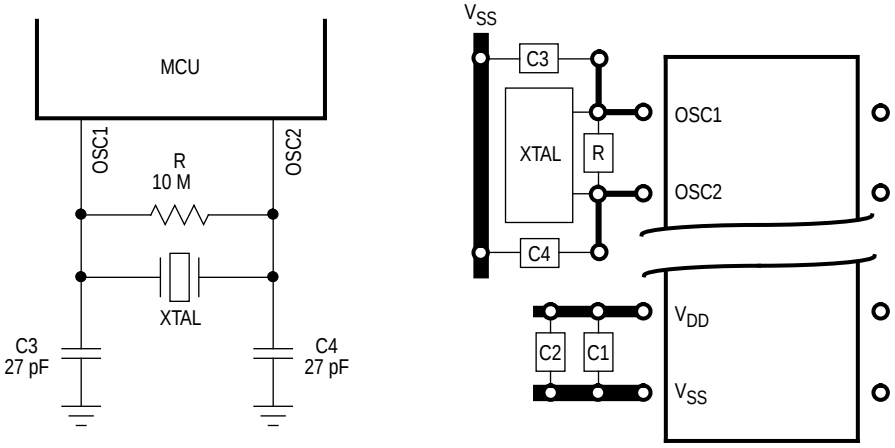


Figure 1-5. Crystal Connections without Feedback Resistor Mask Option

1.4.2.2 Ceramic Resonator

To reduce cost, use a ceramic resonator in place of the crystal. Use the circuit in Figure 1-6 or Figure 1-7 for a ceramic resonator, and follow the resonator manufacturer's recommendations. The load 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.

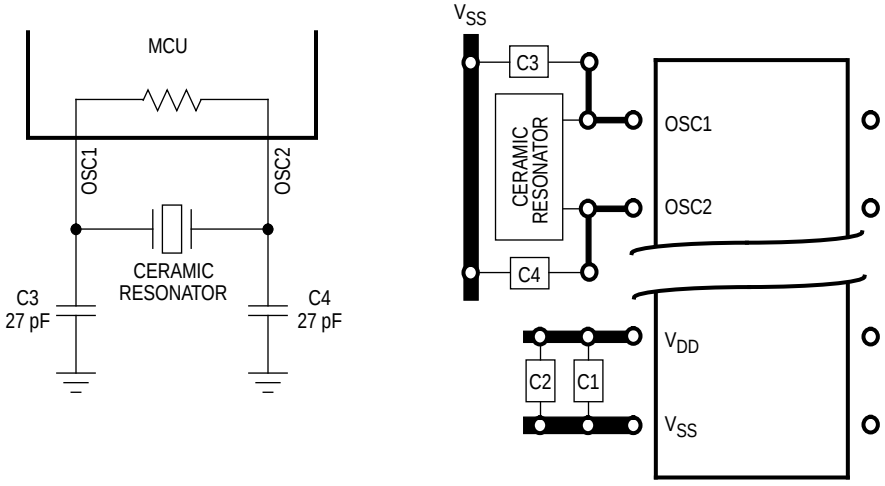


Figure 1-6. Ceramic Resonator Connections with Feedback Resistor Mask Option

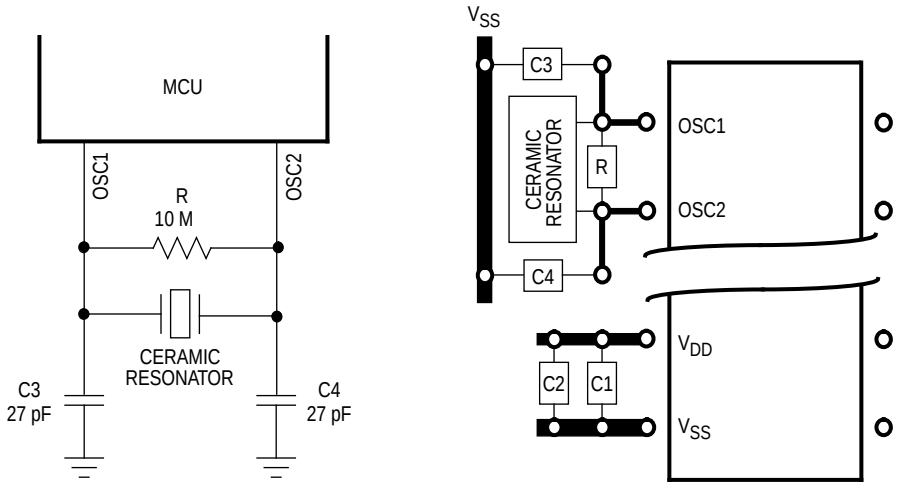


Figure 1-7. Ceramic Resonator Connections without Feedback Resistor Mask Option

1.4.2.3 RC Oscillator

For maximum cost reduction, the RC oscillator mask option allows the configuration shown in Figure 1-8 to drive the on-chip oscillator. The OSC2 signal is a square wave, and the signal on OSC1 is a triangular wave. The optimum frequency for the RC oscillator configuration is 2 MHz. Mount the RC components as close as possible to the pins for start-up stabilization and to minimize output distortion.

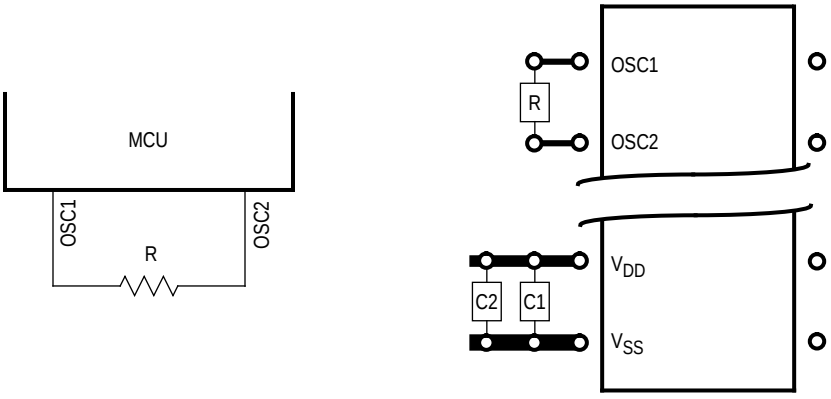


Figure 1-8. RC Oscillator Connections

1.4.2.4 External Clock

With the RC oscillator mask option, an external clock from another CMOS-compatible device can drive the OSC1 input. Leave the OSC2 pin unconnected, as Figure 1-9 shows.

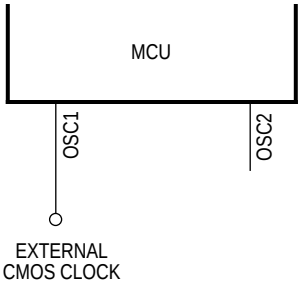


Figure 1-9. External Clock Connections

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 Exernal Reset** for more information.

1.4.4 $\overline{\text{IRQ}}$

The $\overline{\text{IRQ}}$ pin is an asynchronous external interrupt pin. See **4.1.2.1 $\overline{\text{IRQ}}$ Pin**.

1.4.5 PA7–PA0

PA7–PA0 are the pins of port A, a general-purpose bidirectional I/O port. See **7.2 Port A**.

1.4.6 PB5–PB0

PB5–PB0 are the pins of port B, a general-purpose bidirectional I/O port. See **7.3 Port B**.

SECTION 2
MEMORY

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

2.1 Memory Map

The CPU can address 2 Kbytes of memory space as shown in Figure 2-1. 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.

2.2 Input/Output Section

The first 32 addresses of the memory space, $0001–$001F, are the I/O section. These are the addresses of the I/O control registers, status registers, and data registers. See Figure 2-2.

One I/O register shown in Figure 2-2 is located outside the 32-byte I/O section: the computer operating properly (COP) register is mapped at $07F0.

2.3 RAM

The 64 addresses from $00C0 to $00FF serve as both the user RAM and the stack RAM. The CPU uses five stack RAM bytes to save all CPU register contents before processing an interrupt. During a subroutine call, the CPU uses two bytes 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.

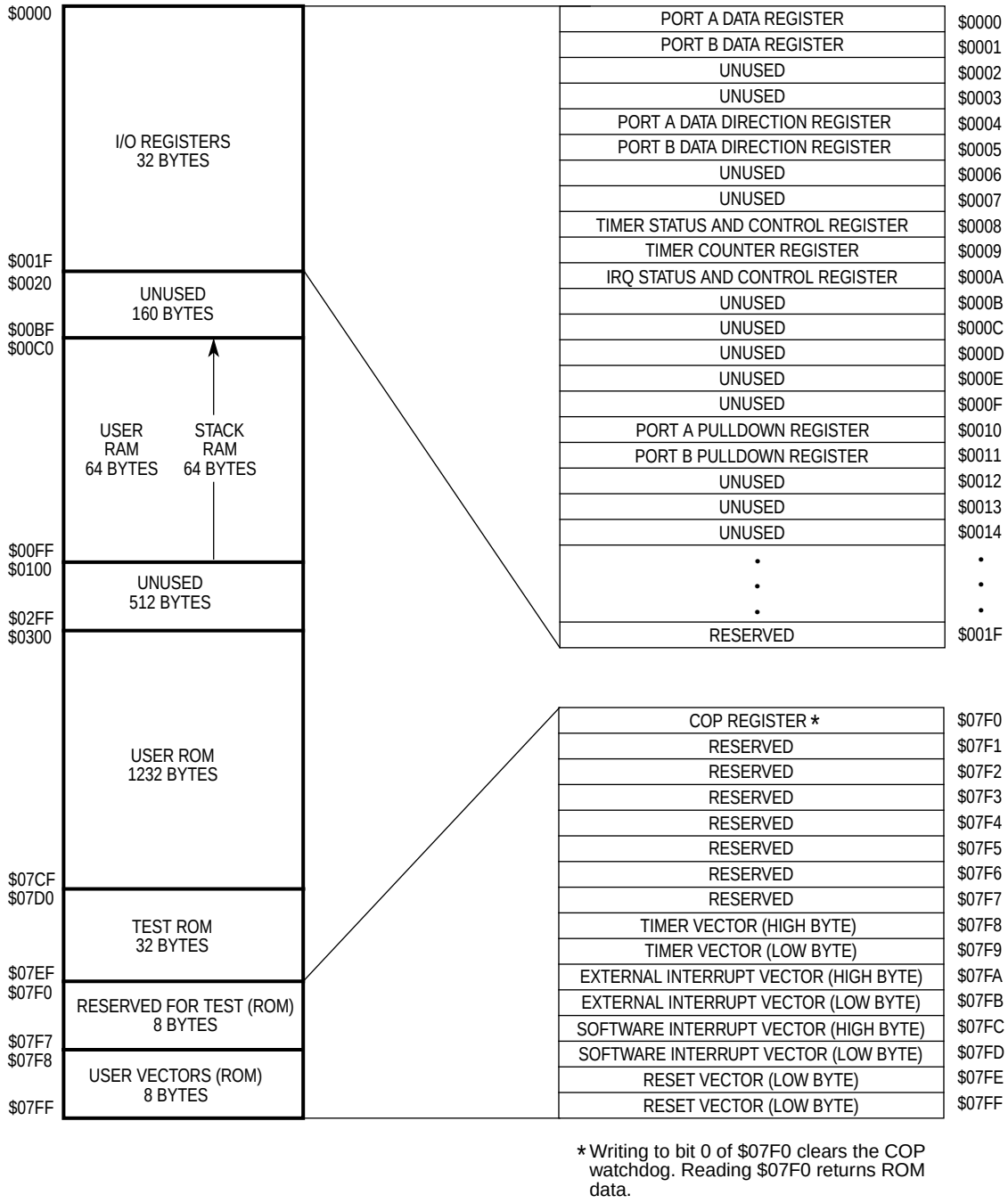


Figure 2-1. Memory Map

	Bit 7	6	5	4	3	2	1	Bit 0	
$0000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PORTA
$0001	0	0	PB5	PB4	PB3	PB2	PB1	PB0	PORTB
$0002									UNUSED
$0003									UNUSED
$0004	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	DDRA
$0005	0	0	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	DDRB
$0006									UNUSED
$0007									UNUSED
$0008	TOF	RTIF	TOIE	RTIE	TOFR	RTIFR	RT1	RT0	TSCR
$0009	Bit 7	6	5	4	3	2	1	Bit 0	TCR
$000A	IRQE	0	0	0	IRQF	0	IRQR	0	ISCR
$000B									UNUSED
$000C									UNUSED
$000D									UNUSED
$000E									UNUSED
$000F									UNUSED
$0010	PDRA7	PDRA6	PDRA5	PDRA4	PDRA3	PDRA2	PDRA1	PDRA0	PDRA
$0011			PDRB5	PDRB4	PDRB3	PDRB2	PDRB1	PDRB0	PDRB
$0012									UNUSED
$001E									UNUSED
$001F									RESERVED

Figure 2-2. I/O Registers

2.4 ROM

The ROM is located in two areas of the memory map:

- Addresses $0300–$07CF contain 1232 bytes of user ROM
- Addresses $07F8–07FF contain 16 bytes of ROM reserved for user vectors

SECTION 3
CENTRAL PROCESSOR UNIT (CPU)

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.

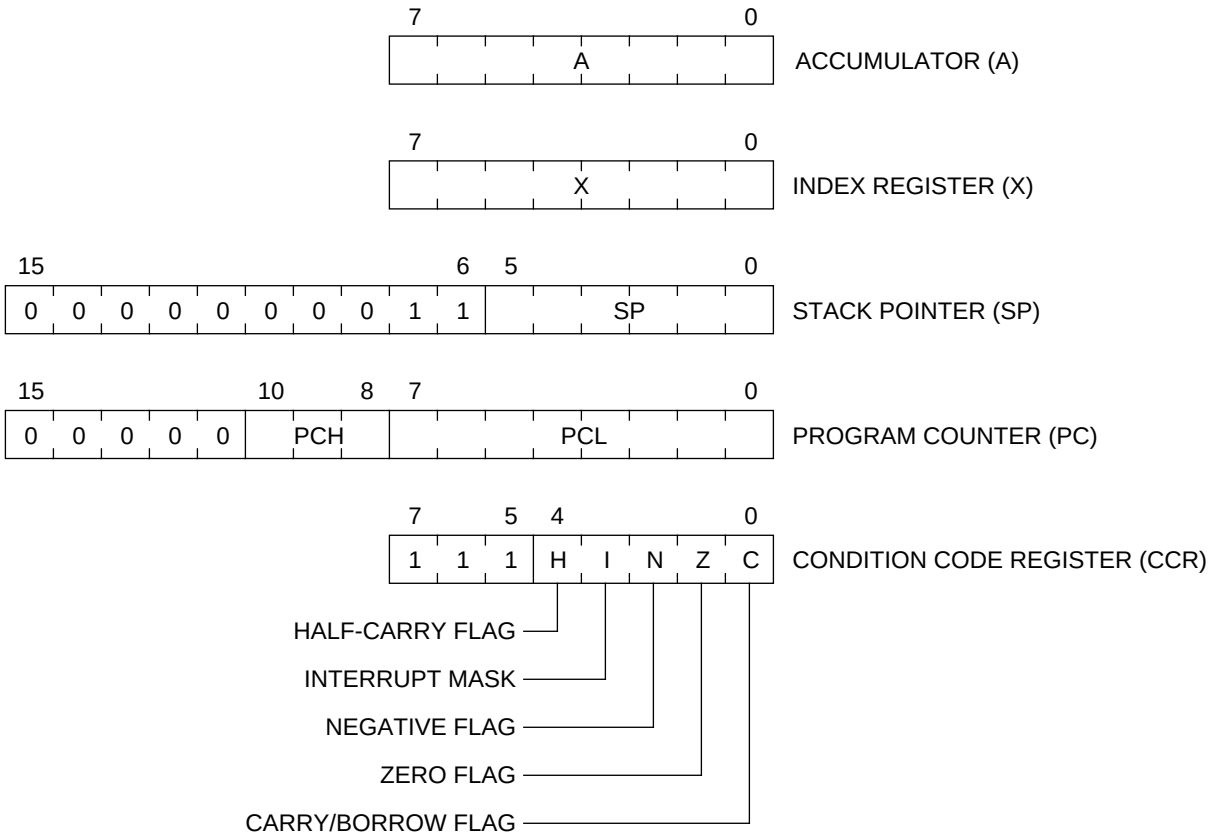


Figure 3-1. Programming Model

3.1.1 Accumulator (A)

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

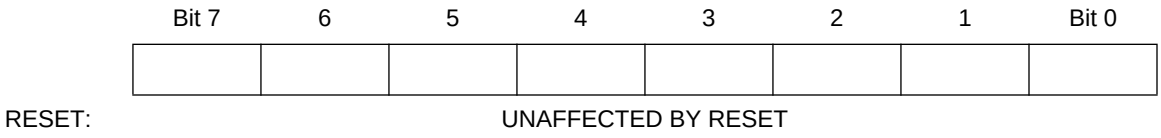


Figure 3-2. Accumulator (A)

3.1.2 Index Register (X)

In the indexed addressing modes, the CPU uses the byte in the index register to determine the conditional address of the operand. (See **9.1 Addressing Modes**.)

