MC68HC705J2/D

HC05

MC68HC705J2

TECHNICAL
DATA

MOTOROLA

MC68HC705J2

HCMOS MICROCONTROLLER UNIT

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

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Section Title Page

Section 5
Resets and Interrupts

TABLE OF CONTENTS (Continued)

Section 6
Memory

Section 7
Timer

Section 8
Bootloader Mode

TABLE OF CONTENTS (Concluded)

Section **Title** **Page**

Section 9
MC68HC05J1 Emulation Mode

Section 10
Electrical Specifications

Section 11
Mechanical Specifications

LIST OF FIGURES

LIST OF FIGURES (Concluded)

LIST OF TABLES

SECTION 1
INTRODUCTION

The MC68HC705J2 is a member of the low-cost, high-performance M68HC05 Family of 8-bit microcontroller units (MCUs). The high-density, complementary metal-oxide semiconductor (HCMOS) 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.

The MC68HC705J2 is an expansion of the MC68HC05J1 design. On-chip memory is enhanced with 2 Kbytes of erasable, programmable ROM (EPROM), 112 Kbytes of RAM, and a bootloader ROM.

1.1 Features

The MCU features include the following:

- Popular M68HC05 CPU
- Memory-Mapped Input/Output (I/O) Registers
- 2064 Bytes of User EPROM Including 16 User Vector Locations
- 112 Bytes of Static RAM (SRAM)
- 14 Bidirectional I/O Pins
- Fully Static Operation With No Minimum Clock Speed
- On-Chip Oscillator With Crystal/Ceramic Resonator Connections
- 15-Bit Multifunction Timer
- Real-Time Interrupt Circuit
- Bootloader ROM
- Power-Saving STOP, WAIT, and Data Retention Modes
- MC68HC05J1 Emulation Mode
- Selectable Edge-Sensitive or Edge- and Level-Sensitive External Interrupt Trigger
- Selectable Computer Operating Properly (COP) Timer
- 8 × 8 Unsigned Multiply Instruction
- One Time Programmable 20-Pin Dual-in-Line Package (DIP)
- One Time Programmable 20-Pin Small Outline Integrated Circuit (SOIC)
- Windowed 20-Pin Cerdip

1.2 Structure

Figure 1-1 shows the organization of the MC68HC705J2 EPROM MCU.

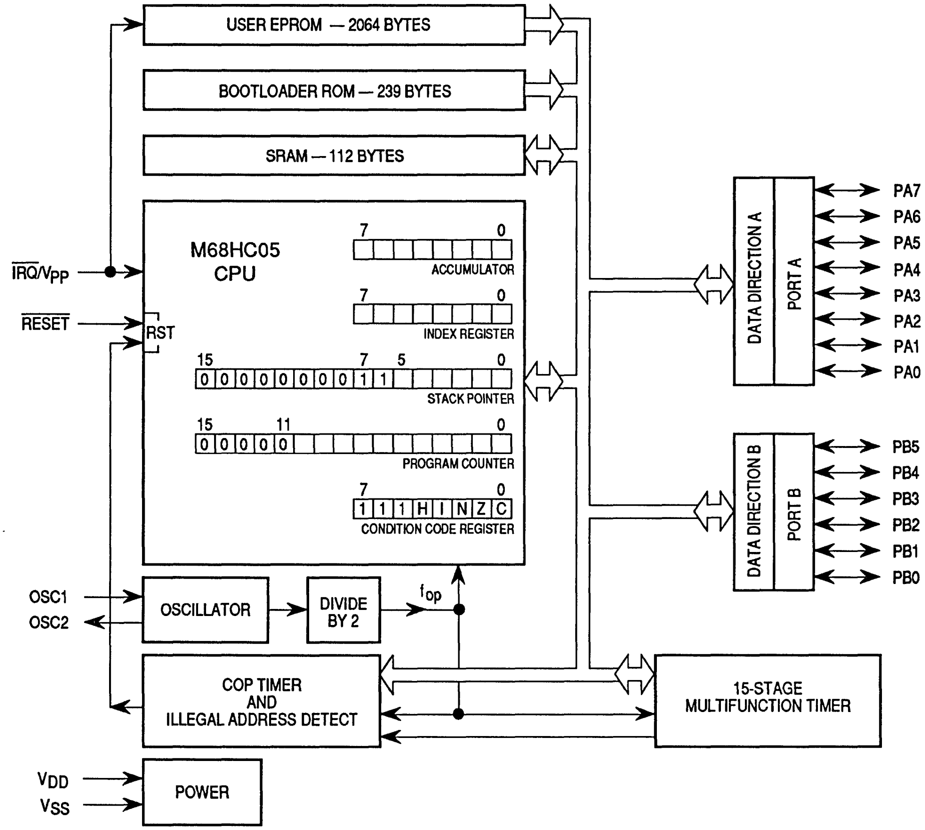
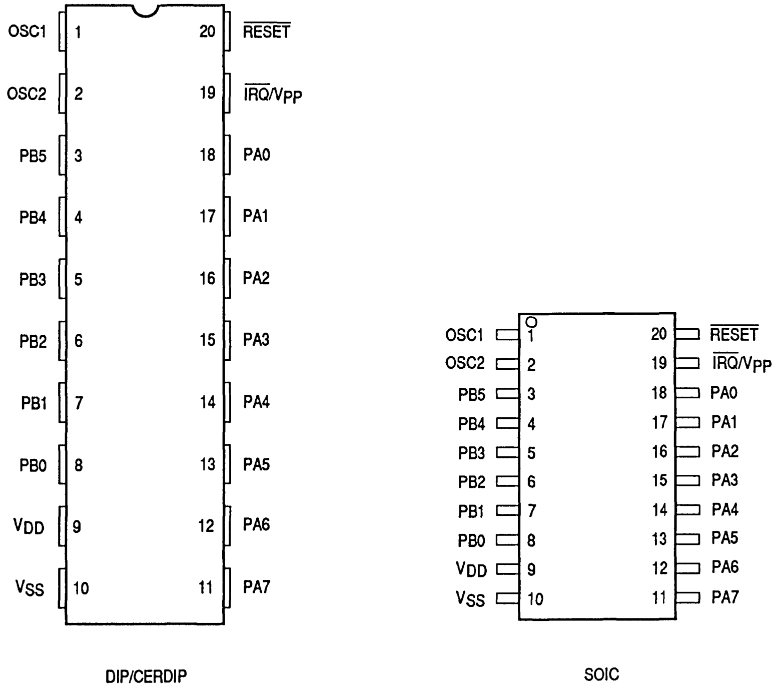


Figure 1-1. MC68HC705J2 Block Diagram

SECTION 2
PIN DESCRIPTIONS

This section describes the function of each pin. Figure 2-1 shows the pin assignments.



DIP/CERDIP

SOIC

Figure 2-1. Pin Assignments

2.1 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. The short rise and fall times place very high short-duration current demands on the power supply. To prevent noise problems, take special care to provide good power supply bypassing at the MCU. Use bypass capacitors with good high-frequency characteristics, and position them as close to the MCU as possible. Bypassing requirements vary, depending on how heavily loaded the MCU pins are.

2.2 OSC1 and OSC2

The OSC1 and OSC2 pins are the control connections for the on-chip oscillator. Connect any of the following to the OSC1 and OSC2 pins:

- A crystal (Refer to Figure 2-2.)
- A ceramic resonator (Refer to Figure 2-2.)
- An external clock signal (Refer to Figure 2-3.)

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

2.2.1 Crystal

The circuit in Figure 2-2 shows a typical crystal oscillator circuit for a parallel resonant crystal. Follow the crystal supplier's recommendations, as the crystal parameters determine the external component values required to provide maximum stability and reliable start-up. The load capacitance values used in the oscillator circuit design should include all stray layout capacitances. Mount the crystal and components as close as possible to the pins for start-up stabilization and to minimize output distortion.

2.2.2 Ceramic Resonator

In cost-sensitive applications, use a ceramic resonator in place of the crystal. Use the circuit in Figure 2-2 for a ceramic resonator, and follow the resonator manufacturer's recommendations, as the resonator parameters determine the external component values required for maximum stability and reliable starting. The load capacitance values used in the oscillator circuit design should include all stray layout capacitances.

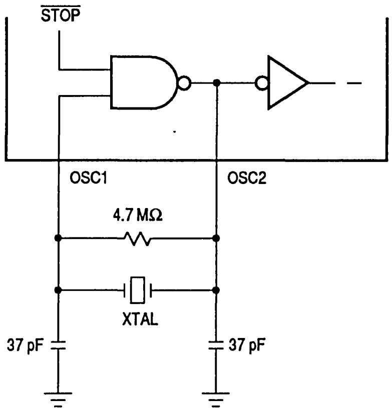


Figure 2-2. Crystal/Ceramic Resonator Connections

2.2.3 External Clock

An external clock from another CMOS-compatible device can drive the OSC1 input, with the OSC2 pin not connected, as Figure 2-3 shows.