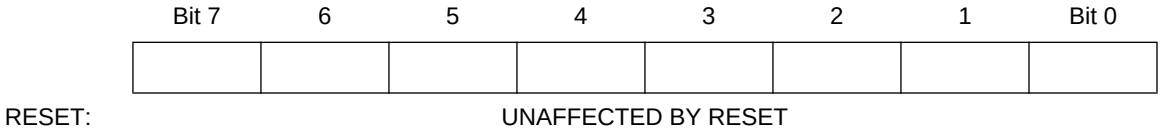


Figure 3-3. Index Register (X)

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

3.1.3 Stack Pointer (SP)

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	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 3-4. Stack Pointer (SP)

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 (PC)

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched. The five most significant bits of the program counter are ignored internally and appear as 00000.

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.

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0											
RESET:	0	0	0	0	0	LOADED WITH VECTOR FROM $07FE AND $07FF										

Figure 3-5. Program Counter (PC)

3.1.5 Condition Code Register (CCR)

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	C	Z
RESET:	1	1	1	U	1	U	U	U

U = UNAFFECTED

Figure 3-6. Condition Code Register (CCR)

3.1.5.1 Half-Carry Flag (H)

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 (I)

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 (N)

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 (Z)

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 (C)

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 clock 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 interrupt requests:

- SWI instruction
- $\overline{\text{IRQ}}$ pin
- PA3–PA0 pins (mask option)
- Multifunction timer

An interrupt temporarily stops normal program execution to process a particular event. An interrupt does not stop the execution of the instruction in progress, 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 vector address.

4.1.1 Software Interrupt

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

4.1.2 External Interrupts

The following sources can generate external interrupts:

- $\overline{\text{IRQ}}$ pin
- PA3–PA0 pins (mask option)

Setting the I bit in the condition code register or clearing the IRQE bit in the interrupt status and control register disables external interrupts.

4.1.2.1 $\overline{\text{IRQ}}$ Pin

An interrupt signal on the $\overline{\text{IRQ}}$ 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 and the IRQE bit in the interrupt status and control register. If the I bit is clear and the IRQE bit is set, the CPU then begins the interrupt sequence. The CPU clears the IRQ latch while it fetches the interrupt vector, so that another external interrupt request can be latched 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 external interrupt logic.

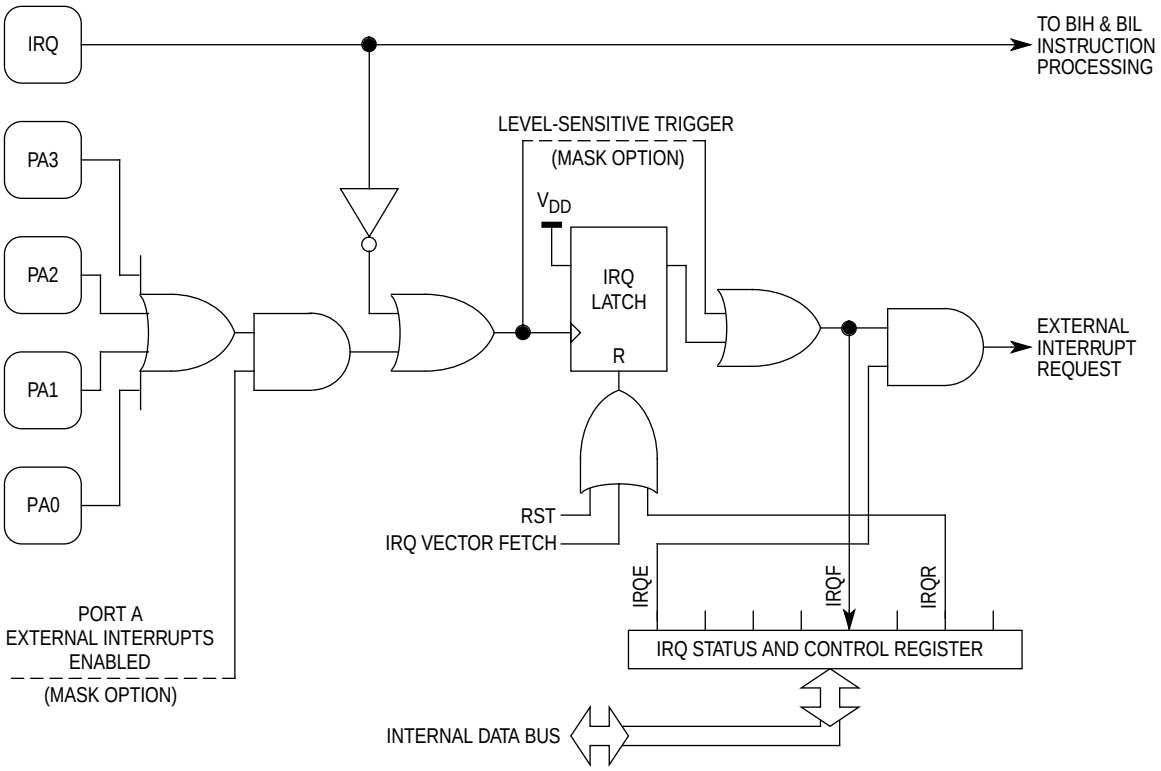


Figure 4-1. External Interrupt Logic

External interrupt triggering sensitivity is a mask option. The $\overline{\text{IRQ}}$ pin can be negative edge-triggered only or negative edge- and low level-triggered.

With the mask option for an edge- and level-sensitive external interrupt trigger, a falling edge or a low level on the $\overline{\text{IRQ}}$ pin latches an external interrupt request. Edge- and level-sensitive triggering allows the use of multiple wired-OR external interrupt sources. An external interrupt request is latched as long as any source is holding the $\overline{\text{IRQ}}$ pin low.

With the mask option for an edge-sensitive only external interrupt trigger, a falling edge on the $\overline{\text{IRQ}}$ pin latches an external interrupt request. A subsequent external interrupt request can be latched only after the voltage level on the $\overline{\text{IRQ}}$ pin returns to logic one and then falls again to logic zero.

4.1.2.2 PA3–PA0 Pins

The mask option for port A external interrupts enables pins PA3–PA0 to serve as additional external interrupt sources. An interrupt signal on a PA3–PA0 pin latches an external interrupt request. After completing the current instruction, the CPU tests the IRQ latch. If the IRQ latch is set, the CPU then tests the I bit in the condition code register and the IRQE bit in the interrupt status and control register. If the I bit is clear and the IRQE bit is set, the CPU then begins the interrupt sequence. The CPU clears the IRQ latch while it fetches the interrupt vector, so that another external interrupt request can be latched 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.

External interrupt triggering sensitivity is a mask option. The PA3–PA0 pins can be positive edge-triggered only or positive edge- and high level-triggered.

With the mask option for an edge- and level-sensitive external interrupt trigger, a rising edge or a high level on a PA3–PA0 pin latches an external interrupt request. Edge- and level-sensitive triggering allows the use of multiple wired-OR external interrupt sources. As long as any source is holding a PA3–PA0 pin high, an external interrupt request is latched, and the CPU continues to execute the interrupt service routine.

With the mask option for an edge-sensitive only external interrupt trigger, a rising edge on a PA3–PA0 pin latches an external interrupt request. A subsequent external interrupt request can be latched only after the voltage level of the previous interrupt signal returns to logic zero and then rises again to logic one.

4.1.2.3 IRQ Status and Control Register (ISCR)

The IRQ status and control register, shown in Figure 4-2, contains an external interrupt mask, an external interrupt flag, and a flag reset bit.

ISCR — IRQ Status and Control Register **$000A**

	Bit 7	6	5	4	3	2	1	Bit 0
	IRQE	0	0	0	IRQF	0	IRQR	0
RESET	1	0	0	0	0	0	0	0

Figure 4-2. IRQ Status and Control Register (ISCR)

IRQE — External Interrupt Request Enable

This read/write bit enables external interrupts. Resets set the IRQE bit.

1 = External interrupt processing enabled
0 = External interrupt processing disabled

IRQF — External Interrupt Request Flag

The IRQ flag is a clearable, read-only bit that is set when an external interrupt request is pending. Resets clear the IRQF bit.

1 = Interrupt request pending
0 = No interrupt request pending

The following conditions set the IRQ flag:

- An external interrupt signal on the $\overline{\text{IRQ}}$ pin
- An external interrupt signal on pin PA3, PA2, PA1, or PA0 when PA3–PA0 are enabled to serve as external interrupt sources

The CPU clears the IRQ flag when fetching the interrupt vector. Writing to the IRQ flag has no effect. Clear the IRQ flag by writing a logic one to the IRQR bit.

IRQR — Interrupt Request Reset

This write-only bit clears the IRQ flag.

1 = IRQF bit cleared
0 = No effect

4.1.3 Timer Interrupts

The multifunction timer can generate the following interrupts:

- Timer overflow interrupt
- Real-time interrupt

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

4.1.3.1 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. See **8.1 Timer Status and Control Register (TSCR)**.

4.1.3.2 Real-Time Interrupt

A real-time interrupt request occurs if the real-time interrupt flag, RTIF, becomes set while the real-time interrupt enable bit, RTIE, is also set. See **8.1 Timer Status and Control Register (TSCR)**.

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 4-3
- 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:
  - $07FC and $07FD (software interrupt vector)
  - $07FA and $07FB (external interrupt vector)
  - $07F8 and $07F9 (timer interrupt vector)

The return from interrupt (RTI) instruction causes the CPU to recover the CPU registers from the stack as shown in Figure 4-3.

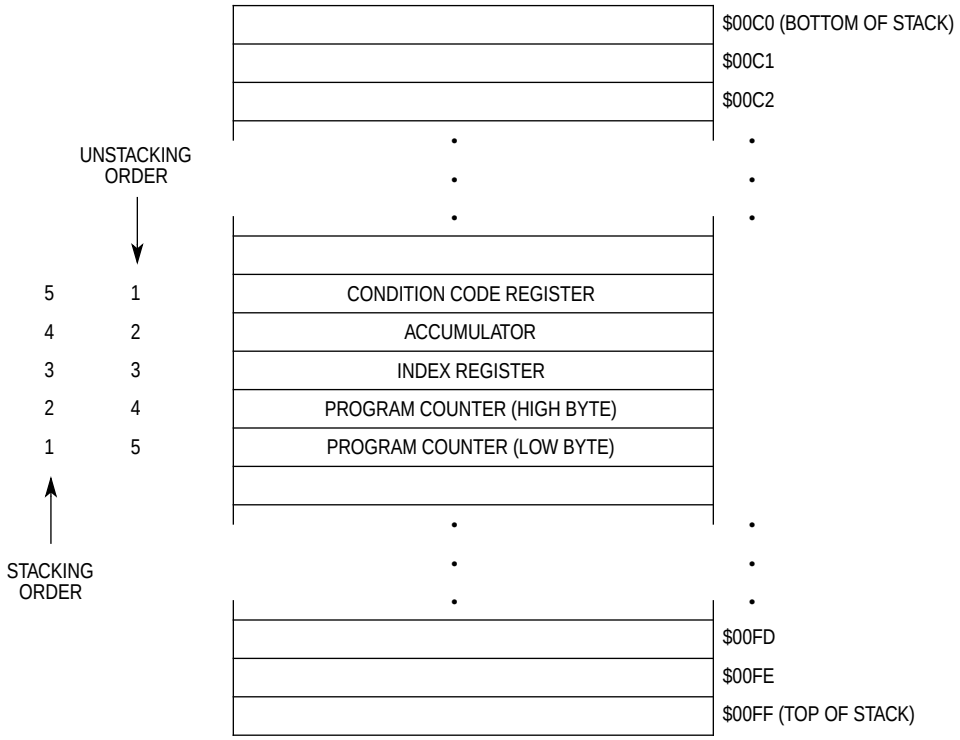


Figure 4-3. Stacking Order

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 RESET Pin COP Watchdog(1) Illegal Address	None	None None None None	1 1 1 1	$07FE–$07FF
Software Interrupt (SWI)	User Code	None	None	Same Priority As Instruction	$07FC–$07FD
External Interrupt	$\overline{\text{IRQ}}$ Pin PA3 Pin(2) PA2 Pin(2) PA1 Pin(2) PA0 Pin(2)	IRQE Bit	I Bit	2	$07FA–$07FB
Timer Interrupts	TOF Bit RTIF Bit	TOFE Bit RTIE Bit	I Bit	3	$07F8–$07F9

1. The COP watchdog is a mask option.
2. Port A external interrupt capability is a mask option.

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

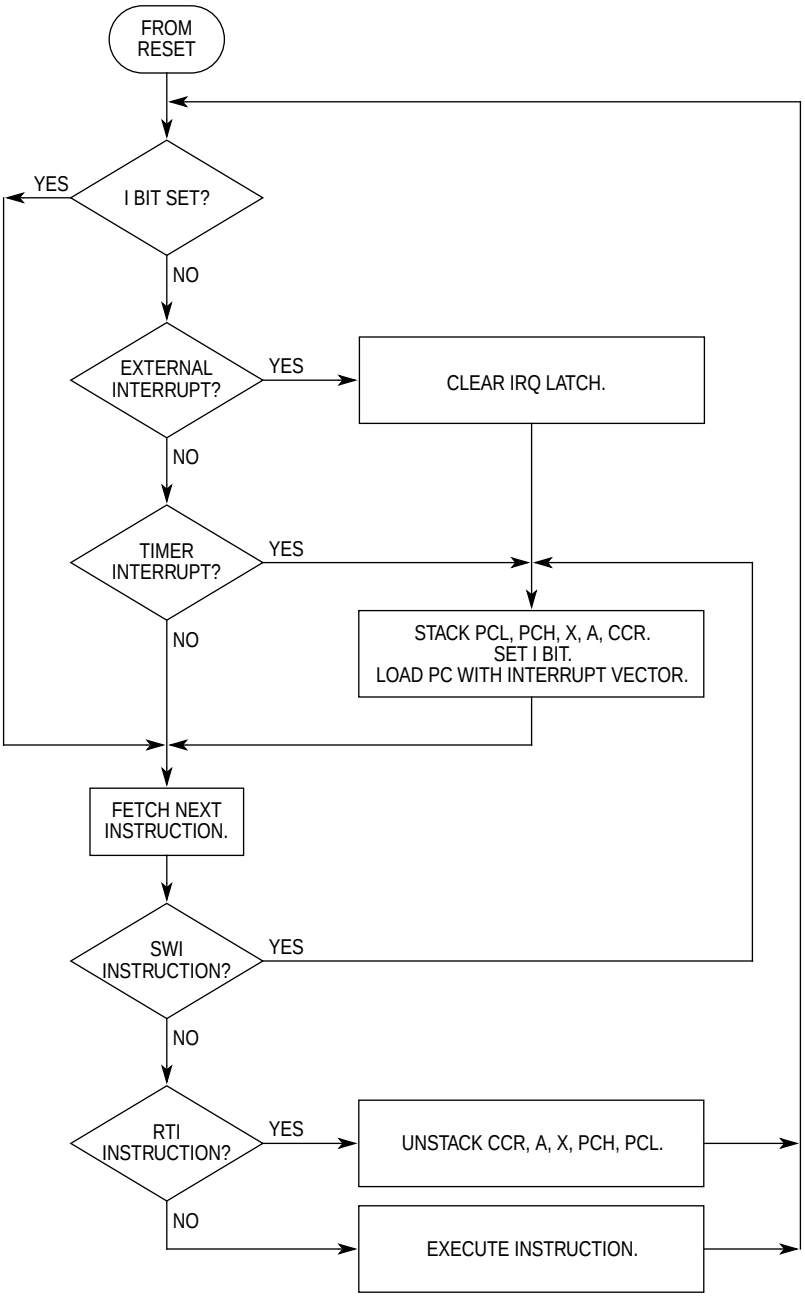


Figure 4-4. Interrupt Flowchart

SECTION 5
RESETS

This section describes the four reset sources and how they initialize the MCU.

5.1 Reset Types

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. The following conditions produce a reset:

- Initial power-up (power-on reset)
- A logic zero applied to the $\overline{\text{RESET}}$ pin (external reset)
- Timeout of the mask-optional COP watchdog (COP reset)
- An opcode fetch from an address not in the memory map (illegal address reset)

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.

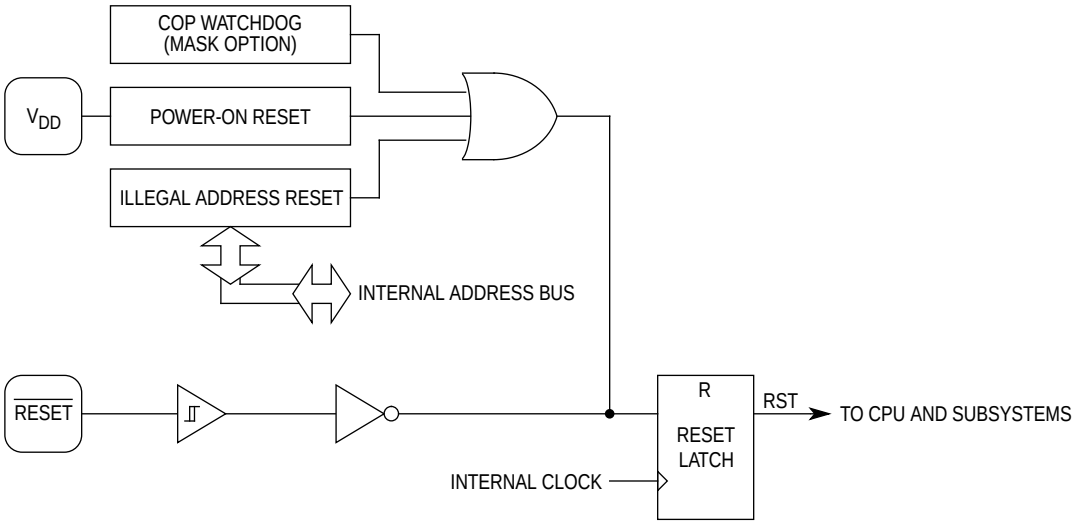


Figure 5-1. Reset Sources

5.1.3 Computer Operating Properly (COP) 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. (See **8.3 COP Watchdog**.) To clear the COP watchdog and prevent a COP reset, write a logic zero to bit 0 (COPC) of the COP register at location $07F0. The COP register, shown in Figure 5-2, is a write-only register that returns the contents of a ROM location when read.

The COP watchdog function is a mask option.

COPR — COP Register **$07F0**

	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.

5.1.4 Illegal Address Reset

An opcode fetch from an address that is not in the ROM (locations $0300–$07FF) or the RAM (locations $00C0–$00FF) generates an illegal address reset.

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
- Sets the IRQE bit in the interrupt status and control register
- Loads the program counter with the user-defined reset vector from locations $07FE and $07FF
- 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 bits DDRA7–DDRA0 in data direction register A so that port A pins are inputs
- Clears bits PDIA7–PDIA0 in pulldown register A so that port A pulldown devices are enabled
- Clears bits DDRB5–DDRB0 in data direction register B so that port B pins are inputs
- Clears bits PDIB5–PDIB0 in pulldown register B so that port B pulldown devices are enabled
- Has no effect on port A or port B data registers

5.2.3 Multifunction Timer

A reset has the following effects on the multifunction timer:

- Clears the timer status and control register
- Clears the timer counter register

5.2.4 COP Watchdog

A reset clears the COP watchdog (if the COP watchdog is enabled by mask option).

SECTION 6
LOW POWER MODES

This section describes the four low power modes:

- Stop mode
- Wait mode
- Halt mode (mask option)
- 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:

- Clears TOF and RTIF, the timer interrupt flags in the timer status and control register, removing any pending timer interrupts
- Clears TOIE and RTIE, the timer interrupt enable bits in the timer status and control register, disabling further timer interrupts
- Clears the multifunction timer counter
- Sets the IRQE bit in the IRQ status and control register to enable external interrupts
- Clears the I bit in the condition code register, enabling interrupts
- Stops the internal oscillator, turning off the CPU clock and the timer clock, including the COP watchdog

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

The following conditions 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 $07FA and $07FB.

- An external interrupt signal on a port A external interrupt pin — If the mask option for the port A external interrupt function is selected, a low-to-high transition on a PA3–PA0 pin loads the program counter with the contents of locations $07FA and $07FB.

- External reset — A logic zero on the $\overline{\text{RESET}}$ pin resets the MCU and loads the program counter with the contents of locations $07FE and $07FF.

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

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

- Sets the IRQE bit in the IRQ status and control register, enabling external interrupts

- Stops the CPU clock, but allows the internal oscillator and timer clock to continue to run

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:

- 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 $07FA and $07FB.

- An external interrupt signal on a port A external interrupt pin — If the mask option for the port A external interrupt function is selected, a low-to-high transition on a PA3–PA0 pin loads the program counter with the contents of locations $07FA and $07FB.

- A timer interrupt — A timer overflow or a real-time interrupt request loads the program counter with the contents of locations $07F8 and $07F9.

- A COP watchdog reset — A timeout of the mask-optional COP watchdog resets the MCU and loads the program counter with the contents of locations $07FE and $07FF. Software can enable real-time 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 $07FE and $07FF.

6.3 Halt Mode

If the mask option to disable the STOP instruction is selected, a STOP instruction puts the MCU in halt mode. The halt mode is identical to the wait mode, except that a recovery delay of 1–4064 internal clock cycles occurs when the MCU exits the halt mode. If the mask option to disable the STOP instruction is selected, the COP watchdog cannot be inadvertently turned off by a STOP instruction.

Figure 6-1 shows the sequence of events in stop, wait, and halt modes.

6.4 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.

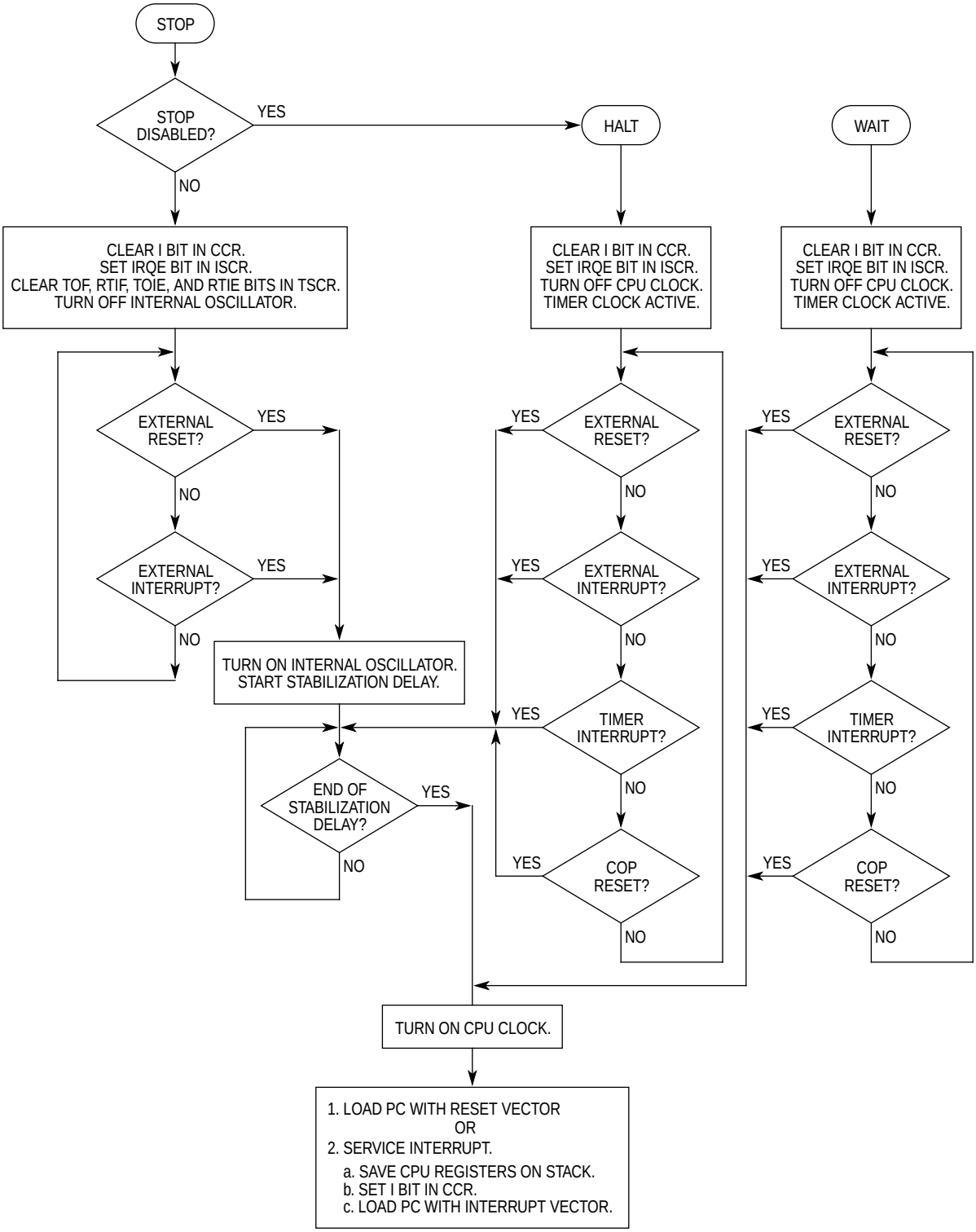


Figure 6-1. STOP/WAIT/HALT Flowchart

SECTION 7
PARALLEL I/O

This section describes the two bidirectional I/O ports.

7.1 I/O Port Function

The 14 bidirectional I/O pins form two parallel I/O ports. Each I/O pin is programmable as an input or an output. The contents of the data direction registers determine the data direction of each I/O pin.

All 14 I/O pins have mask-optional software-programmable pulldown devices.

7.2 Port A

Port A is an 8-bit general-purpose bidirectional I/O port with the following features:

- Programmable pulldown devices (mask option)
- 8 mA current sinking capability (pins PA7–PA4)
- External interrupt capability (mask option: pins PA3–PA0)

7.2.1 Port A Data Register (PORTA)

The port A data register contains a bit for each of the port A pins. When a port A pin is programmed to be an output, the state of its data register bit determines the state of the output pin. When a port A pin is programmed to be an input, reading the port A data register returns the logic state of the pin.

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. Resets have no effect on port A data.

7.2.2 Data Direction Register A (DDRA)

The contents of data direction register A determine 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 associated port A pin; a logic zero disables the output buffer. A reset initializes all DDRA bits to 0, 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.

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 DDRA bits from logic zero to logic one.

7.2.3 Pulldown Register A (PDRA)

All port A pins have mask-optional programmable pulldown devices that typically sink 100 µA. Clearing the PDIA7–PDIA0 bits in pulldown register A turns on the pulldown devices. (See Figure 7-3.) Pulldown register A can turn on a port A pulldown device only when the port A pin is an input. Reset clears the PDIA7–PDIA0 bits, turning on all the port A pulldown devices.

PDRA — Pulldown Register A **$0010**

	Bit 7	6	5	4	3	2	1	Bit 0
	PDIA7	PDIA6	PDIA5	PDIA4	PDIA3	PDIA2	PDIA1	PDIA0
RESET:	0	0	0	0	0	0	0	0

Figure 7-3. Pulldown Register A (PDRA)

PDIA7–PDIA0 — Port A Pulldown Inhibit Bits 7–0

Writing logic zeros to these write-only bits turns on the port A pulldown devices. Reading pulldown register A returns undefined data.

1 = Corresponding port A pin pulldown device turned off
0 = Corresponding port A pin pulldown device turned on

NOTE

Avoid a floating port A input by clearing its pulldown register bit before changing its DDRA bit from logic one to logic zero.

NOTE

Do not use read-modify-write instructions on pulldown register A.

7.2.4 Port A External Interrupts

If the port A external interrupt mask option is selected, the PA3–PA0 pins serve as external interrupt pins in addition to the $\overline{IRQ}$ pin.

External interrupt triggering sensitivity is a mask option. The PA3–PA0 pins can be positive edge-triggered or positive edge- and high level-triggered.

NOTE

When testing for external interrupts, the BIH and BIL instructions test the voltage on the $\overline{IRQ}$ pin, not the state of the internal IRQ signal. Therefore, BIH and BIL cannot test the port A external interrupt pins.

Figure 7-4 shows the port A I/O logic.

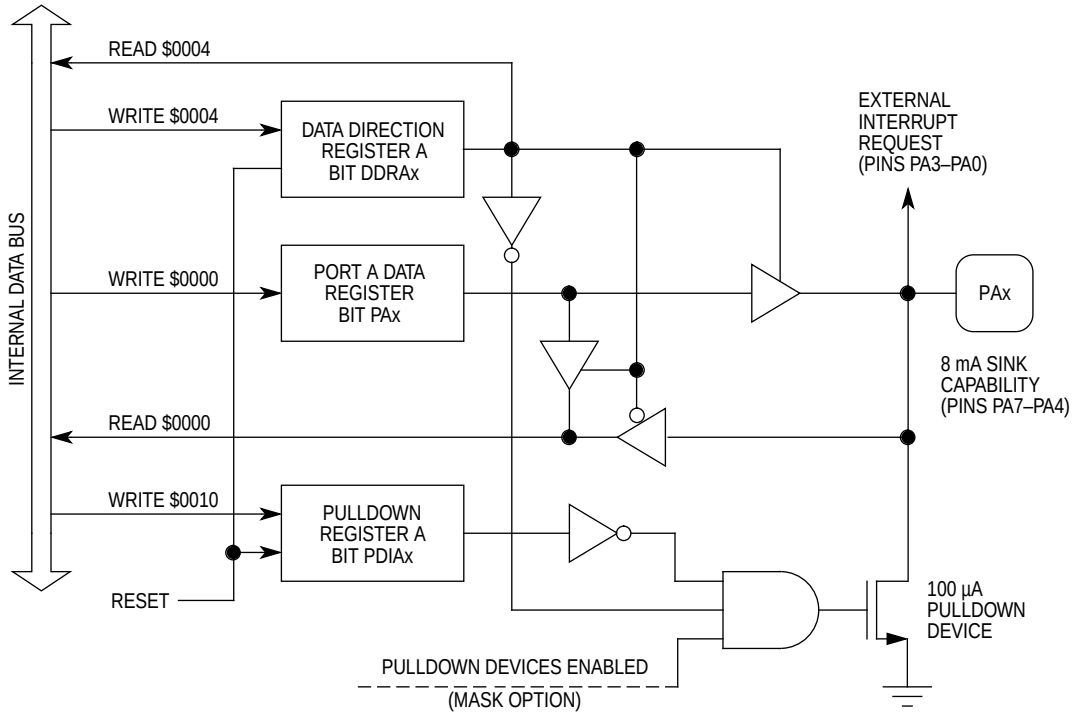


Figure 7-4. Port A I/O Circuit

When a port A pin is programmed as an output, reading the port bit actually reads the value of the data latch and not the voltage on the pin itself. 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 DDR bit. Table 7-1 summarizes the operations of the port A pins.