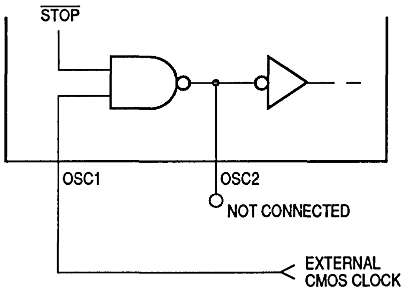


Figure 2-3. External Clock Connections

2.3 $\overline{\text{RESET}}$

A zero on the $\overline{\text{RESET}}$ pin forces the MCU to a known start-up state. See **5.1 Resets** for more information.

2.4 $\overline{\text{IRQ}}/V_{PP}$ (External Interrupt Request/Programming Voltage)

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

- Applying asynchronous external interrupt signals (See **5.2 Interrupts**.)
- Applying the programming voltage for programming the EPROM (See **6.1.3.1 EPROM Programming** and **8.1.1 External EPROM Downloading**.)

SECTION 3
PARALLEL I/O

This section describes the two bidirectional I/O ports.

3.1 I/O Port Function

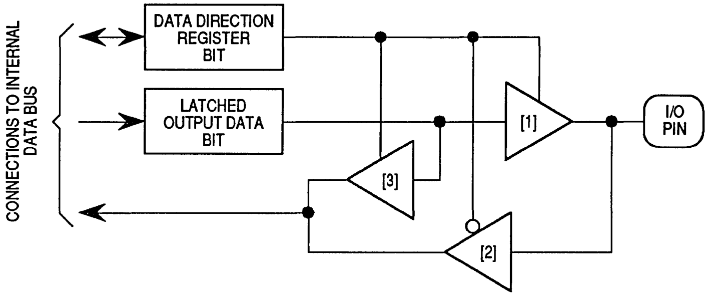
The 14 I/O pins form two I/O ports. Each I/O pin is programmable as an input or an output. The contents of a port data direction register (DDR) determine the data direction for the port. Writing a 1 to a DDR bit enables the output buffer for the associated port pin; a 0 disables the output buffer. A reset initializes all implemented DDR bits to 0, configuring all I/O pins as inputs.

NOTE

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

A reset does not initialize the two port data registers. The port data registers for ports A and B are at addresses $0000 and $0001. To avoid undefined levels, write the data registers before writing the data direction registers.

With an I/O port pin programmed as an output, reading the pin actually reads the value of the output data latch and not the voltage on the pin itself. When a pin is programmed as an input, reading the port bit reads the voltage level on the I/O pin. The output data latch can always be written, regardless of the state of its DDR bit. Refer to Figure 3-1 for typical port circuitry, and to Table 3-1 for a summary of I/O pin functions.



[1] Output buffer enables latched output to drive I/O pin when DDR bit is 1 (output mode).
[2] Input buffer enabled when DDR bit is 0 (input mode).
[3] Input buffer enabled when DDR bit is 1 (output mode).

Figure 3-1. Parallel I/O Port Circuit

Table 3-1. I/O Pin Functions

R/$\overline{\text{W}}$	DDR Bit	I/O Pin Function
0	0	The I/O pin is an input. Data is written into the output data latch.
0	1	Data is written into the output data latch, which drives the I/O pin.
1	0	The state of the I/O pin is read.
1	1	The I/O pin is an output. The output data latch is read.

NOTE: R/$\overline{\text{W}}$ is an internal MCU signal.

3.2 Port A

Port A is an 8-bit general-purpose bidirectional I/O port. The contents of DDRA determine whether each pin is an input or an output. Figures 3-2 and 3-3 show the port A data register and DDRA.

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:	NOT CHANGED BY RESET							

Figure 3-2. Port A Data Register

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 DDRA bit.

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

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

Figure 3-3. Port A Data Direction Register

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

3.3 Port B

Port B is a 6-bit general-purpose bidirectional I/O port. The contents of DDRB determine whether each pin is an input or an output. Figures 3-4 and 3-5 show the port B data register and DDRB.

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:	NOT CHANGED BY RESET							

Figure 3-4. Port B Data Register

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 DDRA bit.

DDRB — Port B Data Direction Register $0005

	Bit 7	6	5	4	3	2	1	Bit 0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
RESET:	0	0	0	0	0	0	0	0

Figure 3-5. Port B Data Direction Register

DDRB7–DDRB0 — Port B Data Direction Bits

These read/write bits control port B data direction.

1 = Corresponding port B pin configured as output

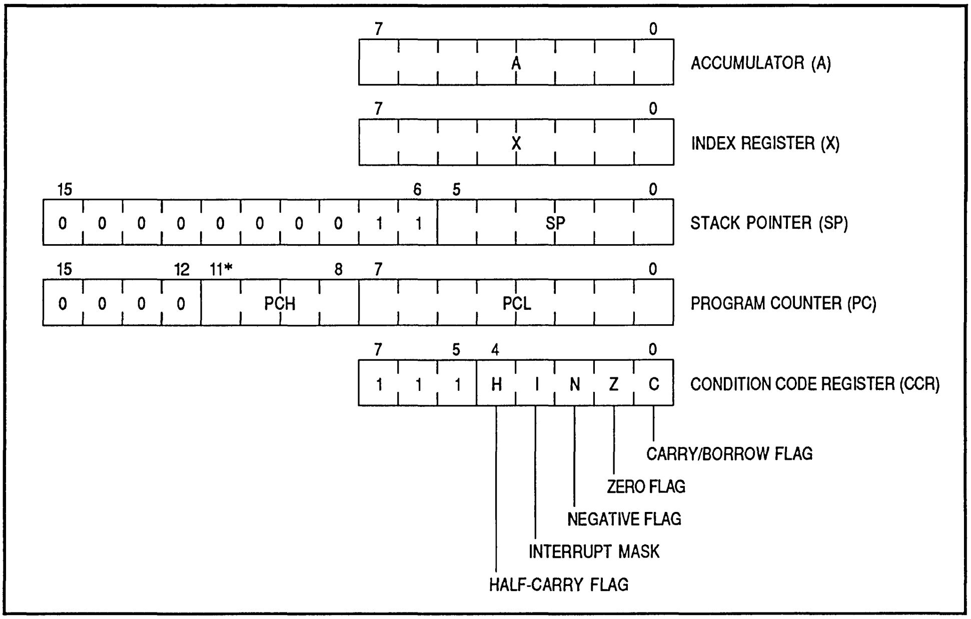
0 = Corresponding port B pin configured as input

SECTION 4
CENTRAL PROCESSOR UNIT

This section describes the registers, instruction set, and addressing modes of the M68HC05 central processor unit (CPU).

4.1 CPU Registers

Figure 4-1 shows the five CPU registers. These are hard-wired registers within the CPU and are not part of the memory map.



*Bit 11 of the program counter is fixed at 0 in MC68HC05J1 emulation mode.

Figure 4-1. Programming Model

4.1.1 Accumulator

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

4.1.2 Index Register

The 8-bit index register can perform two functions:

- Indexed addressing
- Temporary storage

In indexed addressing, the CPU uses the byte in the index register to determine the conditional address of the operand. See **4.3.5 Indexed, No Offset, 4.3.6 Indexed, 8-Bit Offset**, and **4.3.7 Indexed, 16-Bit Offset**.

The index register can also serve as an auxiliary accumulator for temporary storage.

4.1.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next free location on the stack. During a reset or after the reset stack pointer (RSP) instruction, the stack pointer contents are 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.

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 $00C0 and begins writing over the previously stored data. A subroutine uses two stack locations; an interrupt uses five locations.

4.1.4 Program Counter

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched. The four most significant bits of the program counter are permanently fixed at 0000. In MC68HC05J1 emulation mode, the five most significant bits are fixed at 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.

4.1.5 Condition Code Register

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

4.1.5.1 Half-Carry Flag

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

4.1.5.2 Interrupt Mask

Setting the interrupt mask disables interrupts. If an interrupt request occurs while the interrupt mask is 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.

4.1.5.3 Negative Flag

The CPU sets the negative flag when an arithmetic operation, logical operation, or data manipulation produces a negative result. Bit 7 of the negative result is automatically set, so the negative flag can be used to check an often-tested bit by assigning it to bit 7 of a register or memory location. Loading the accumulator with the contents of that register or location then sets or clears the negative flag according to the state of the tested bit.

4.1.5.4 Zero Flag

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

4.1.5.5 Carry/Borrow Flag

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

4.2 Arithmetic/Logic Unit (ALU)

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

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

4.3 Addressing Modes

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

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

4.3.1 Inherent

Inherent instructions are those that have no operand, such as return from interrupt (RTI) and stop (STOP). Other inherent instructions are those that act on data in the CPU registers, such as set carry flag (SEC) and increment accumulator (INCA). Inherent instructions require no memory address and are one byte long. Table 4-1 lists the instructions that use the inherent addressing mode.