Table 7-1. Port A Pin Functions

Pulldown Mask Option	Control Bits		I/O Pin Mode	Accesses to PDRA		Accesses to DDRA	Accesses to PORTA	
	PDIAx	DDRAx		Read	Write	Read/Write	Read	Write
No	X(1)	0	Input, Hi-Z	U(2)	PDIA7-0	DDRA7-0	Pin	PA7-0
No	X	1	Output	U	PDIA7-0	DDRA7-0	PA7-0	PA7-0
Yes	0	0	Input, Pulldown On	U	PDIA7-0	DDRA7-0	Pin	PA7-0
Yes	0	1	Output, Pulldown On	U	PDIA7-0	DDRA7-0	PA7-0	PA7-0
Yes	1	0	Input, Hi-Z	U	PDIA7-0	DDRA7-0	Pin	PA7-0
Yes	1	1	Output	U	PDIA7-0	DDRA7-0	PA7-0	PA7-0

1. X = Don’t care
2. U = Undefined

7.3 Port B

Port B is a 6-bit general-purpose bidirectional I/O port with programmable pulldown devices.

7.3.1 Port B Data Register (PORTB)

The port B data register contains a bit for each of the port B pins. When a port B pin is programmed to be an output, the state of its data register bit determines the state of the output pin. When a port B pin is programmed to be an input, reading the port B data register returns the logic state of the pin.

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

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	PB5	PB4	PB3	PB2	PB1	PB0
RESET:	UNAFFECTED BY RESET							

Figure 7-5. Port B Data Register (PORTB)

PB5–PB0 — 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 the port B data direction register.

Bits 7 and 6 — Not used
Bits 7 and 6 always read as logic zeros. Writes to these bits have no effect.

7.3.2 Data Direction Register B (DDRB)

The contents of DDRB determine whether each port B pin is an input or an output. Writing a logic one to a DDRB bit enables the output buffer for the associated port B pin; a logic zero disables the output buffer. A reset initializes all DDRB bits to logic zero, configuring all port B pins as inputs.

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

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
RESET:	0	0	0	0	0	0	0	0

Figure 7-6. Data Direction Register B (DDRB)

DDRB5–DDRB0 — Data Direction Bits B5–B0

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

Bit 7 and 6 — Not used

Bits 7 and 6 always read as logic zeros. Writes to these bits have no effect.

NOTE

Avoid glitches on port B pins by writing to the port B data register before changing DDRB bits from logic zero to logic one.

7.3.3 Pulldown Register B (PDRB)

All port B pins have mask-optional programmable pulldown devices that typically sink 100 µA. Clearing any of the PDIB5–PDIB0 bits in pulldown register B turns on the pulldown devices. (See Figure 7-7.) Pulldown register B can turn on a port B pulldown device only when the port B pin is an input. Reset clears bits PDIB5–PDIB0, turning on the port B pulldown devices.

PDRB — Pulldown Register B **$0011**

	Bit 7	6	5	4	3	2	1	Bit 0
			PDIB5	PDIB4	PDIB3	PDIB2	PDIB1	PDIB0
RESET:	—	—	0	0	0	0	0	0

Figure 7-7. Pulldown Register B (PDRB)

PDIB5–PDIB0 — Pulldown Inhibit Bits 5–0

Writing logic zeros to these write-only bits turns on the port B pulldown devices. Reading pulldown register B returns undefined data.

1 = Corresponding port B pin pulldown device turned off
0 = Corresponding port B pin pulldown device turned on

Bits 7 and 6 — Not used

NOTE

Avoid a floating port B input by clearing its pulldown register bit before changing its DDRB bit from logic one to logic zero.

NOTE

Do not use read-modify-write instructions on pulldown register B.

Figure 7-8 shows the port B I/O logic.

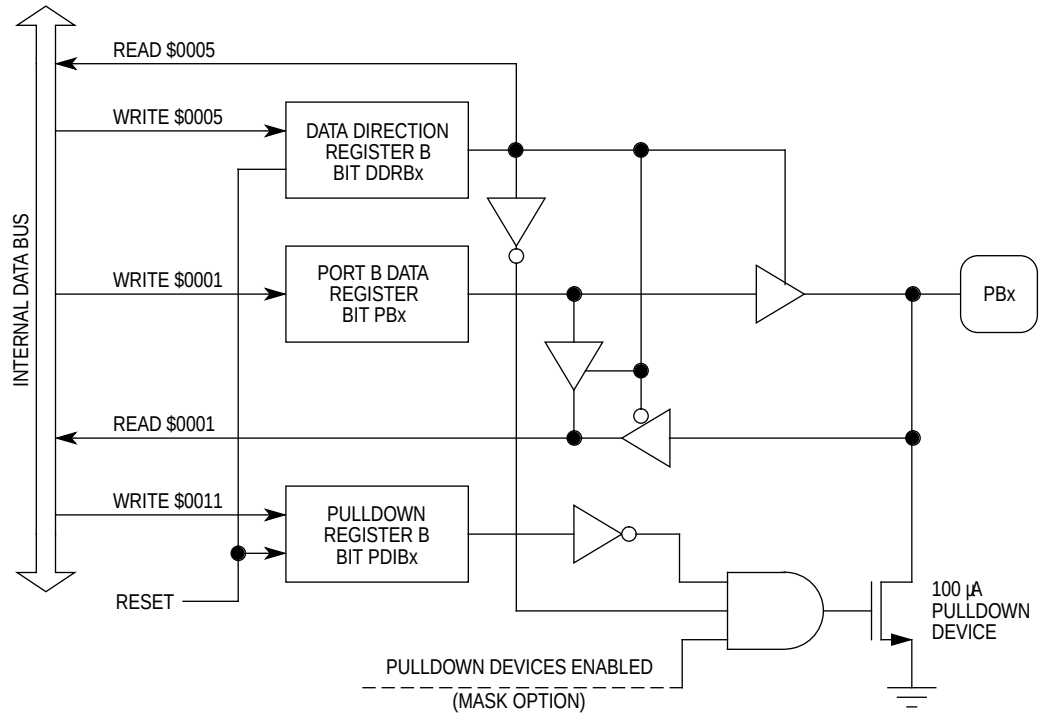


Figure 7-8. Port B I/O Circuit

Reading a port B output actually reads the value of the data latch and not the voltage on the pin itself. 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 DDR bit. Table 7-2 summarizes the operation of the port B pins.

Table 7-2. Port B Pin Functions

Pulldown Mask Option	Control Bits		I/O Pin Mode	Accesses to PDRB		Accesses to DDRB	Accesses to PORTB	
	PDIBx	DDRBx		Read	Write	Read/Write	Read	Write
No	X(1)	0	Input, Hi-Z	U(2)	PDIB7-0	DDRB7-0	Pin	PB7-0
No	X	1	Output	U	PDIB7-0	DDRB7-0	PB7-0	PB7-0
Yes	0	0	Input, Pulldown On	U	PDIB7-0	DDRB7-0	Pin	PB7-0
Yes	0	1	Output, Pulldown On	U	PDIB7-0	DDRB7-0	PB7-0	PB7-0
Yes	1	0	Input, Hi-Z	U	PDIB7-0	DDRB7-0	Pin	PB7-0
Yes	1	1	Output	U	PDIB7-0	DDRB7-0	PB7-0	PB7-0

1. X = Don’t care.
2. U = Undefined.

SECTION 8
MULTIFUNCTION TIMER

This section describes the operation of the multifunction timer and the COP watchdog. Figure 8-1 shows the organization of the timer subsystem.

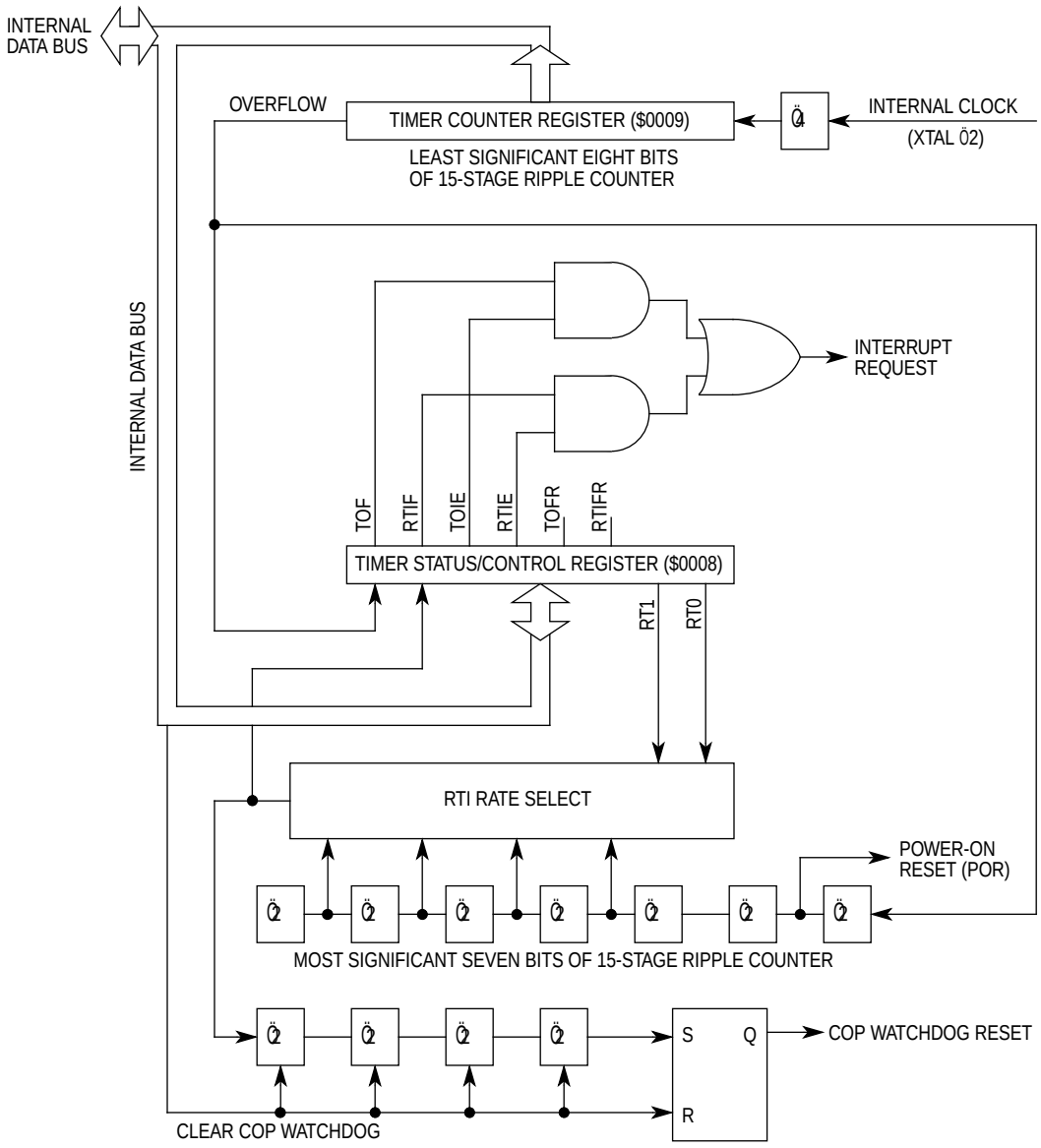


Figure 8-1. Multifunction Timer Block Diagram

8.1 Timer Status and Control Register (TSCR)

The read/write timer status and control register contains the following bits:

- Timer interrupt enable bits
- Timer interrupt flags
- Timer interrupt flag reset bits
- Timer interrupt rate select bits

TSCR — Timer Status and Control Register **$0008**

	Bit 7	6	5	4	3	2	1	Bit 0
	TOF	RTIF	TOIE	RTIE	TOFR	RTIFR	RT1	RT0
RESET:	0	0	0	0	0	0	1	1

Figure 8-2. Timer Status and Control Register (TSCR)

TOF — Timer Overflow Flag
This read-only flag becomes set when the first eight stages of the counter roll over from $FF to $00. TOF generates a timer overflow interrupt request if TOIE is also set. Clear TOF by writing a logic one to the TOFR bit. Writing to TOF has no effect. Reset clears TOF.

RTIF — Real-Time Interrupt Flag
This read-only flag becomes set when the selected RTI output becomes active. RTIF generates a real-time interrupt request if RTIE is also set. Clear RTIF by writing a logic one to the RTIFR bit. Writing to RTIF has no effect. Reset clears RTIF.

TOIE — Timer Overflow Interrupt Enable
This read/write bit enables timer overflow interrupts.
1 = Timer overflow interrupts enabled
0 = Timer overflow interrupts disabled

RTIE — Real-Time Interrupt Enable
This read/write bit enables real-time interrupts
1 = Real-time interrupts enabled
0 = Real-time interrupts disabled

TOFR — Timer Overflow Flag Reset
Writing a logic one to this write-only bit clears the TOF bit. TOFR always reads as logic zero. Reset clears TOFR.

RTIFR — Real-Time Interrupt Flag Reset

Writing a logic one to this write-only bit clears the RTIF bit. RTIFR always reads as logic zero. Reset clears RTIFR.

RT1, RT0 — Real-Time Interrupt Select Bits 1 and 0

These read/write bits select one of four real-time interrupt rates, as shown in Table 8-1. Because the selected RTI output drives the COP watchdog, changing the real-time interrupt rate also changes the counting rate of the COP watchdog. Reset sets RT1 and RT0.

NOTE

Changing RT1 and RT0 when a COP timeout is imminent or uncertain may cause a real-time interrupt request to be missed or an additional real-time interrupt request to be generated. Clear the COP timer just before changing RT1 and RT0.

Table 8-1. Real-Time Interrupt Rate Selection

RT1:RT0	RTI Rate	RTI Period (f_{OP} = 2 MHz)	COP Timeout Period (-0/+1 RTI Period)	Minimum COP Timeout Period (f_{OP} = 2 MHz)
0 0	$f_{OP} \div 2^{14}$	8.2 ms	8 × RTI Period	65.5 ms
0 1	$f_{OP} \div 2^{15}$	16.4 ms	8 × RTI Period	131.1 ms
1 0	$f_{OP} \div 2^{16}$	32.8 ms	8 × RTI Period	262.1 ms
1 1	$f_{OP} \div 2^{17}$	65.5 ms	8 × RTI Period	524.3 ms

8.2 Timer Counter Register (TCNTR)

A 15-stage ripple counter is the core of the timer. The value of the first eight stages is readable at any time from the read-only timer counter register.

TCNTR — Timer Counter Register **$0009**

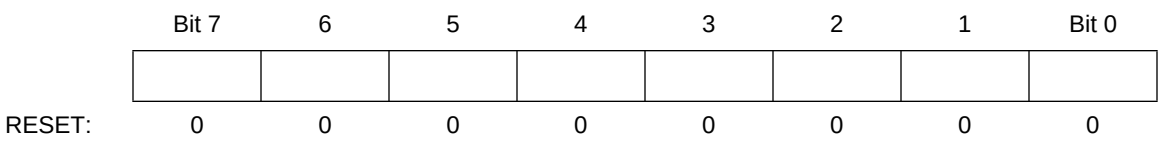


Figure 8-3. Timer Counter Register (TCNTR)

Power-on clears the entire counter chain and begins clocking the counter. After 4064 cycles, the power-on reset circuit is released, clearing the counter again and allowing the MCU to come out of reset.

A timer overflow function at the eighth counter stage allows a timer interrupt every 1024 internal clock cycles.

8.3 COP Watchdog

Four counter stages at the end of the timer make up the mask-optional computer operating properly (COP) watchdog. (See Figure 8-1.) The COP watchdog is a software error detection system that automatically times out and resets the MCU if not cleared periodically by a program sequence. Writing a logic zero to bit 0 of the COP register clears the COP watchdog and prevents a COP reset.

COPR — COP Register **$07F0**

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

Figure 8-4. COP Register (COPR)

COPC — COP Clear

This write-only bit resets the COP watchdog. Reading address $07F0 returns the ROM data at that address.

NOTE

The STOP instruction turns off the COP watchdog. In applications that depend on the COP watchdog, the STOP instruction can be disabled by a mask option.

SECTION 9
INSTRUCTION SET

This section describes the M68HC05 addressing modes and instruction types.

9.1 Addressing Modes

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

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

9.1.1 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.

9.1.2 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.

9.1.3 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.

9.1.4 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.

9.1.5 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 locaitons $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.

9.1.6 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 intructions 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.

9.1.7 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.

9.1.8 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 –128 to +127 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.

9.2 Instruction Types

The 117 MCU instructions fall into the following five categories:

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

9.2.1 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. Table 9-1 lists the register/memory instructions.

Table 9-1. Register/Memory Instructions

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

9.2.2 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 instruction (TST) is an exception to the read-modify-write sequence because it does not write a replacement value. Table 9-2 lists the read-modify-write instructions.

Table 9-2. Read-Modify-Write Instructions

Instruction	Mnemonic
Arithmetic Shift Left	ASL
Arithmetic Shift Right	ASR
Clear Bit in Memory	BCLR
Set Bit in Memory	BSET
Clear	CLR
Complement (One's Complement)	COM
Decrement	DEC
Increment	INC
Logical Shift Left	LSL
Logical Shift Right	LSR
Negate (Two's Complement)	NEG
Rotate Left through Carry Bit	ROL
Rotate Right through Carry Bit	ROR
Test for Negative or Zero	TST

9.2.3 Jump/Branch Instructions

Jump instructions allow the CPU to interrupt the normal sequence of the program counter. The unconditional jump instruction (JMP) and the jump to subroutine instruction (JSR) have no register operand. 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 9-3 lists the jump and branch instructions.

Table 9-3. Jump and Branch Instructions

Instruction	Mne-monic
Branch if Carry Bit Clear	BCC
Branch if Carry Bit Set	BCS
Branch if Equal	BEQ
Branch if Half-Carry Bit Clear	BHCC
Branch if Half-Carry Bit Set	BHCS
Branch if Higher	BHI
Branch if Higher or Same	BHS
Branch if $\overline{\text{IRQ}}$ Pin High	BIH
Branch if $\overline{\text{IRQ}}$ Pin 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 Never	BRN
Branch if Bit Set	BRSET
Branch to Subroutine	BSR
Unconditional Jump	JMP
Jump to Subroutine	JSR

9.2.4 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 9-4 lists these instructions.

Table 9-4. Bit Manipulation Instructions

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

9.2.5 Control Instructions

These register reference instructions control CPU operation during program execution. Control instructions, listed in Table 9-5, use inherent addressing.

Table 9-5. Control Instructions

Instruction	Mnemonic
Clear Carry Bit	CLC
Clear Interrupt Mask	CLI
No Operation	NOP
Reset Stack Pointer	RSP
Return from Interrupt	RTI
Return from Subroutine	RTS
Set Carry Bit	SEC
Set Interrupt Mask	SEI
Stop Oscillator and Enable $\overline{\text{IRQ}}$ Pin	STOP
Software Interrupt	SWI
Transfer Accumulator to Index Register	TAX
Transfer Index Register to Accumulator	TXA
Stop CPU Clock and Enable Interrupts	WAIT

9.3 Instruction Set Summary

Table 9-6 is an alphabetical list of all MC68HC05J1A instructions and shows the effect of each instruction on the condition code register.

Table 9-6. Instruction Set Summary

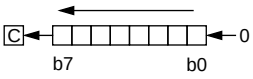
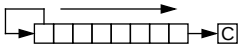
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X	Add with Carry	A ← (A) + (M) + (C)	↕	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A9 B9 C9 D9 E9 F9	ii dd hh ll ee ff ff	2 3 4 5 4 3
ADD #*opr* ADD *opr* ADD *opr* ADD *opr*,X ADD *opr*,X ADD ,X	Add without Carry	A ← (A) + (M)	↕	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	AB BB CB DB EB FB	ii dd hh ll ee ff ff	2 3 4 5 4 3
AND #*opr* AND *opr* AN*D opr* AND *opr*,X AND *opr*,X AND ,X	Logical AND	A ← (A) ∧ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A4 B4 C4 D4 E4 F4	ii dd hh ll ee ff ff	2 3 4 5 4 3
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X	Arithmetic Shift Left (Same as LSL)		—	—	↕	↕	↕	DIR INH INH IX1 IX	38 48 58 68 78	dd ff 	5 3 3 6 5
ASR *opr* ASRA ASRX ASR *opr*,X ASR ,X	Arithmetic Shift Right		—	—	↕	↕	↕	DIR INH INH IX1 IX	37 47 57 67 77	dd ff 	5 3 3 6 5
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + 2 + *rel* ? C = 0	—	—	—	—	—	REL	24	rr	3
BCLR *n opr*	Clear Bit n	Mn ← 0	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? C = 1	—	—	—	—	—	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? Z = 1	—	—	—	—	—	REL	27	rr	3

Table 9-6. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
BHCC *rel*	Branch if Half-Carry Bit Clear	PC ← (PC) + 2 + *rel* ? H = 0	—	—	—	—	—	REL	28	rr	3
BHCS *rel*	Branch if Half-Carry Bit Set	PC ← (PC) + 2 + *rel* ? H = 1	—	—	—	—	—	REL	29	rr	3
BHI *rel*	Branch if Higher	PC ← (PC) + 2 + *rel* ? C ∨ Z = 0	—	—	—	—	—	REL	22	rr	3
BHS *rel*	Branch if Higher or Same	PC ← (PC) + 2 + *rel* ? C = 0	—	—	—	—	—	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	—	—	—	—	—	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	—	—	—	—	—	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X	Bit Test Accumulator with Memory Byte	(A) ∧ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A5 B5 C5 D5 E5 F5	ii dd hh ll ee ff ff p	2 3 4 5 4 3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? C = 1	—	—	—	—	—	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? C ∨ Z = 1	—	—	—	—	—	REL	23	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? I = 0	—	—	—	—	—	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? N = 1	—	—	—	—	—	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? I = 1	—	—	—	—	—	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? Z = 0	—	—	—	—	—	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? N = 0	—	—	—	—	—	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel* ? 1 = 1	—	—	—	—	—	REL	20	rr	3
BRCLR *n opr rel*	Branch if bit n clear	PC ← (PC) + 2 + *rel* ? Mn = 0	—	—	—	—	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRSET *n opr rel*	Branch if Bit n Set	PC ← (PC) + 2 + *rel* ? Mn = 1	—	—	—	—	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2 + *rel* ? 1 = 0	—	—	—	—	—	REL	21	rr	3

Table 9-6. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
BSET *n opr*	Set Bit n	Mn ← 1	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	—	—	—	—	—	REL	AD	rr	6
CLC	Clear Carry Bit	C ← 0	—	—	—	—	0	INH	98		2
CLI	Clear Interrupt Mask	I ← 0	—	0	—	—	—	INH	9A		2
CLR *opr* CLRA CLRX CLR *opr*,X CLR ,X	Clear Byte	M ← $00 A ← $00 X ← $00 M ← $00 M ← $00	—	—	0	1	—	DIR INH INH IX1 IX	3F 4F 5F 6F 7F	dd ff 	5 3 3 6 5
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X	Compare Accumulator with Memory Byte	(A) – (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A1 B1 C1 D1 E1 F1	ii dd hh ll ee ff ff 	2 3 4 5 4 3
COM *opr* COMA COMX COM *opr*,X COM ,X	Complement Byte (One's Complement)	M ← ($\overline{M}$) = $FF – (M) A ← ($\overline{A}$) = $FF – (M) X ← ($\overline{X}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	—	—	↕	↕	1	DIR INH INH IX1 IX	33 43 53 63 73	dd ff 	5 3 3 6 5
CPX #*opr* CPX *opr* CPX *opr* CPX *opr*,X CPX *opr*,X CPX ,X	Compare Index Register with Memory Byte	(X) – (M)	—	—	↕	↕	1	IMM DIR EXT IX2 IX1 IX	A3 B3 C3 D3 E3 F3	ii dd hh ll ee ff ff 	2 3 4 5 4 3
DEC *opr* DECA DECX DEC *opr*,X DEC ,X	Decrement Byte	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1	—	—	↕	↕	—	DIR INH INH IX1 IX	3A 4A 5A 6A 7A	dd ff 	5 3 3 6 5
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X	EXCLUSIVE OR Accumulator with Memory Byte	A ← (A) ⊕ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A8 B8 C8 D8 E8 F8	ii dd hh ll ee ff ff 	2 3 4 5 4 3

Table 9-6. Instruction Set Summary (Continued)

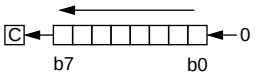
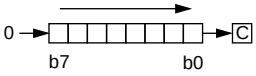
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
INC *opr* INCA INCX INC *opr*,X INC ,X	Increment Byte	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1	—	—	↕	↕	—	DIR INH INH IX1 IX	3C 4C 5C 6C 7C	dd ff 	5 3 3 6 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Unconditional Jump	PC ← Jump Address	—	—	—	—	—	DIR EXT IX2 IX1 IX	BC C C D C EC FC	 dd hh ll ee ff ff 	 2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + n (n = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Conditional Address	—	—	—	—	—	DIR EXT IX2 IX1 IX	BD C D D D ED FD	 dd hh ll ee ff ff 	 5 6 7 6 5
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X	Load Accumulator with Memory Byte	A ← (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A6 B6 C6 D6 E6 F6	ii dd hh ll ee ff ff 	2 3 4 5 4 3
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X	Load Index Register with Memory Byte	X ← (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	AE BE CE DE EE FE	ii dd hh ll ee ff ff 	2 3 4 5 4 3
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X	Logical Shift Left (Same as ASL)		—	—	↕	↕	↕	DIR INH INH IX1 IX	38 48 58 68 78	dd ff 	5 3 3 6 5
LSR *opr* LSRA LSRX LSR *opr*,X LSR ,X	Logical Shift Right		—	—	0	↕	↕	DIR INH INH IX1 IX	34 44 54 64 74	dd ff 	5 3 3 6 5
MUL	Unsigned Multiply	X : A ← (X) × (A)	0	—	—	—	0	INH	42		11
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X	Negate Byte (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	—	—	↕	↕	↕	DIR INH INH IX1 IX	30 40 50 60 70	ii ff 	5 3 3 6 5
NOP	No Operation		—	—	—	—	—	INH	9D		2

Table 9-6. Instruction Set Summary (Continued)

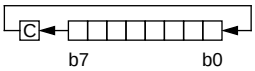
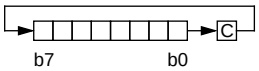
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X	Logical OR Accumulator with Memory	A ← (A) ∨ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	AA BA CA DA EA FA	ii dd hh ll ee ff ff	2 3 4 5 4 3
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X	Rotate Byte Left through Carry Bit		—	—	↕	↕	↕	DIR INH INH IX1 IX	39 49 59 69 79	dd ff	5 3 3 6 5
ROR *opr* RORA RORX ROR *opr*,X ROR ,X	Rotate Byte Right through Carry Bit		—	—	↕	↕	↕	DIR INH INH IX1 IX	36 46 56 66 76	dd ff	5 3 3 6 5
RSP	Reset Stack Pointer	SP ← $00FF	—	—	—	—	—	INH	9C		2
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	INH	80		6
RTS	Return from Subroutine	SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)						INH			
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X	Subtract Memory Byte and Carry Bit from Accumulator	A ← (A) – (M) – (C)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A2 B2 C2 D2 E2 F2	ii dd hh ll ee ff ff	2 3 4 5 4 3
SEC	Set Carry Bit	C ← 1	—	—	—	—	1	INH	99		2
SEI	Set Interrupt Mask	I ← 1	—	1	—	—	—	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X	Store Accumulator in Memory	M ← (A)	—	—	↕	↕	—	DIR EXT IX2 IX1 IX	B7 C7 D7 E7 F7	dd hh ll ee ff ff	4 5 6 5 4
STOP	Stop Oscillator and Enable $\overline{\text{IRQ}}$ Pin		—	0	—	—	—	INH	8E		2
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X	Store Index Register In Memory	M ← (X)	—	—	↕	↕	—	DIR EXT IX2 IX1 IX	BF CF DF EF FF	dd hh ll ee ff ff	4 5 6 5 4

Table 9-6. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X	Subtract Memory Byte from Accumulator	A ← (A) – (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A0 B0 C0 D0 E0 F0	ii dd hh ll ee ff ff	2 3 4 5 4 3
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	—	1	—	—	—	INH	83		10
TAX	Transfer Accumulator to Index Register	X ← (A)	—	—	—	—	—	INH	97		2
TST *opr* TSTA TSTX TST *opr*,X TST ,X	Test Memory Byte for Negative or Zero	(M) – $00	—	—	—	—	—	DIR INH INH IX1 IX	3D 4D 5D 6D 7D	dd ff	4 3 3 5 4
TXA	Transfer Index Register to Accumulator	A ← (X)	—	—	—	—	—	INH	9F		2
WAIT	Stop CPU Clock and Enable Interrupts		—	↕	—	—	—	INH	8F		2

A	Accumulator	*opr*	Operand (one or two bytes)
C	Carry/borrow flag	PC	Program counter
CCR	Condition code register	PCH	Program counter high byte
dd	Direct address of operand	PCL	Program counter low byte
dd rr	Direct address of operand and relative offset of branch instruction	REL	Relative addressing mode
DIR	Direct addressing mode	*rel*	Relative program counter offset byte
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing	rr	Relative program counter offset byte
EXT	Extended addressing mode	SP	Stack pointer
ff	Offset byte in indexed, 8-bit offset addressing	X	Index register
H	Half-carry flag	Z	Zero flag
hh ll	High and low bytes of operand address in extended addressing	#	Immediate value
I	Interrupt mask	∧	Logical AND
ii	Immediate operand byte	∨	Logical OR
IMM	Immediate addressing mode	⊕	Logical EXCLUSIVE OR
INH	Inherent addressing mode	( )	Contents of
IX	Indexed, no offset addressing mode	–( )	Negation (two's complement)
IX1	Indexed, 8-bit offset addressing mode	←	Loaded with
IX2	Indexed, 16-bit offset addressing mode	?	If
M	Memory location	:	Concatenated with
N	Negative flag	↕	Set or cleared
n	Any bit	—	Not affected