Table 4-1. Inherent Addressing Instructions

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

4.3.2 Immediate

Immediate instructions are those that contain a value to be used in an operation with the value in the accumulator or index register. Immediate instructions require no memory address and are two bytes long. The opcode is the first byte and the immediate data value is the second byte. Table 4-2 lists the instructions that use the immediate addressing mode.

Table 4-2. Immediate Addressing Instructions

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

4.3.3 Direct

Direct instructions can access any of the first 256 memory addresses with only two bytes. The first byte is the opcode and the second byte is the low byte of the operand's address. In the direct addressing mode, the CPU automatically uses $00 as the high byte of the operand's address. BRSET and BRCLR are three-byte instructions that use direct addressing to access the operand and relative addressing to specify a branch destination. Table 4-3 lists the instructions that use the direct addressing mode.

Table 4-3. Direct Addressing Instructions

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

4.3.4 Extended

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

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

Table 4-4. Extended Addressing Instructions

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

4.3.5 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 operand's conditional address. The CPU automatically uses $00 as the high byte of the operand's conditional address, so these instructions can address locations $0000–$00FF.

Indexed, no offset instructions are often used to move a pointer through a table or to hold the address of a frequently used RAM or I/O location. Table 4-5 lists the instructions that use the indexed, no offset addressing mode.

4.3.6 Indexed, 8-Bit Offset

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

Indexed, 8-bit offset instructions are useful for selecting the kth element in an n-element table. The table can begin anywhere within the first 256 memory locations and could extend as far as location 510 ($01FE). The k value would typically be in the index register, and the address of the beginning of the table would be in the byte following the opcode. Table 4-5 lists the instructions that use the indexed, 8-bit offset addressing mode.

4.3.7 Indexed, 16-Bit Offset

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

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

As with direct and extended addressing, the Motorola assembler determines the shortest form of indexed addressing. Table 4-5 lists the instructions that can use the indexed, 16-bit offset addressing mode.

Table 4-5. Indexed Addressing Instructions

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

4.3.8 Relative

The relative addressing mode is only for branch instructions and bit test and branch instructions. 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 true. If the branch condition is not true, the CPU goes to the next instruction. To permit branching either forward or backward, the offset is a signed, two's complement byte that gives a branching range of –127 to +128 bytes from the address of the next location after the branch instruction.

When using the Motorola assembler, the programmer does not need to calculate the offset, because the assembler determines the proper offset and verifies that it is within the span of the branch. Table 4-6 lists the instructions that use the relative addressing mode.

Table 4-6. Relative Addressing Instructions

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

4.4 Instruction Set

The MCU uses all the instructions available in the M146805 CMOS Family plus the unsigned multiply (MUL) instruction. The MUL instruction allows unsigned multiplication of the contents of the accumulator and the index register. The CPU stores the high-order product in the index register, and the low-order product in the accumulator.

The MCU instructions fall into the following five categories:

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

4.4.1 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 using one of the addressing modes. Most register/memory instructions use the following addressing modes:

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

Table 4-7 lists the register/memory instructions.

Table 4-7. Register/Memory Instructions

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

4.4.2 Read-Modify-Write Instructions

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

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

Table 4-8 lists the read-modify-write instructions.

Table 4-8. Read-Modify-Write Instructions

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

4.4.3 Jump/Branch Instructions

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

- Direct
- Extended
- Indexed, no offset
- Indexed, 8-bit offset
- Indexed, 16-bit offset

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

Bit test and branch instructions cause a branch based on the condition 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 4-9 lists the jump and branch instructions.

Table 4-9. Jump and Branch Instructions

Instruction	Mnemonic
Branch Always	BRA
Branch Never	BRN
Branch if Bit n of M = 0	BRCLR
Branch if Bit n of M = 1	BRSET
Branch if Higher	BHI
Branch if Lower or Same	BLS
Branch if Carry Clear	BCC
Branch if Higher or Same	BHS
Branch if Carry Set	BCS
Branch if Lower	BLO
Branch if Not Equal	BND
Branch if Equal	BEQ
Branch if Half-Carry Clear	BHCC
Branch if Half-Carry Set	BHCS
Branch if Plus	BPL
Branch if Minus	BMI
Branch if Interrupt Mask Clear	BMC
Branch if Interrupt Mask Set	BMS
Branch if Interrupt Line Low	BIL
Branch if Interrupt Line High	BIH
Branch to Subroutine	BSR
Jump Unconditional	JMP
Jump to Subroutine	JSR

4.4.4 Bit Manipulation Instructions

The CPU can set or clear any writable bit in the first 256 bytes of memory. Port register, 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 the direct addressing mode. Table 4-10 lists these instructions.

Table 4-10. Bit Manipulation Instructions

Instruction	Mnemonic
Set Bit n	BSET n (n = 0 . . . 7)
Clear Bit n	BCLR n (n = 0 . . . 7)
Branch if Bit n of M = 0	BRCLR
Branch if Bit n of M = 1	BRSET

4.4.5 Control Instructions

These register reference instructions control CPU operation during program execution. Control instructions, listed in Table 4-11, use the inherent addressing mode.

Table 4-11. Control Instructions

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

4.4.6 Instruction Set Summary

Table 4-12 shows all MC68HC705J2 instructions in all possible addressing modes. For each instruction, the operand construction and the execution time in internal clock cycles (t_{cyc}) are shown. One internal clock cycle equals two oscillator input cycles. The following legend summarizes the symbols and abbreviations used in Table 4-12.

Abbreviations and Symbols

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

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

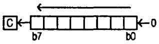
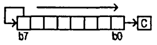
Source Form(s)	Operation	Description	Addressing Mode for Operand	Machine Coding (hexadecimal)		Cycles	Condition Code				
				Opcode	Operand		H	I	N	Z	C
ADC opr	Add with carry	A ← (A) + (M) + C	IMM DIR EXT IX2 IX1 IX	A9 B9 C9 D9 E9 F9	ii dd hh ll ee ff ff	2 3 4 5 4 3	↕	–	↕	↕	↕
ADD opr	Add without carry	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	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 ASL opr	Arithmetic shift left		DIR INH INH IX1 IX	38 48 58 68 78	dd ff	5 3 3 6 5	–	–	↕	↕	↕
ASR opr ASRA ASRX ASR opr ASR opr	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	?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	?C=1	REL	25	rr	3	–	–	–	–	–
BEQ rel	Branch if equal	?Z=1	REL	27	rr	3	–	–	–	–	–
BHCC rel	Branch if half carry bit clear	?H=0	REL	28	rr	3	–	–	–	–	–
BHCS rel	Branch if half carry bit set	?H=1	REL	29	rr	3	–	–	–	–	–
BHI rel	Branch if higher	?C+Z=0	REL	22	rr	3	–	–	–	–	–
BHS rel	Branch if higher or same	?C=0	REL	24	rr	3	–	–	–	–	–
BIH rel	Branch if $\overline{\text{IRQ}}$ pin high	?$\overline{\text{IRQ}}$=1	REL	2F	rr	3	–	–	–	–	–
BIL rel	Branch if $\overline{\text{IRQ}}$ pin low	?$\overline{\text{IRQ}}$=0	REL	2E	rr	3	–	–	–	–	–
BIT rel	Bit test accumulator contents with memory contents	(A) • (M)	IMM DIR EXT IX2 IX1 IX	A5 B5 C5 D5 E5 F5	ii dd hh ll ee ff ff	2 3 4 5 4 3	–	–	↕	↕	–
BLO rel	Branch if lower	?C=1	REL	25	rr	3	–	–	–	–	–
BLS rel	Branch if lower or same	?C+Z=1	REL	23	rr	3	–	–	–	–	–
BMC rel	Branch if interrupt mask clear	?I=0	REL	2C	rr	3	–	–	–	–	–
BMI rel	Branch if minus	?N=1	REL	2B	rr	3	–	–	–	–	–
BMS rel	Branch if interrupt mask set	?I=0	REL	2D	rr	3	–	–	–	–	–
BNE rel	Branch if not equal	?Z=0	REL	26	rr	3	–	–	–	–	–
BPL rel	Branch if plus	?N=0	REL	2A	rr	3	–	–	–	–	–

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

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

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

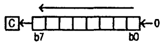
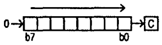
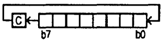
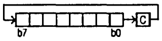
Source Form(s)	Operation	Description	Addressing Mode for Operand	Machine Coding (hexadecimal)		Cycles	Condition Code				
				Opcode	Operand		H	I	N	Z	C
EOR opr	Exclusive OR accumulator contents with memory contents	A ← (A) ⊕ (M)	IMM	A8	ii	2	–	–	↕	↕	–
			DIR	B8	dd	3					
			EXT	C8	hh ll	4					
			IX2	D8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

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

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

4.4.7 Opcode Map

Table 4-13 is an opcode map of the M68HC05 instruction set.

Table 4-13. Opcode Map

	Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory						
	DIR	DIR	REL	DIR	INH	INH	IX1	IX	INH	INH	IMM	DIR	EXT	IX2	IX1	IX	
HI LO	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	HI LO
0 0000	5 BRSET0 3 DIR	5 BSET0 2 DIR	3 BRA 2 REL	5 NEG 2 DIR	3 NEGA 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 0000
1 0001	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 0001
2 0010	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 0010
3 0011	5 BRCLR1 3 DIR	5 BSCLR1 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 0011
4 0100	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 0100
5 0101	5 BRCLR2 3 DIR	5 BCLR2 2 DIR	3 BCS 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 0101
6 0110	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 0110
7 0111	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 0111
8 1000	5 BRSET4 3 DIR	5 BSET4 2 DIR	3 BHCC 2 REL	5 LSL 2 DIR	3 LSLA 1 INH	3 LSLX 1 INH	6 LSL 2 IX1	5 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 1000
9 1001	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	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 1001
A 1010	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 INH	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 1010
B 1011	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 1011
C 1100	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 INH	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 1100
D 1101	5 BRCLR6 3 DIR	5 BCLR6 2 DIR	3 BMS 2 REL	4 TST 2 DIR	3 TSTA 1 INH	3 TSTX 1 INH	6 TST 2 INH	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 1101
E 1110	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 1110
F 1111	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 INH	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 1111

ABBREVIATIONS FOR ADDRESSING MODES

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

LEGEND

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

4.5 Low-Power Modes

The following paragraphs describe the STOP and WAIT modes. (Refer also to **6.2 Data Retention Mode.**)

4.5.1 STOP Mode

The STOP instruction puts the MCU in its lowest power-consumption mode. In STOP mode, the following events occur:

- The CPU clears TOF and RTIF, the timer interrupt flags in the timer control and status register, removing any pending timer interrupts.
- The CPU clears TOIE and RTIE, the timer interrupt enable bits in the timer control and status register, disabling further timer interrupts.
- The CPU clears the divide-by-four timer prescaler.
- The CPU clears the interrupt mask in the condition code register, enabling external interrupts.
- The internal oscillator stops, halting all internal processing, including operation of the timer and the COP timer.

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. An external interrupt automatically loads the program counter with the contents of locations $0FFA and $0FFB, the locations of the vector address of the external interrupt service routine.
- A reset signal on the $\overline{\text{RESET}}$ pin. A reset automatically loads the program counter with the contents of locations $0FFE and $0FFF, the locations of the vector address of the reset service routine.

Refer to Figure 10-7 in **SECTION 10 ELECTRICAL SPECIFICATIONS** for STOP recovery timing.

Figure 4-2 shows the sequence of events caused by the STOP instruction.

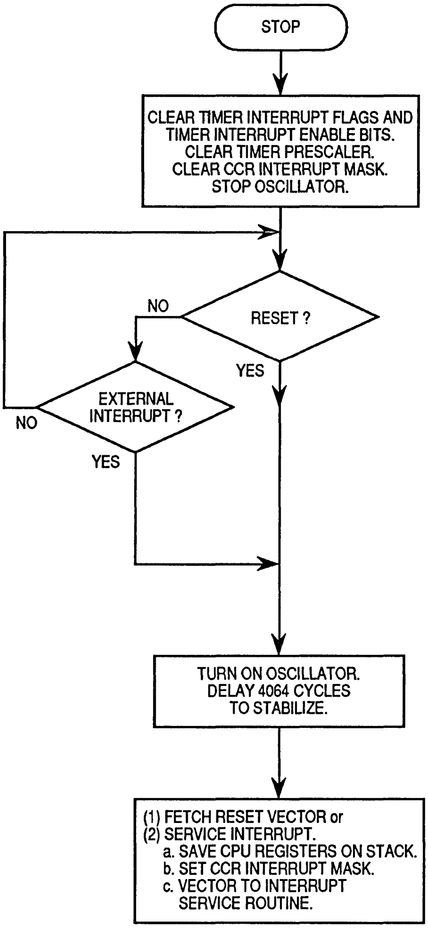

STOP
CLEAR TIMER INTERRUPT FLAGS AND TIMER INTERRUPT ENABLE BITS.
CLEAR TIMER PRESCALER.
CLEAR CCR INTERRUPT MASK.
STOP OSCILLATOR.
RESET ?
NO
YES
EXTERNAL INTERRUPT ?
NO
YES
TURN ON OSCILLATOR.
DELAY 4064 CYCLES TO STABILIZE.
(1) FETCH RESET VECTOR or
(2) SERVICE INTERRUPT.
a. SAVE CPU REGISTERS ON STACK.
b. SET CCR INTERRUPT MASK.
c. VECTOR TO INTERRUPT SERVICE ROUTINE.

Figure 4-2. STOP Instruction Flowchart

4.5.2 WAIT Mode

The WAIT instruction puts the MCU in an intermediate power-consumption mode. In WAIT mode, the following events occur:

- All CPU clocks stop.
- The CPU clears the interrupt mask in the condition code register, enabling external interrupts and timer interrupts.

The WAIT instruction does not affect any other registers or any I/O lines. The timer and COP timer remain active in WAIT mode.

The following conditions bring the MCU out of WAIT mode:

- A timer interrupt. If a real-time interrupt or a timer overflow interrupt occurs during WAIT mode, the MCU loads the program counter with the contents of locations $0FF8 and $0FF9, the locations of the vector address of the timer interrupt service routine.
- An external interrupt. An external interrupt automatically loads the program counter with the contents of locations $0FFA and $0FFB, the locations of the vector address of the external interrupt service routine.
- A COP timer reset. A timeout of the COP timer during WAIT mode resets the MCU. The programmer can enable real-time interrupts so the MCU can periodically exit WAIT mode to reset the COP timer.
- A reset signal on the $\overline{\text{RESET}}$ pin during WAIT mode resets the MCU.

A COP timer reset or a reset signal on the $\overline{\text{RESET}}$ pin automatically loads the program counter with the contents of locations $0FFE and $0FFF, the locations of the vector address of the reset service routine.

Figure 4-3 shows the sequence of events caused by the WAIT instruction.

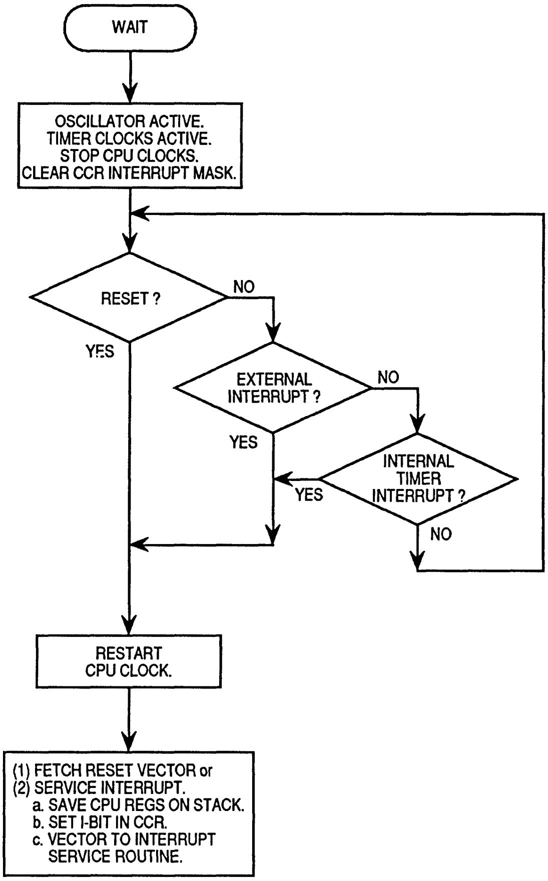


Figure 4-3. WAIT Instruction Flowchart

SECTION 5
RESETS AND INTERRUPTS

This section describes how resets reinitialize the MCU and how interrupts temporarily change the normal processing sequence.

5.1 Resets

A reset immediately stops the operation of the instruction being executed. A reset initializes certain control bits to known conditions 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 logical zero applied to the $\overline{\text{RESET}}$ pin (external reset)
- Timeout of the COP timer (COP reset)
- An opcode fetch from an address not in the memory map (illegal address reset)

A reset does the following things to reinitialize the MCU:

- Clears all implemented data direction register bits so that the corresponding I/O pins are inputs
- Loads the stack pointer with $FF
- Sets the interrupt mask, inhibiting interrupts
- Clears the TOFE and RTIE bits in the timer control and status register
- Clears the STOP latch, enabling the CPU clocks
- Clears the WAIT latch, waking the CPU from the WAIT mode
- Loads the program counter with the user-defined reset vector

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 a logical 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 a logical one.

5.1.2 External Reset

A 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.