Table 9-7. 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	
MSB / LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSB / LSB
0	5 BRSET0 3 DIR	5 BSET0 2 DIR	3 BRA 2 REL	5 NEG 2 DIR	3 NEGA 1 INH	3 NEGX 1 INH	6 NEG 2 IX1	5 NEG 1 IX	9 RTI 1 INH		2 SUB 2 IMM	3 SUB 2 DIR	4 SUB 3 EXT	5 SUB 3 IX2	4 SUB 2 IX1	3 SUB 1 IX	0
1	5 BRCLR0 3 DIR	5 BCLR0 2 DIR	3 BRN 2 REL						6 RTS 1 INH		2 CMP 2 IMM	3 CMP 2 DIR	4 CMP 3 EXT	5 CMP 3 IX2	4 CMP 2 IX1	3 CMP 1 IX	1
2	5 BRSET1 3 DIR	5 BSET1 2 DIR	3 BHI 2 REL		11 MUL 1 INH						2 SBC 2 IMM	3 SBC 2 DIR	4 SBC 3 EXT	5 SBC 3 IX2	4 SBC 2 IX1	3 SBC 1 IX	2
3	5 BRCLR1 3 DIR	5 BCLR1 2 DIR	3 BLS 2 REL	5 COM 2 DIR	3 COMA 1 INH	3 COMX 1 INH	6 COM 2 IX1	5 COM 1 IX	10 SWI 1 INH		2 CPX 2 IMM	3 CPX 2 DIR	4 CPX 3 EXT	5 CPX 3 IX2	4 CPX 2 IX1	3 CPX 1 IX	3
4	5 BRSET2 3 DIR	5 BSET2 2 DIR	3 BCC 2 REL	5 LSR 2 DIR	3 LSRA 1 INH	3 LSRX 1 INH	6 LSR 2 IX1	5 LSR 1 IX			2 AND 2 IMM	3 AND 2 DIR	4 AND 3 EXT	5 AND 3 IX2	4 AND 2 IX1	3 AND 1 IX	4
5	5 BRCLR2 3 DIR	5 BCLR2 2 DIR	3 BCS/BLO 2 REL								2 BIT 2 IMM	3 BIT 2 DIR	4 BIT 3 EXT	5 BIT 3 IX2	4 BIT 2 IX1	3 BIT 1 IX	5
6	5 BRSET3 3 DIR	5 BSET3 2 DIR	3 BNE 2 REL	5 ROR 2 DIR	3 RORA 1 INH	3 RORX 1 INH	6 ROR 2 IX1	5 ROR 1 IX			2 LDA 2 IMM	3 LDA 2 DIR	4 LDA 3 EXT	5 LDA 3 IX2	4 LDA 2 IX1	3 LDA 1 IX	6
7	5 BRCLR3 3 DIR	5 BCLR3 2 DIR	3 BEQ 2 REL	5 ASR 2 DIR	3 ASRA 1 INH	3 ASRX 1 INH	6 ASR 2 IX1	5 ASR 1 IX		2 TAX 1 INH		4 STA 2 DIR	5 STA 3 EXT	6 STA 3 IX2	5 STA 2 IX1	4 STA 1 IX	7
8	5 BRSET4 3 DIR	5 BSET4 2 DIR	3 BHCC 2 REL	5 ASL/LSL 2 DIR	3 ASLA/LSLA 1 INH	3 ASLX/LSLX 1 INH	6 ASL/LSL 2 IX1	5 ASL/LSL 1 IX		2 CLC 1 INH	2 EOR 2 IMM	3 EOR 2 DIR	4 EOR 3 EXT	5 EOR 3 IX2	4 EOR 2 IX1	3 EOR 1 IX	8
9	5 BRCLR4 3 DIR	5 BCLR4 2 DIR	3 BHCS 2 REL	5 ROL 2 DIR	3 ROLA 1 INH	3 ROLX 1 INH	6 ROL 2 IX1	5 ROL 1 IX		2 SEC 1 INH	2 ADC 2 IMM	3 ADC 2 DIR	4 ADC 3 EXT	5 ADC 3 IX2	4 ADC 2 IX1	3 ADC 1 IX	9
A	5 BRSET5 3 DIR	5 BSET5 2 DIR	3 BPL 2 REL	5 DEC 2 DIR	3 DECA 1 INH	3 DECX 1 INH	6 DEC 2 IX1	5 DEC 1 IX		2 CLI 1 INH	2 ORA 2 IMM	3 ORA 2 DIR	4 ORA 3 EXT	5 ORA 3 IX2	4 ORA 2 IX1	3 ORA 1 IX	A
B	5 BRCLR5 3 DIR	5 BCLR5 2 DIR	3 BMI 2 REL							2 SEI 1 INH	2 ADD 2 IMM	3 ADD 2 DIR	4 ADD 3 EXT	5 ADD 3 IX2	4 ADD 2 IX1	3 ADD 1 IX	B
C	5 BRSET6 3 DIR	5 BSET6 2 DIR	3 BMC 2 REL	5 INC 2 DIR	3 INCA 1 INH	3 INCX 1 INH	6 INC 2 IX1	5 INC 1 IX		2 RSP 1 INH		2 JMP 2 DIR	3 JMP 3 EXT	4 JMP 3 IX2	3 JMP 2 IX1	2 JMP 1 IX	C
D	5 BRCLR6 3 DIR	5 BCLR6 2 DIR	3 BMS 2 REL	4 TST 2 DIR	3 TSTA 1 INH	3 TSTX 1 INH	5 TST 2 IX1	4 TST 1 IX		2 NOP 1 INH	6 BSR 2 REL	5 JSR 2 DIR	6 JSR 3 EXT	7 JSR 3 IX2	6 JSR 2 IX1	5 JSR 1 IX	D
E	5 BRSET7 3 DIR	5 BSET7 2 DIR	3 BIL 2 REL						2 STOP 1 INH		2 LDX 2 IMM	3 LDX 2 DIR	4 LDX 3 EXT	5 LDX 3 IX2	4 LDX 2 IX1	3 LDX 1 IX	E
F	5 BRCLR7 3 DIR	5 BCLR7 2 DIR	3 BIH 2 REL	5 CLR 2 DIR	3 CLRA 1 INH	3 CLRX 1 INH	6 CLR 2 IX1	5 CLR 1 IX	2 WAIT 1 INH	2 TXA 1 INH		4 STX 2 DIR	5 STX 3 EXT	6 STX 3 IX2	5 STX 2 IX1	4 STX 1 IX	F

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

LSB of Opcode in Hexadecimal

MSB / LSB	0
0	5 BRSET0 3 DIR

MSB of Opcode in Hexadecimal
Number of Cycles
Opcode Mnemonic
Number of Bytes/Addressing Mode

SECTION 10
ELECTRICAL SPECIFICATIONS

This section contains electrical and timing specifications.

10.1 Maximum Ratings

CAUTION

To avoid damage to the MCU, do not exceed the values listed in Table 10-1. Keep V_{IN} and V_{OUT} within the range V_{SS} (V_{IN} or V_{OUT}) V_{DD}.

Table 10-1. Maximum Ratings(1)

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
Current Drain Per Pin (Excluding V_{DD} and V_{SS})	I	25	mA

1. Maximum values are not guaranteed operating values.

10.2 Thermal Characteristics

Table 10-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Maximum Junction Temperature	T_J	150	C
Thermal Resistance MC68HC05J1AP(1) MC68HC05J1ADW(2)	θ_{TA}	 68 85	 C/W C/W
Operating Temperature Range MC68HC05J1AP, DW MC68HC05J1AC(3)P, CDW MC68HC05J1AV(4)P MC68HC05J1AVDW	T_A	 0 to +70 –40 to +85 –40 to +105 –40 to +105	 C
Storage Temperature Range	T_{STG}	–65 to +150	C

1. P = Plastic dual in-line package (PDIP).
2. DW = Small outline integrated circuit (SOIC).
3. C = Extended temperature range (–40 C to +85 C).

4. V = Automotive temperature range (–40 C to +105 C).

10.3 Power Considerations

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

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

where:

T_A = Ambient temperature, C

θ_{JA} = Package thermal resistance, junction -to-ambient, C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{DD} \times V_{DD}$ 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 (neglecting $P_{I/O}$):

$$P_D = K + (T_J + 273\ \ C) \quad (2)$$

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

$$K = P_D \times (T_A + 273\ \ C) + \theta_{JA} \times (P_D)^2 \quad (3)$$

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.

10.4 DC Electrical Characteristics

Table 10-3. DC Electrical Characteristics (V_{DD} = 5.0 V)[1]

Characteristic[2]	Symbol	Min	Typ	Max	Unit
Output Voltage I_{LOAD} = 10.0 µA I_{LOAD} = –10.0 µA	 V_{OL} V_{OH}	 — V_{DD} – 0.1	 — —	 0.1 —	 V V
Output High Voltage (I_{LOAD} = –0.8 mA) PA7–PA0, PB5–PB0	V_{OH}	V_{DD} – 0.8	—	—	V
Output Low Voltage PA3–PA0 (I_{LOAD} = 1.6 mA) PA7–PA4 (I_{LOAD} = 8.0 mA)	V_{OL}	— —	— —	0.4 0.4	V V
Input High Voltage PA7–PA0, PB5–PB0, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input Low Voltage PA7–PA0, PB5–PB0, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IL}	V_{SS}	—	0.2 × V_{DD}	V
Supply Current Run[3] Wait[4] Stop[5] 25 C –40 C to +85 C	 I_{DD} 	 — — — —	 3.0 1.6 0.2 2.0	 4.0 2.5 10 20	 mA mA µA µA
I/O Ports Hi-Z Leakage Current PA7–PA0, PB5–PB0 (Pulldown Device Off)	I_{IL}	—	—	10	µA
Input Pulldown Current PA7–PA0, PB5–PB0 (Pulldown Device On)	I_{IL}	50	100	200	µA
Input Current $\overline{RESET}$, $\overline{IRQ}$, OSC1	I_{IN}	—	—	1	µA
Capacitance PA7–PA0, PB5–PB0 (Input or Output) $\overline{RESET}$, $\overline{IRQ}$, OSC1, OSC2	 C_{OUT} C_{IN}	 — —	 — —	 12 8	 pF pF
Oscillator Internal Resistor (Crystal/Ceramic Resonator Mask Option)	R_{OSC}	1.0	2.0	3.0	M

1. V_{DD} = 5.0 Vdc 10%; V_{SS} = 0 Vdc; T_A = –40 C to +85 C
2. Values reflect average measurements at midpoint of voltage range at 25 C
3. Run (operating) I_{DD} measured using external square wave clock source (f_{OSC} = 4.2 MHz) with 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} measured using external square wave clock source (f_{OSC} = 4.2 MHz) with all inputs 0.2 V from rail and only the timer active. No dc loads. Less than 50 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. V_{IL} = 0.2 V. V_{IH} = V_{DD} – 0.2 V. OSC2 capacitance linearly affects wait I_{DD}.
5. Stop I_{DD} measured with OSC1 = V_{SS}. All ports configured as inputs. V_{IL} = 0.2 V. V_{IH} = V_{DD} – 0.2 V.

Table 10-4. DC Electrical Characteristics (V_{DD} = 3.3 V)[1]

Characteristic[2]	Symbol	Min	Typ	Max	Unit
Output Voltage I_{LOAD} 10.0 µA I_{LOAD} –10.0 µA	 V_{OL} V_{OH}	 — V_{DD} – 0.1	 — —	 0.1 —	 V V
Output High Voltage (I_{LOAD} = –0.2 mA) PA7–PA0, PB5–PB0	V_{OH}	V_{DD} – 0.3	—	—	V
Output Low Voltage PA3–PA0 (I_{LOAD} = –0.4 mA) PA7–PA4 (I_{LOAD} = 5.0 mA)	V_{OL}	 —	 —	0.3 0.3	V V
Input High Voltage PA7–PA0, PB5–PB0, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input Low Voltage PA7–PA0, PB5–PB0, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IL}	V_{SS}	—	0.2 × V_{DD}	V
Supply Current Run[3] Wait[4] Stop[5] 25 C –40 C to +85 C	 I_{DD}	 — — — —	 1.0 0.5 0.1 1	 2.0 1.0 5 10	 mA mA µA µA
I/O Ports Hi-Z Leakage Current PA7–PA0, PB5–PB0 (Pulldown Device Off)	I_{IL}	—	—	10	µA
Input Pulldown Current PA7–PA0, PB5–PB0 (Pulldown Device On)	I_{IL}	20	40	100	µA
Input Current $\overline{RESET}$, $\overline{IRQ}$, OSC1	I_{IN}	—	—	1	µA
Capacitance PA7–PA0, PB5–PB0 (Input or Output) $\overline{RESET}$, $\overline{IRQ}$, OSC1, OSC2	 C_{OUT} C_{IN}	 — —	 — —	 12 8	 pF pF
Oscillator Internal Resistor (Crystal/Ceramic Resonator Mask Option)	R_{OSC}	1.0	2.0	3.0	M

1. V_{DD} = 3.3 Vdc 10%; V_{SS} = 0 Vdc; T_A = –40 C to +85 C
2. Values reflect average measurements at midpoint of voltage range at 25 C
3. Run (operating) I_{DD} measured using external square wave clock source (f_{OSC} = 2.0 MHz) with 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} measured using external square wave clock source (f_{OSC} = 2.0 MHz) with all inputs 0.2 V from rail and only the timer active. No dc loads. Less than 50 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. V_{IL} = 0.2 V; V_{IH} = V_{DD} – 0.2 V. OSC2 capacitance linearly affects wait I_{DD}.
5. Stop I_{DD} measured with OSC1 = V_{SS}. All ports configured as inputs. V_{IL} = 0.2 V. V_{IH} = V_{DD} – 0.2 V.

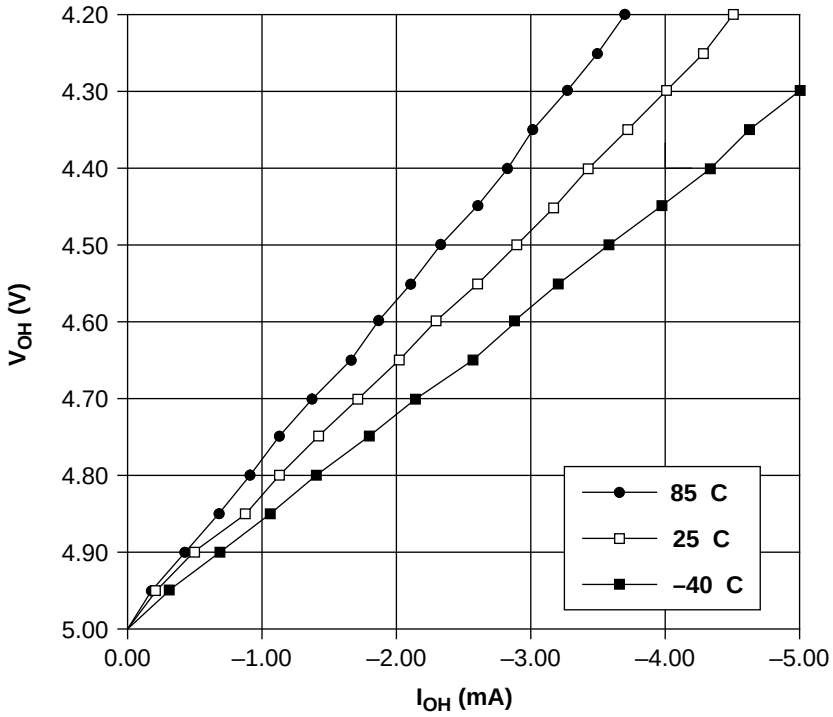


Figure 10-1. V_{OH}/I_{OH} (V_{DD} = 5.0 V)