5.1.3 Computer Operating Properly (COP) Reset

A timeout of the COP timer generates a COP reset. The COP timer is part of a software error detection system and must be cleared periodically to start a new timeout period. (See **7.3 COP Timer**.) To clear the COP timer and prevent a COP reset, write a zero to bit 0 (COPR) of the COP control register at location $0FF0 before the COP timer times out. The COP control register is a write-only register that returns the contents of an EPROM location when read. See Figure 5-1.

COPR — COP Control Register **$0FF0**

	Bit 7	6	5	4	3	2	1	Bit 0
	—	—	—	—	—	—	—	COPR
RESET	—	—	—	—	—	—	—	0

Figure 5-1. COP Control Register

COPR — COP Reset

COPR is a write-only bit. Periodically writing a zero to COPR prevents the COP timer from resetting the MCU.

5.1.4 Illegal Address Reset

An opcode fetch from an address that is not in the EPROM (locations $0700–$0EFF), or the RAM ($0090–$00FF) generates an illegal address reset.

5.2 Interrupts

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

- Timer overflow or real-time interrupt request (timer interrupts)
- A logical zero applied to the $\overline{\text{IRQ}}$ pin (external interrupt)
- SWI instruction (software interrupt)

The CPU does the following things to begin servicing an interrupt:

- Stores the contents of the CPU registers on the stack as shown in Figure 5-2

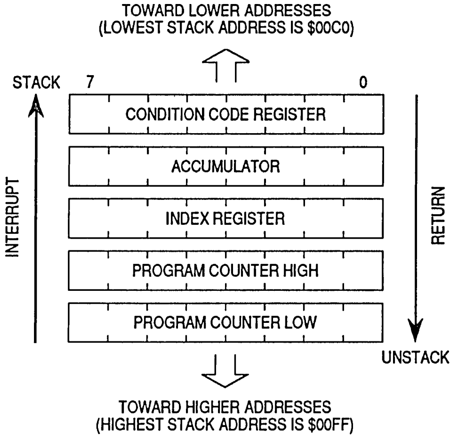


Figure 5-2. Interrupt Stacking Order

- Sets the interrupt mask to prevent further interrupts
- Loads the program counter with the contents of the appropriate interrupt vector locations:

— $0FF8 and $0FF9 (timer interrupt vector)
— $0FFA and $0FFB (external interrupt vector)
— $0FFC and $0FFD (software interrupt vector)

The return from interrupt (RTI) instruction causes the CPU to recover the CPU registers from the stack as shown in Figure 5-2.

5.2.1 Timer Interrupts

The timer generates two kinds of interrupts:

- Timer overflow interrupt
- Real-time interrupt

Setting the interrupt mask in the condition code register disables timer interrupts.

5.2.1.1 Timer Overflow Interrupts

A timer overflow interrupt occurs if the timer overflow flag, TOF, becomes set while the timer overflow interrupt enable bit, TOIE, is also set. TOF and TOIE are in the timer control and status register. See **7.2 Timer Control and Status Register**.

5.2.1.2 Real-Time Interrupts

A real-time interrupt occurs if the real-time interrupt flag, RTIF, becomes set while the real-time interrupt enable bit, RTIE, is also set. RTIF and RTIE are in the timer control and status register. See **7.2 Timer Control and Status Register**.

5.2.2 External Interrupt

When a falling edge occurs on the $\overline{\text{IRQ}}$ pin, an external interrupt request is latched. When the CPU completes its current instruction, it tests the external interrupt latch. If the interrupt latch is set and the interrupt mask in the condition code register is reset, the CPU then begins the interrupt sequence. The CPU clears the interrupt 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 interrupt mask is cleared (usually during the return from interrupt), the CPU can recognize the new interrupt request.

Figure 5-3 shows the sequence of events caused by an interrupt.

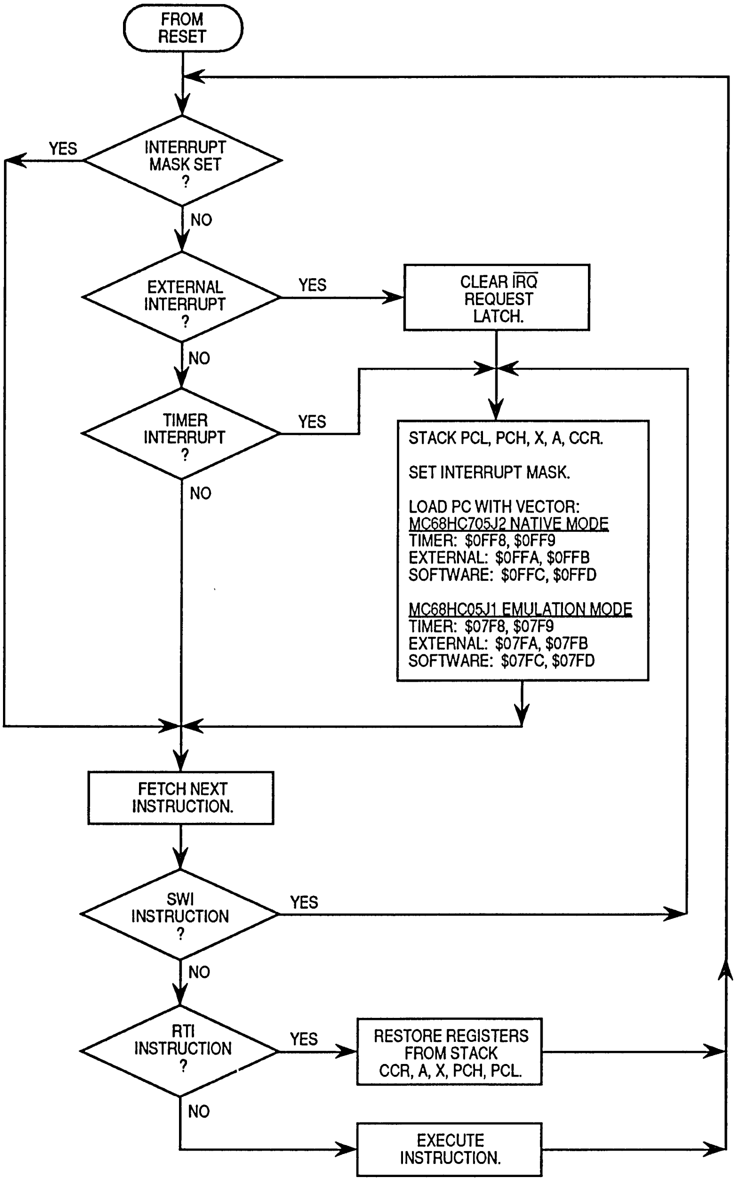


Figure 5-3. Interrupt Flowchart

Either an edge-sensitive or an edge- and level-sensitive external interrupt trigger is programmable in the mask option register. Figure 5-4 shows the internal logic of this programmable option.

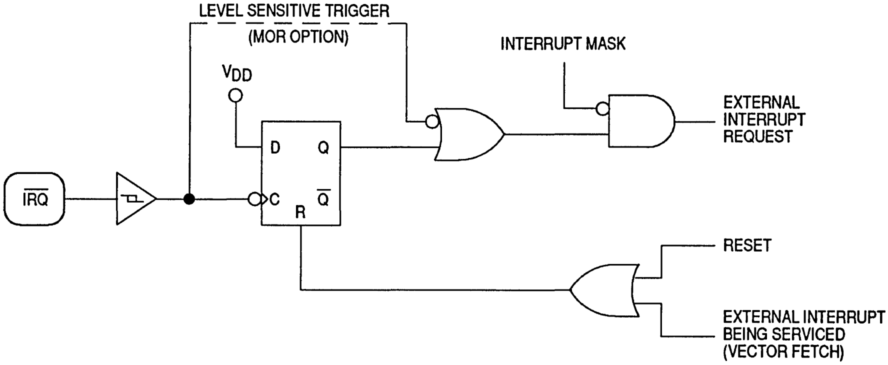


Figure 5-4. External Interrupt Trigger Option

The edge- and level-sensitive trigger option allows multiple external interrupt sources to be wire-ORed to the $\overline{IRQ}$ pin. With the level-sensitive trigger option, an external interrupt request is latched as long as any source is holding the $\overline{IRQ}$ pin low.

Setting the interrupt mask in the condition code register disables external interrupts.

5.2.3 Software Interrupt

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

SECTION 6
MEMORY

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

6.1 Memory Map

The CPU can address 4 Kbytes of memory space. The program counter normally advances one address at a time through the memory, reading the program instructions and data. The EPROM portion of memory holds the program instructions, fixed data, user-defined vectors, and service routines. The RAM portion of memory holds variable data. I/O registers are memory-mapped so that the CPU can access their locations in the same way that it accesses all other memory locations.

Figure 6-1 is a memory map of the MCU. Figure 6-2 is a more detailed memory map of the 32-byte I/O register section.

6.1.1 Input/Output Section

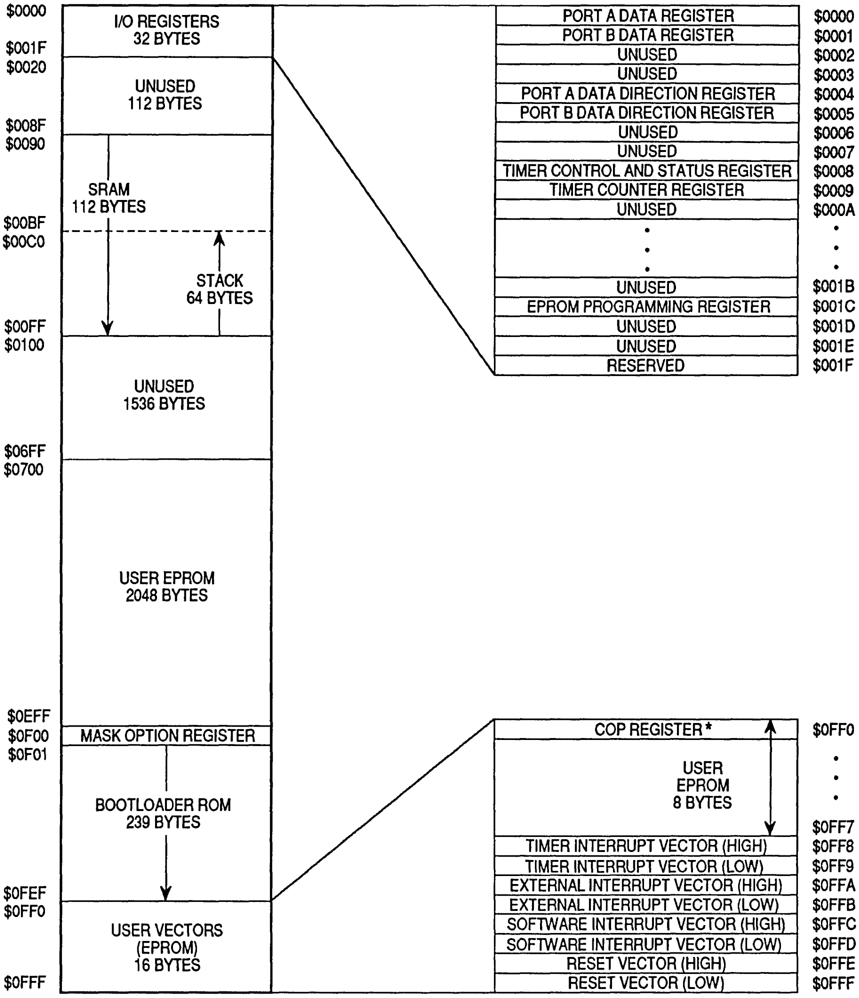
The first 32 addresses of the memory space, $0000–$001F, are defined as the I/O section. These are the addresses of the I/O control registers, I/O status registers, and I/O data registers.

6.1.2 RAM

The MCU has 112 bytes of fully static read/write memory for storage of variable and temporary data during program execution. RAM addresses $00C0–$00FF serve as the stack. The CPU uses the stack to save CPU register contents before processing an interrupt or subroutine call. The stack pointer decrements during pushes and increments during pulls.

NOTE

Be careful if using the stack addresses ($00C0–$00FF) for data storage or as a temporary work area. The CPU may overwrite data in the stack during a subroutine or interrupt.



PORT A DATA REGISTER	$0000
PORT B DATA REGISTER	$0001
UNUSED	$0002
UNUSED	$0003
PORT A DATA DIRECTION REGISTER	$0004
PORT B DATA DIRECTION REGISTER	$0005
UNUSED	$0006
UNUSED	$0007
TIMER CONTROL AND STATUS REGISTER	$0008
TIMER COUNTER REGISTER	$0009
UNUSED	$000A
• • •	• • •
UNUSED	$001B
EPROM PROGRAMMING REGISTER	$001C
UNUSED	$001D
UNUSED	$001E
RESERVED	$001F

COP REGISTER *	$0FF0
USER EPROM 8 BYTES	• • • $0FF7
TIMER INTERRUPT VECTOR (HIGH)	$0FF8
TIMER INTERRUPT VECTOR (LOW)	$0FF9
EXTERNAL INTERRUPT VECTOR (HIGH)	$0FFA
EXTERNAL INTERRUPT VECTOR (LOW)	$0FFB
SOFTWARE INTERRUPT VECTOR (HIGH)	$0FFC
SOFTWARE INTERRUPT VECTOR (LOW)	$0FFD
RESET VECTOR (HIGH)	$0FFE
RESET VECTOR (LOW)	$0FFF

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

Figure 6-1. 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	0	0	RT1	RT0	TCSR
$0009	Bit 7	6	5	4	3	2	1	Bit 0	TCR
$000A	—	—	—	—	—	—	—	—	UNUSED
$000B	—	—	—	—	—	—	—	—	UNUSED
$000C	—	—	—	—	—	—	—	—	UNUSED
•									•
•									•
•									•
$0019	—	—	—	—	—	—	—	—	UNUSED
$001A	—	—	—	—	—	—	—	—	UNUSED
$001B	—	—	—	—	—	—	—	—	UNUSED
$001C	0	0	0	0	0	LATCH	0	EPGM	PROG
$001D	—	—	—	—	—	—	—	—	UNUSED
$001E	—	—	—	—	—	—	—	—	UNUSED
$001F	—	—	—	—	—	—	—	—	RESERVED
$0F00	—	—	—	—	—	J1	IRQ	COP	MOR
$0FF0								COPR	COP

Figure 6-2. I/O Registers

6.1.3 EPROM

Two Kbytes of user EPROM for storage of program instructions and fixed data are located at addresses \$0700–\$0EFF. The eight addresses from \$0FF8–\$0FFF are EPROM locations reserved for interrupt vectors and reset vectors. Eight additional EPROM bytes are located at \$0FF0–\$0FF8. There are two ways to write data to the EPROM:

- The EPROM programming register contains the control bits for programming the EPROM on a byte-by-byte basis.
- The bootloader ROM contains routines to download the contents of an external memory device to the on-chip EPROM.

6.1.3.1 EPROM Programming

The EPROM programming register, shown in Figure 6-3, contains the control bits for programming the EPROM.

PROG — EPROM Programming Register **\$001C**

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

Figure 6-3. EPROM Programming Register (PROG)

LATCH — EPROM Bus Latch

This read/write bit causes address and data buses to be latched for EPROM programming. Clearing the LATCH bit automatically clears the EPGM bit.

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

EPGM — EPROM Programming

This read/write bit applies programming power to the EPROM. To write the EPGM bit, the LATCH bit must already be set.

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

Bits 7–3 and 1 — Not used; always read as zeros.

Take the following steps to program a byte of EPROM:

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

6.1.3.2 EPROM Erasing

The erased state of an EPROM bit is zero. Erase the EPROM by exposing it to 15 Ws/cm^2 of ultraviolet light with a wavelength of 2537 angstroms. Position the ultraviolet light source 1 inch from the EPROM. Do not use a shortwave filter.

NOTE

Windowed packages must have the window covered during programming and operation.

6.1.4 Bootloader ROM

Addresses $0F01–$0FEF contain the bootloader ROM, which can copy and verify the contents of an external EPROM to the on-chip EPROM. See **SECTION 8 BOOTLOADER MODE**.

6.2 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 zero.
2. Lower the V_{DD} voltage. The $\overline{\text{RESET}}$ line 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 logical one.

SECTION 7
TIMER

This section describes the operation of the timer and the COP timer. Figure 7-1 shows the organization of the timer system.

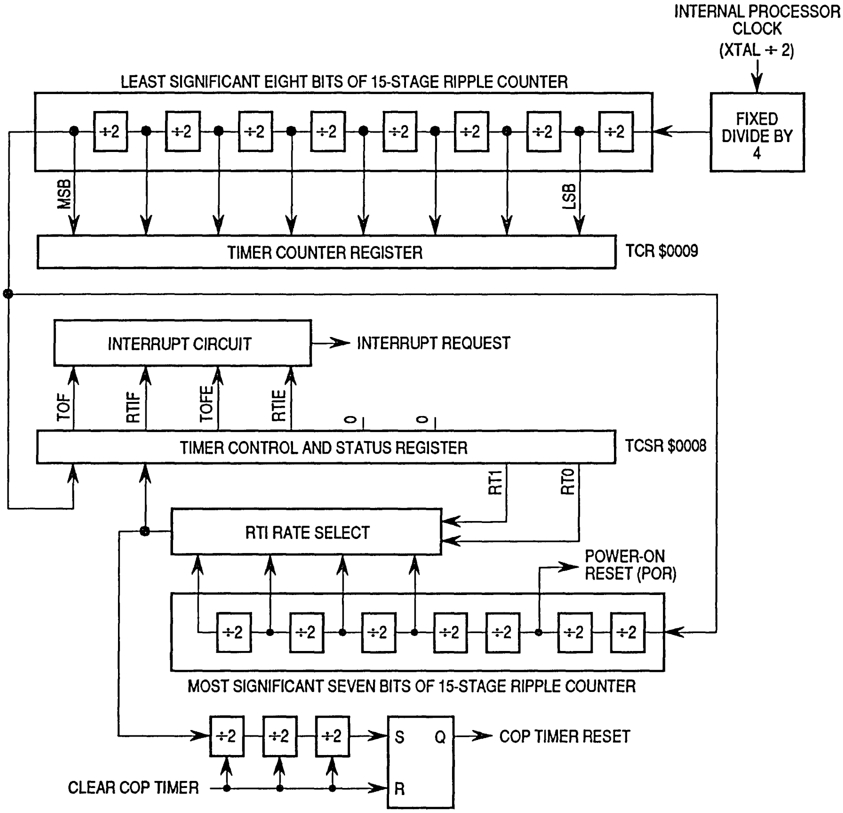


Figure 7-1. Timer

7.1 Timer Counter Register (TCR)

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 shown in Figure 7-2.