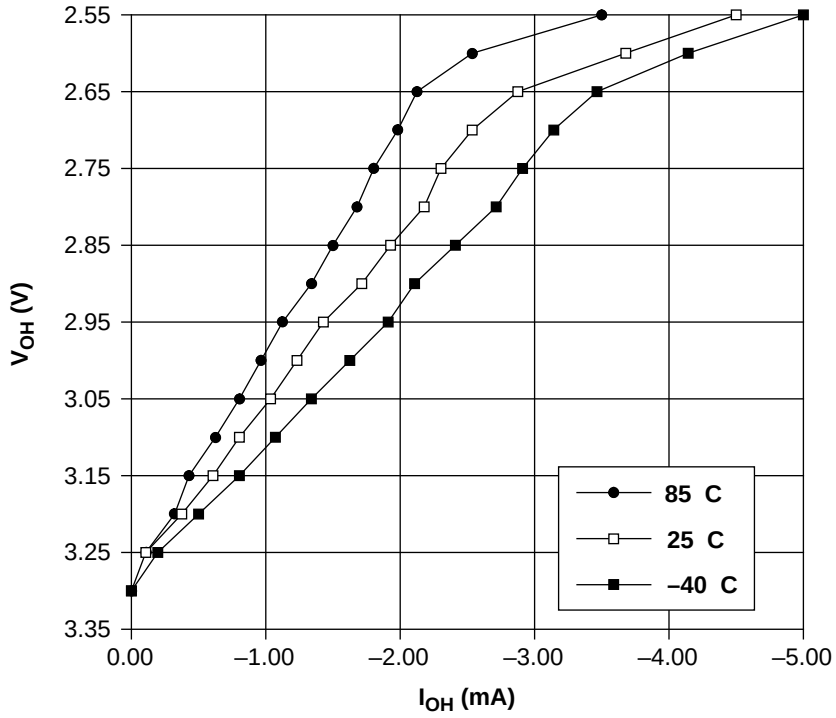


Figure 10-2. V_{OH}/I_{OH} (V_{DD} = 3.3 V)

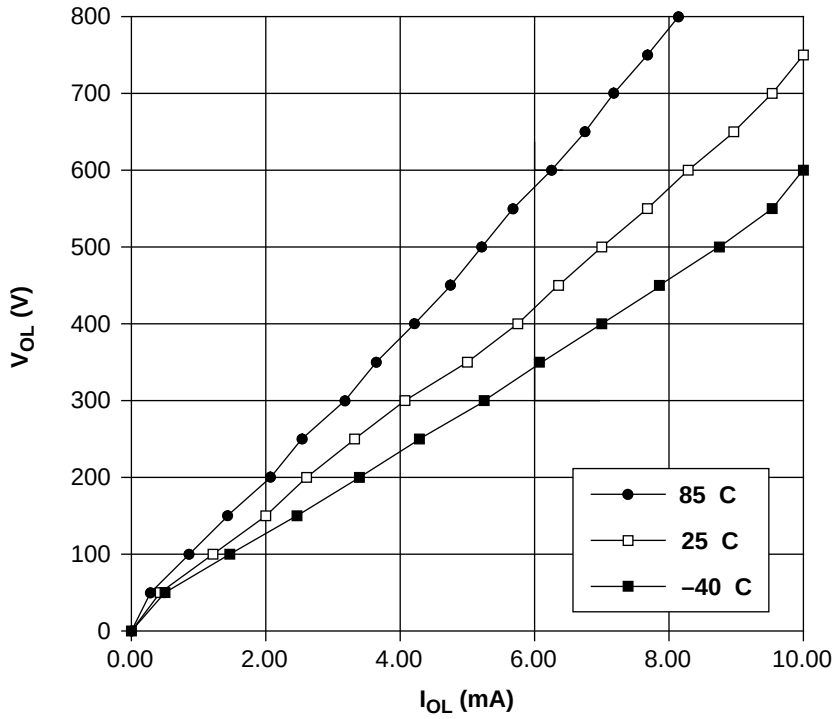


Figure 10-3. V_{OL}/I_{OL} (V_{DD} = 5.0 V)

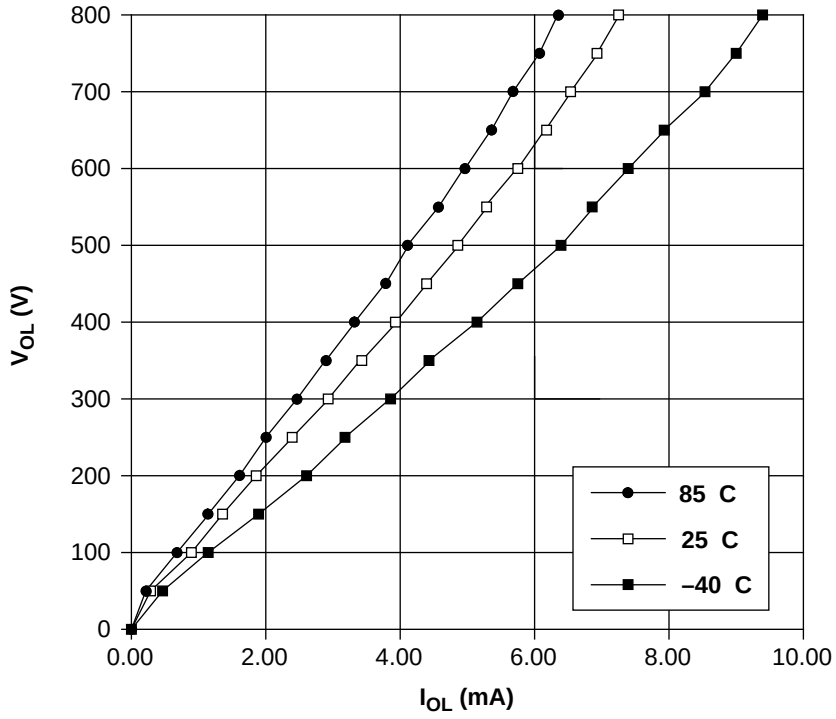


Figure 10-4. V_{OL}/I_{OL} (V_{DD} = 3.3 V)

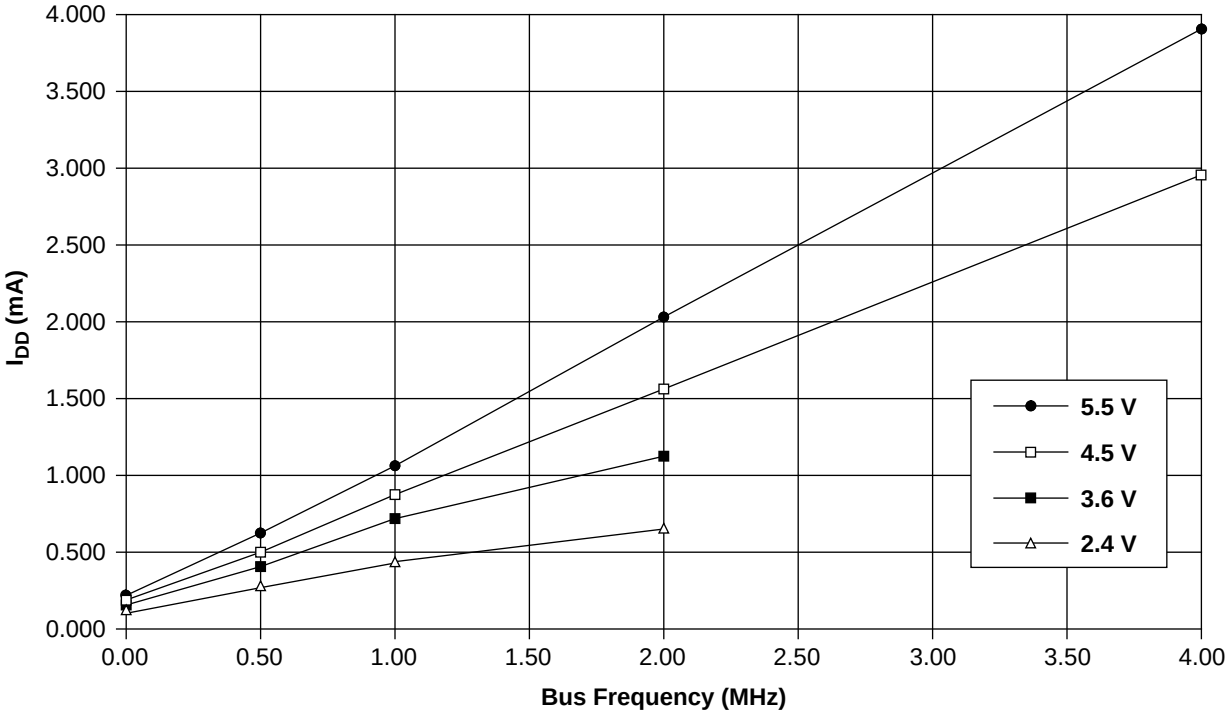


Figure 10-5. Typical Operating I_{DD} (25 C)

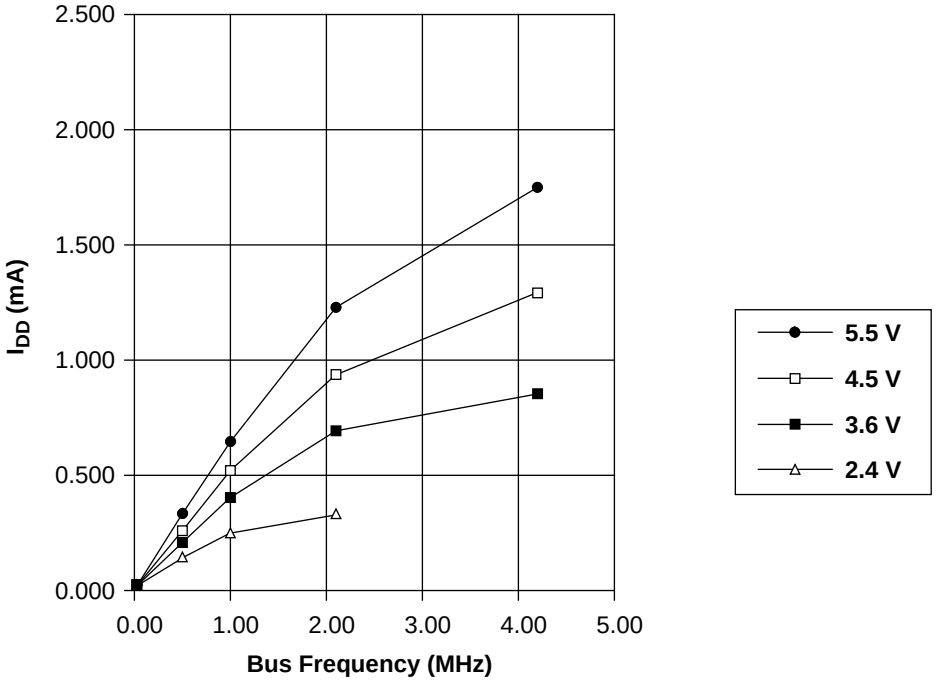


Figure 10-6. Typical Wait Mode I_{DD} (25 C)

10.5 Control Timing

Table 10-5. Control Timing (V_{DD} = 5.0 V)(1)

Characteristic	Symbol	Min	Max	Unit
Oscillator Frequency Crystal/Ceramic Resonator Mask Option(2) RC Oscillator Mask Option External Clock Mask Option	f_{OSC}	— dc —	 4.2 4.2 4.2	MHz
Internal Operating Frequency (f_{OSC} Ö2) Crystal Oscillator Ceramic Resonator RC Oscillator External Clock	f_{OP}	— dc —	 2.1 2.1 2.1 2.1	MHz
Cycle Time (1 Öf_{OP})	t_{CYC}	476	—	ns
$\overline{\text{RESET}}$ Pulse Width Low (Edge-Triggered)	t_{RL}	1.5	—	t_{CYC}
Timer Resolution(3)	t_{RESL}	4.0	—	t_{CYC}
$\overline{\text{IRQ}}$ Interrupt Pulse Width Low (Edge-Triggered)	t_{ILIH}	125	—	ns
$\overline{\text{IRQ}}$ Interrupt Pulse Period	t_{ILIL}	(4)	—	t_{CYC}
PA3–PA0 Interrupt Pulse Width High (Edge-Triggered)	t_{IHIL}	125	—	ns
PA3–PA0 Interrupt Pulse Period	t_{IHIH}	(4)	—	t_{CYC}
OSC1 Pulse Width	t_{OH}, t_{OL}	200	—	ns

1. V_{DD} = 5.0 Vdc 10%; V_{SS} = 0 Vdc; T_A = T_L to T_H.
2. Use only AT-cut crystals.
3. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
4. The minimum period t_{ILIL} or t_{IHIH} should not be less than the number of cycles required to execute the interrupt service routine plus 19 t_{CYC}.

Table 10-6. Control Timing (V_{DD} = 3.3 V)(1)

Characteristic	Symbol	Min	Max	Unit
Oscillator Frequency Crystal/Ceramic Resonator(2) RC Oscillator External Clock	f_{OSC}	— dc —	2.0 2.0 2.0	MHz
Internal Operating Frequency (f_{OSC} ÷2) Crystal Oscillator Ceramic Resonator RC Oscillator External Clock	f_{OP}	— dc —	1.0 1.0 1.0	MHz
Cycle Time (1 ÷ f_{OP})	t_{CYC}	1000	—	ns
$\overline{RESET}$ Pulse Width Low (Edge-Triggered)	t_{RL}	1.5	—	t_{CYC}
Timer Resolution(3)	t_{RESL}	4.0	—	t_{CYC}
$\overline{IRQ}$ Interrupt Pulse Width Low (Edge-Triggered)	t_{ILIH}	250	—	ns
$\overline{IRQ}$ Interrupt Pulse Period	t_{ILIL}	(4)	—	t_{CYC}
PA3–PA0 Interrupt Pulse Width High (Edge-Triggered)	t_{IHIL}	250	—	t_{CYC}
PA3–PA0 Interrupt Pulse Period	t_{IHIH}	(4)	—	t_{CYC}
OSC1 Pulse Width	t_{OH},t_{OL}	400	—	ns

1. V_{DD} = 3.3 Vdc 10%; V_{SS} = 0 Vdc; T_A = T_L to T_H.
2. Use only AT-cut crystals.
3. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
4. The minimum period t_{ILIL} or t_{IHIH} should not be less than the number of cycles required to execute the interrupt service routine plus 19 t_{CYC}.