TCR — Timer Counter Register **$0009**

	Bit 7	6	5	4	3	2	1	Bit 0
RESET	0	0	0	0	0	0	0	0

Figure 7-2. Timer Counter Register (TCR)

Power-on clears the entire counter chain and begins clocking the counter. After 4064 cycles of the internal clock, 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 makes timer interrupts possible every 1024 internal clock cycles.

7.2 Timer Control and Status Register (TCSR)

Timer interrupt flags, timer interrupt enable bits, and real-time interrupt rate select bits are in the read/write timer control and status register.

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

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

Figure 7-3. Timer Control and Status Register (TCSR)

TOF — Timer Overflow Flag

This clearable, read-only bit becomes set when the first eight stages of the counter roll over from $FF to $00. TOF generates a timer overflow interrupt request if TOFE is also set. Clear TOF by writing a zero to it. Writing a one to TOF has no effect.

RTIF — Real-Time Interrupt Flag

This clearable, read-only bit 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 zero to it. Writing a one to RTIF has no effect.

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

Bits 3 and 2 — Not used. Always read as zeros.

RT1, RT0 — Real-Time 1 and 0

These read/write bits select one of four real-time interrupt rates. See Table 7-1.

The real-time interrupt rate should be selected by reset initialization software. A reset sets both RT1 and RT0, selecting the lowest real-time interrupt rate. Changing the real-time interrupt rate near the end of the RTI period or during a cycle in which the counter is switching can produce unpredictable results.

Because the selected RTI output drives the COP timer, changing the real-time interrupt rate also changes the counting rate of the COP timer.

Table 7-1. 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	7 × RTI Period	57.3 ms
0 1	$f_{op} \div 2^{15}$	16.4 ms	7 × RTI Period	114.7 ms
1 0	$f_{op} \div 2^{16}$	32.8 ms	7 × RTI Period	229.4 ms
1 1	$f_{op} \div 2^{17}$	65.5 ms	7 × RTI Period	458.8 ms

7.3 COP Timer

Three counter stages at the end of the timer make up the computer operating properly (COP) timer. (See Figure 7-1.) The COP timer is a software error detection system that automatically times out and resets the MCU if not cleared periodically by a program sequence. Writing a zero to bit 0 of the COP register clears the COP timer and prevents a COP timer reset. (See Figure 7-4.)

COPR — COP Register **$0FF0**

MC68HC05J1 Emulation Mode: **$07F0**

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

Figure 7-4. COP Register (COPR)

COPC — COP Clear

This write-only bit resets the COP timer. Reading address $0FF0 returns the EPROM data at that address.

SECTION 8
BOOTLOADER MODE

This section describes how to use the bootloader ROM to download to the on-chip EPROM.

8.1 Bootloader ROM

The bootloader ROM, located at addresses $0F01–$0FEF, contains routines for copying to the on-chip EPROM from an external EPROM or from a personal computer.

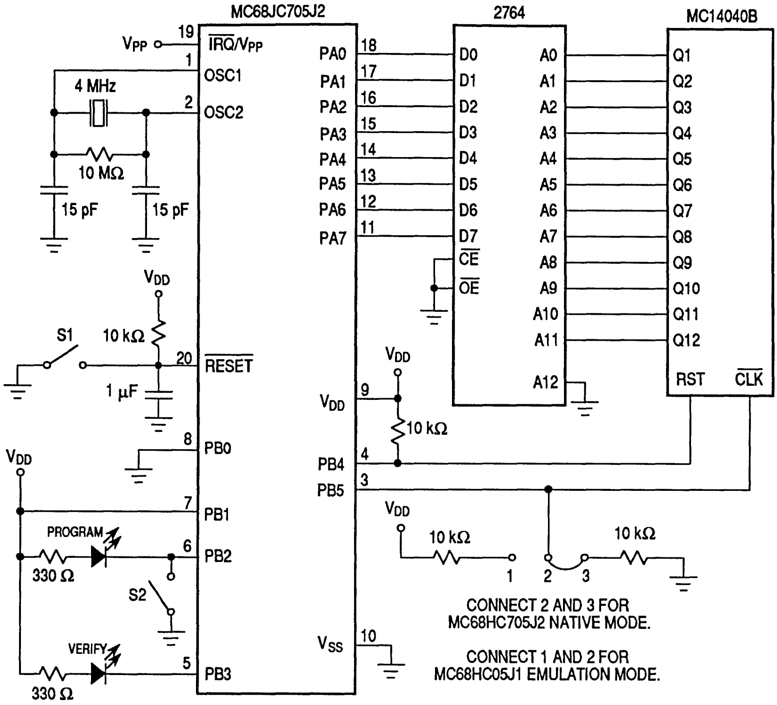
In MC68HC705J2 native mode, the bootloader copies to the 2 Kbyte space located at EPROM addresses $0700–$0EFF. In MC68HC05J1 emulation mode, the bootloader copies to the 1 Kbyte space located at EPROM addresses $0300–$06FF. The addresses of the copied code must correspond to the internal addresses to which the code is copied. The bootloader ignores all other addresses.

The COP timer is automatically disabled in bootloader mode.

8.1.1 External EPROM Downloading

Figure 8-1 shows the circuit used to download to the on-chip EPROM from a 2764 EPROM. The bootloader circuit includes an external 12-bit counter to address the EPROM containing the code to be copied.

Operation is fastest when unused external EPROM addresses contain $00.



CONNECT 2 AND 3 FOR MC68HC705J2 NATIVE MODE.

CONNECT 1 AND 2 FOR MC68HC05J1 EMULATION MODE.

Figure 8-1. Bootloader Circuit

The bootloader function begins when a rising edge occurs on the $\overline{\text{RESET}}$ pin while the $\overline{\text{IRQ}}/V_{PP}$ pin is at V_{PP}, the PB1 pin is at logical one, and the PB0 pin is grounded.

The PB2 pin selects the bootloader function, as the following table shows.

Table 8-1. Bootloader Function Selection

PB2	Bootloader Function
1	Program and Verify
0	Verify

Complete the following steps to bootload the MCU:

1. Turn off all power to the circuit.
2. Install the MCU and the EPROM.
3. Select the MCU mode:
   a. Install a jumper between points 2 and 3 to program the MCU as an MC68HC705J2.
   b. Install a jumper between points 1 and 2 to program the MCU as an MC68HC05J1.
4. Select the bootloader function:
   a. Open switch S2 to select the program and verify function.
   b. Close switch S2 to select the verify only function.
5. Close switch S1 to reset the MCU.
6. Apply V_{DD} to the circuit.
7. Apply the EPROM programming voltage, V_{PP}, to the circuit.
8. Open switch S1 to take the MCU out of reset. During programming the PROGRAM LED turns on. It turns off when the verification routine begins. If verification is successful, the VERIFY LED turns on. If the bootloader finds an error during verification, it puts the error address on the external address bus and stops running.
9. Close switch S1 to reset the MCU.
10. Remove the V_{PP} voltage.
11. Remove the V_{DD} voltage.

8.2 Host Downloading

The MC68HC05P8EVS board supports downloading user programs directly from a personal computer. Refer to *MC68HC05P8EVS Customer Specified Integrated Circuit (CSIC) Evaluation System*, Motorola document number BR735/D.

8.3 Mask Option Register (MOR)

The mask option register is an EPROM byte that contains three bits to control the following options:

- MC68HC05J1 emulation mode
- External interrupt trigger sensitivity
- COP timer (enable/disable)

The mask option register is programmable only when using the bootloader function to download to the EPROM.

MOR — Mask Option Register **$0F00**

MC68HC05J1 Emulation Mode: **$0700**

Bit 7	6	5	4	3	2	1	Bit 0
—	—	—	—	—	J1	IRQ	COP

Figure 8-2. Mask Option Register (MOR)

J1 — MC68HC05J1 Emulation Mode Select

This bit can be read at any time, but can be programmed only by the bootloader.