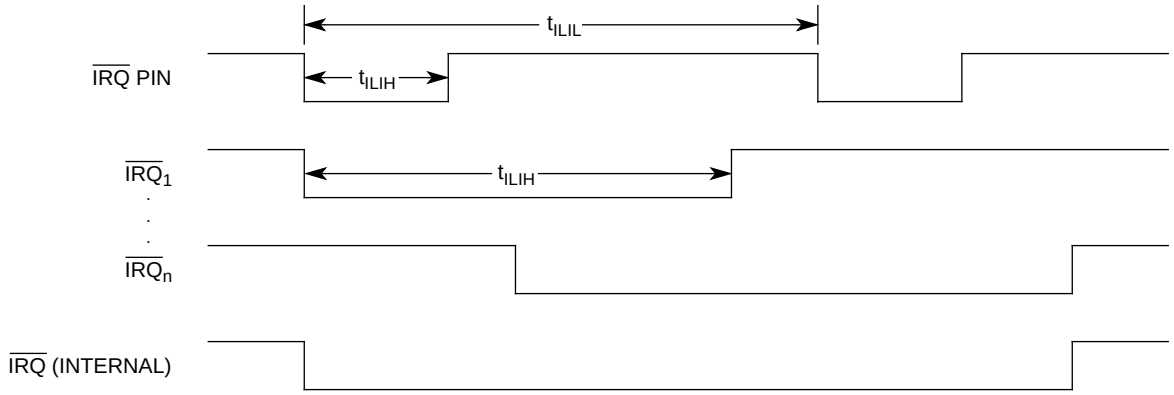
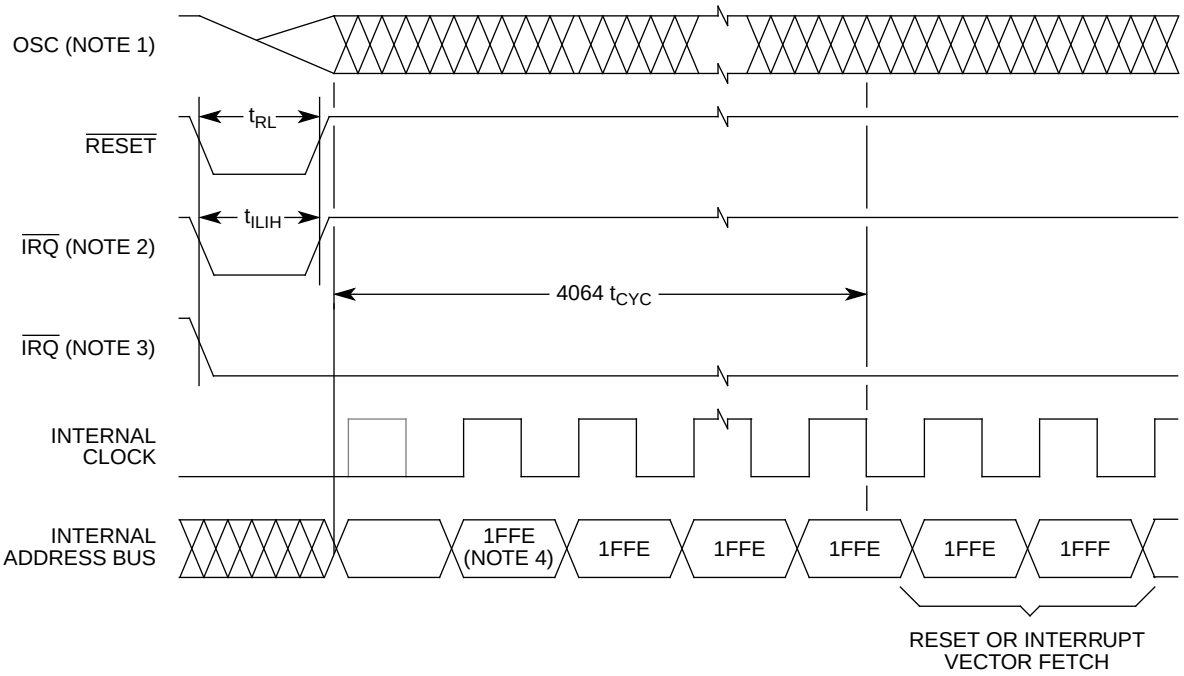


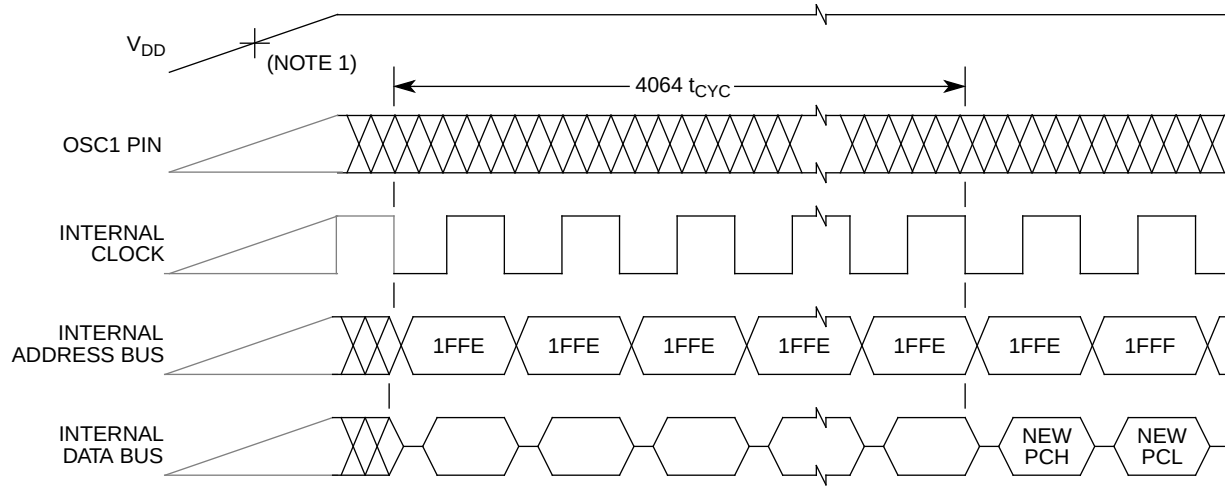
Figure 10-7. External Interrupt Timing



NOTES:

1. Internal clocking from OSC1 pin.
2. Edge-triggered external interrupt mask option.
3. Edge- and level-triggered external interrupt mask option.
4. Reset vector shown as example.

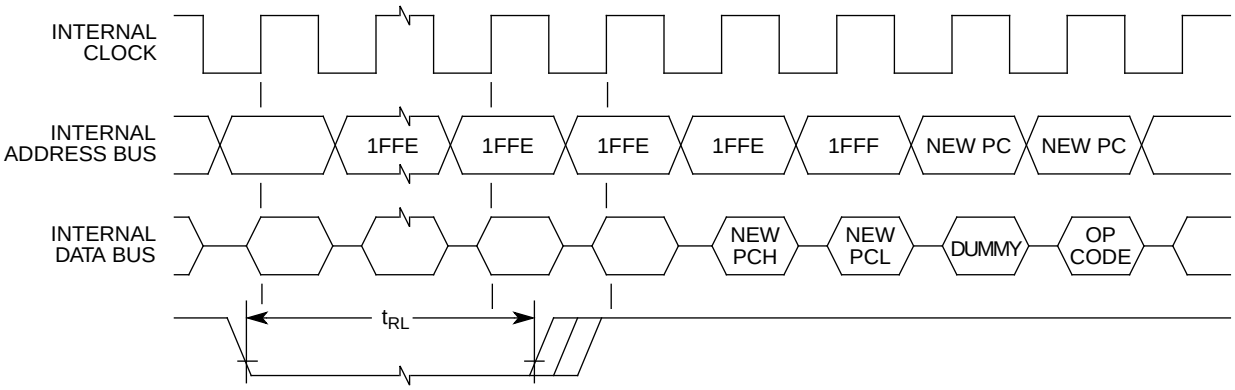
Figure 10-8. Stop Mode Recovery Timing



NOTES:

1. Power-on reset threshold is typically between 1 V and 2 V.
2. Internal clock, internal address bus, and internal data bus are not available externally.

Figure 10-9. Power-On Reset Timing



NOTES:

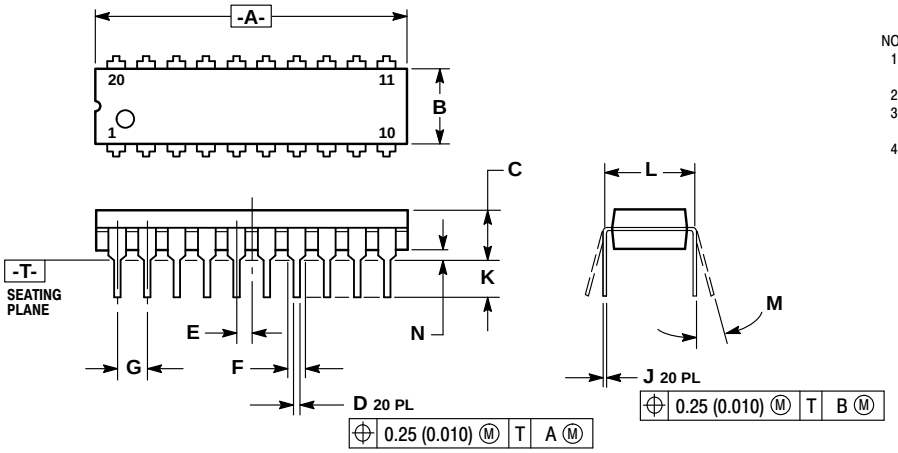
1. Internal clock, internal address bus, and internal data bus are not available externally.
2. The next rising edge of the internal clock after the rising edge of $\overline{\text{RESET}}$ initiates the reset sequence.

Figure 10-10. External Reset Timing

SECTION 11
MECHANICAL SPECIFICATIONS

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

11.1 PDIP



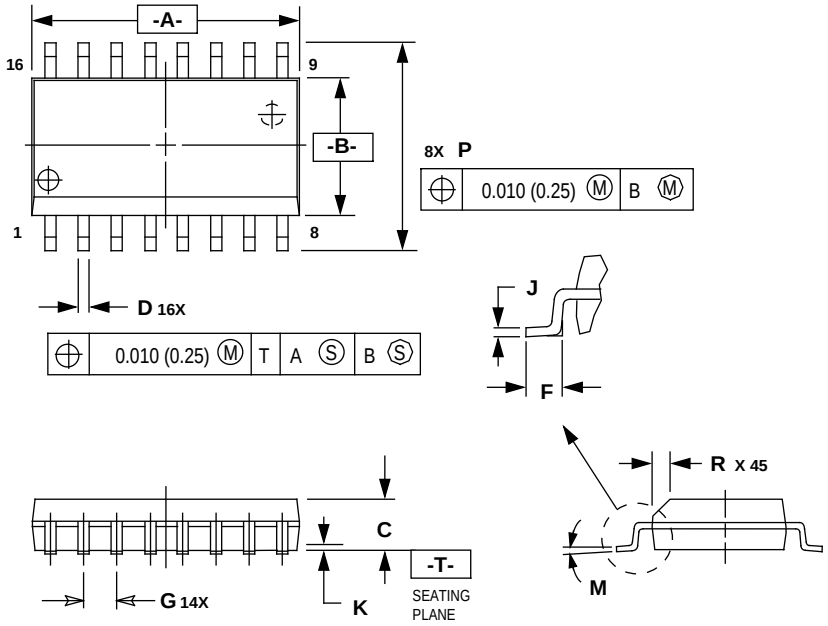
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	1.010	1.070	25.66	27.17
B	0.240	0.260	6.10	6.60
C	0.150	0.180	3.81	4.57
D	0.015	0.022	0.39	0.55
E	0.050 BSC		1.27 BSC	
F	0.050	0.070	1.27	1.77
G	0.100 BSC		2.54 BSC	
J	0.008	0.015	0.21	0.38
K	0.110	0.140	2.80	3.55
L	0.300 BSC		7.62 BSC	
M	0°	15°	0°	15°
N	0.020	0.040	0.51	1.01

CASE 738-03

Figure 11-1. MC68HC05J1AP

11.2 SOIC



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	10.15	10.45	0.400	0.411
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	0.50	0.90	0.020	0.035
G	1.27 BSC		0.050 BSC	
J	0.25	0.32	0.010	0.012
K	0.10	0.25	0.004	0.009
M	0°	7°	0°	7°
P	10.05	10.55	0.395	0.415
R	0.25	0.75	0.010	0.029

CASE 751

Figure 11-2. MC68HC05J1ADW

SECTION 12
ORDERING INFORMATION

This section contains instructions for ordering custom-masked ROM MCUs.

12.1 MCU Ordering Forms

To initiate an order for a ROM-based MCU, first obtain the current ordering form for the MCU from a Motorola representative. Submit the following items when ordering MCUs:

- A current MCU ordering form that is **completely filled out** (Contact your Motorola sales office for assistance.)
- A copy of the customer specification if the customer specification deviates from the Motorola specification for the MCU
- Customer's application program on one of the media listed in **12.2 Application Program Media**

The current MCU ordering form is also available through the Motorola Freeware Bulletin Board Service (BBS). The telephone number is (512) 891-FREE. After making the connection, type bbs in lowercase letters, and press the return key to start the BBS software.

12.2 Application Program Media

Please deliver the application program to Motorola in one of the following media:

- Macintosh®[1] 3-1/2-inch diskette (double-sided double-density 800K or double-sided high-density 1.4M)
- MS-DOS®[2] or PC-DOS®[3] 3-1/2-inch diskette (double-sided double-density 720K or double-sided high-density 1.44M)
- MS-DOS® or PC-DOS® 5-1/4-inch diskette (double-sided double-density 360K or double-sided high-density 1.2M)
- EPROM(s) 2716, 2732, 2764, 27128, 27256, or 27512 (depending on the size of the memory map of the MCU)

Use positive logic for data and addresses.

12.2.1 Diskettes

If submitting the application program on a diskette, clearly label the diskette with the following information:

- Customer name
- Customer part number
- Project or product name
- Filename of object code
- Date
- Name of operating system that formatted diskette
- Formatted capacity of diskette

On diskettes, the application program must be in Motorola's S-record format (S1 and S9 records), a character-based object file format generated by M6805 cross assemblers and linkers.

1. Macintosh is a registered trademark of Apple Computer, Inc.
2. MS-DOS is a registered trademark of Microsoft, Inc.
3. PC-DOS is a registered trademark of International Business Machines Corporation.

NOTE

Begin the application program at the first user ROM location. Program addresses must correspond exactly to the available on-chip user ROM addresses as shown in the memory map. **Write $00 in all non-user ROM locations or leave all non-user ROM locations blank**. See the current MCU ordering form for additional requirements.

If the memory map has two user ROM areas with the same addresses, then write the two areas in separate files on the diskette. Label the diskette with both filenames.

In addition to the object code, a file containing the source code can be included. Motorola keeps this code confidential and uses it only to expedite ROM pattern generation in case of any difficulty with the object code. Label the diskette with the filename of the source code.

12.2.2 EPROMs

If submitting the application program in an EPROM, clearly label the EPROM with the following information:

- Customer name
- Customer part number
- Checksum
- Project or product name
- Date

NOTE

Begin the application program at the first user ROM location. Program addresses must correspond exactly to the available on-chip user ROM addresses as shown in the memory map. **Write $00 in all non-user ROM loctions**. See the current MCU ordering form for additional requirements.

Submit the application program in one EPROM large enough to contain the entire memory map. If the memory map has two user ROM areas with the same addresses, then write the two areas on separate EPROMs. Label the EPROMs with the addresses they contain.

Pack EPROMs securely in a conductive IC carrier for shipment. Do not use Styrofoam.

12.3 ROM Program Verification

The primary use for the on-chip ROM is to hold the customer's application program. The customer develops and debugs the application program and then submits the MCU order along with the application program.

Motorola inputs the customer's application program code into a computer program that generates a listing verify file. The listing verify file represents the memory map of the MCU. The listing verify file contains the user ROM code and may also contain non-user ROM code, such as self-check code. Motorola sends the customer a computer printout of the listing verify file along with a listing verify form.

To aid the customer in checking the listing verify file, Motorola will program the listing verify file into customer-supplied blank EPROMs or preformatted Macintosh or DOS disks. All original pattern media are filed for contractual purposes and are not returned.

Check the listing verify file thoroughly, then complete and sign the listing verify form and return the listing verify form to Motorola. The signed listing verify form constitutes the contractual agreement for the creation of the custom mask.

12.4 ROM Verification Units (RVUs)

After receiving the signed listing verify form, Motorola manufactures a custom photographic mask. The mask contains the customer's application program and is used to process silicon wafers. The application program cannot be changed after the manufacture of the mask begins. Motorola then produces ten MCUs, called RVUs, and sends the RVUs to the customer. RVUs are usually packaged in unmarked ceramic and tested to 5 Vdc at room temperature. RVUs are not tested to environmental extremes because their sole purpose is to demonstrate that the customer's user ROM pattern was properly implemented. The ten RVUs are free of charge with the minimum order quantity but are not production parts. RVUs are not guaranteed by Motorola Quality Assurance.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-800-441-2447 or 303-675-2140

Mfax™: RMFAX0@email.sps.mot.com – TOUCHTONE 602-244-6609, US & Canada ONLY 1-800-774-1848

INTERNET: http://motorola.com/sps

JAPAN: Nippon Motorola Ltd. SPD, Strategic Planning Office 4-32-1, Nishi-Gotanda Shinagawa-ku, Tokyo 141, Japan. 81-3-5487-8488

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

MC68HC05J1A/D