1 = Emulation mode selected; MCU functions as MC68HC05J1

0 = (Erased state) MC68HC705J2 native mode selected

IRQ — Interrupt Request

This bit can be read at any time, but can be programmed only by the bootloader.

1 = IRQ trigger is both edge-sensitive and level-sensitive

0 = (Erased state) IRQ trigger is edge-sensitive only

COP — COP Timer Enable

This bit can be read at any time, but can be programmed only by the bootloader.

1 = COP timer enabled

0 = (Erased state) COP timer disabled

SECTION 9
MC68HC05J1 EMULATION MODE

This section describes how to use the MC68HC05J1 emulation mode to achieve compatibility with MC68HC05J1 devices.

9.1 Bootloading

Use the bootloader function to put the MCU in MC68HC05J1 emulation mode. To activate the emulation mode:

1. Connect pin PB5 to V_{DD} in the bootloader circuit.
2. Program the J1 bit (in the mask option register) high.

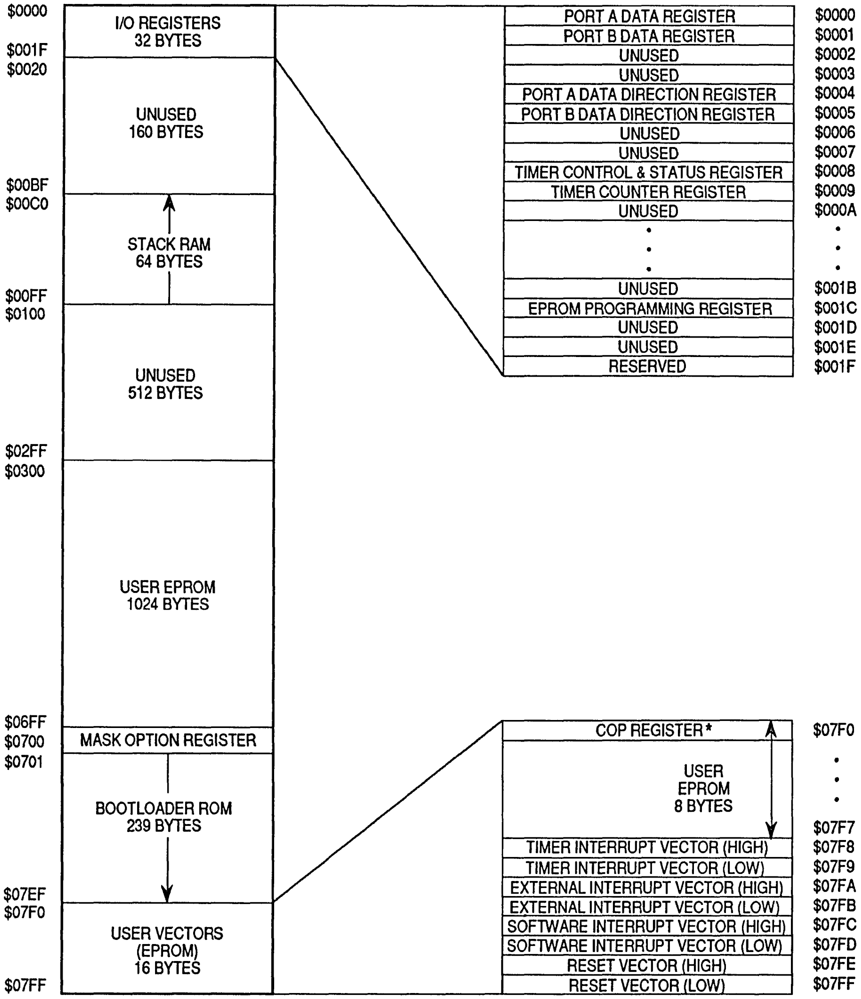
9.2 MC68HC05J1 Emulation

In MC68HC05J1 emulation mode, the MCU operates as an MC68HC05J1 with the following exceptions:

- The emulation mode does not support the RC oscillator mask option of the MC68HC05J1.
- The emulation mode does not support the STOP disable mask option of the MC68HC05J1.
- The emulation mode has no self-check function.

9.3 Memory Map

Figure 9-1 shows the 2 Kbyte MC68HC05J1 emulation mode memory map.


$0000
I/O REGISTERS
32 BYTES
$001F
$0020
UNUSED
160 BYTES
$00BF
$00C0
STACK RAM
64 BYTES
$00FF
$0100
UNUSED
512 BYTES
$02FF
$0300
USER EPROM
1024 BYTES
$06FF
$0700
MASK OPTION REGISTER
$0701
BOOTLOADER ROM
239 BYTES
$07EF
$07F0
USER VECTORS
(EPROM)
16 BYTES
$07FF
PORT A DATA REGISTER $0000
PORT B DATA REGISTER $0001
UNUSED $0002
UNUSED $0003
PORT A DATA DIRECTION REGISTER $0004
PORT B DATA DIRECTION REGISTER $0005
UNUSED $0006
UNUSED $0007
TIMER CONTROL & STATUS REGISTER $0008
TIMER COUNTER REGISTER $0009
UNUSED $000A
UNUSED $001B
EPROM PROGRAMMING REGISTER $001C
UNUSED $001D
UNUSED $001E
RESERVED $001F
COP REGISTER* $07F0
USER
EPROM
8 BYTES
$07F7
TIMER INTERRUPT VECTOR (HIGH) $07F8
TIMER INTERRUPT VECTOR (LOW) $07F9
EXTERNAL INTERRUPT VECTOR (HIGH) $07FA
EXTERNAL INTERRUPT VECTOR (LOW) $07FB
SOFTWARE INTERRUPT VECTOR (HIGH) $07FC
SOFTWARE INTERRUPT VECTOR (LOW) $07FD
RESET VECTOR (HIGH) $07FE
RESET VECTOR (LOW) $07FF

*WRITING 0 TO BIT 0 OF $07F0 CLEARS COP TIMER. READING $07F0 RETURNS USER EPROM DATA.

Figure 9-1. MC68HC05J1 Emulation Mode Memory Map

SECTION 10
ELECTRICAL SPECIFICATIONS

This section contains parametric and timing information.

10.1 Maximum Ratings

The MCU contains circuitry that protects the inputs against damage from high static voltages; however, do not apply voltages higher than those shown in Table 10-1. Keep V_{in} and V_{out} within the range $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{DD}$. Connect unused inputs to the appropriate logical voltage level, either V_{SS} or V_{DD}.

Table 10-1. Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage	V_{DD}	–0.3 to +7.0	V
Input Voltage All Pins in Normal Operation $\overline{IRQ}/V_{PP}$ Pin in Bootloader Mode	V_{in}	V_{SS} – 0.3 to V_{DD} + 0.3 V_{SS} – 0.3 to 2 × V_{DD} + 0.3	V
EPROM Programming Voltage ($\overline{IRQ}/V_{PP}$ Pin)	V_{PP}	16.75	V
Current Drain Per Pin (ExcludingV_{DD} and V_{SS})	I	25	mA
Operating Temperature Range MC68HC705J2P, DW (Standard) MC68HC705J2CP, CDW (Extended) MC68HC705J2VP , VDW	T_A	0 to +70 –40 to +85 –40 to +105	°C
Storage Temperature Range	T_{STG}	–65 to +150	°C

10.2 Thermal Characteristics

Table 10-2. Thermal Resistance

Characteristic	Symbol	Value	Unit
Thermal Resistance PDIP SOIC	θ_{JA}	60 60	°C/W

10.3 Power Considerations

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

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

where:

T_A = Ambient temperature, °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 $P_{I/O}$):

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

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

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

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

10.4 DC Electrical Characteristics (V_{DD} = 5.0 Vdc)

Table 10-3. DC Electrical Characteristics (V_{DD} = 5.0 Vdc)

Characteristic	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
Output High Voltage (I_{load} = –0.8 mA) PA7–PA0, PB5–PB0	V_{OH}	V_{DD} – 0.8	—	—	V
Output Low Voltage (I_{load} = 1.6 mA) PA7–PA0, PB5–PB0	V_{OL}	—	—	0.4	V
Input High Voltage PA7–PA0, PB5–PB0, $\overline{IRQ}$/V_{PP}, $\overline{RESET}$, OSC1	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input Low Voltage PA7–PA0, PB5–PB0, $\overline{IRQ}$/V_{PP}, $\overline{RESET}$, OSC1	V_{IL}	V_{SS}	—	0.2 × V_{DD}	V
Supply Current (See NOTES.) Run Wait Stop 25 °C –40 to +85 °C	I_{DD}	 — — — —	 5.0 1.3 2.0 —	 7.0 2.5 30 100	 mA mA µA µA
I/O Ports High-Z Leakage Current PA7–PA0, PB5–PB0	I_{OZ}	—	—	±10	µA
Input Current $\overline{RESET}$, $\overline{IRQ}$/V_{PP}, OSC1	I_{in}	—	—	±1	µA
Capacitance Ports (as input or output) $\overline{RESET}$, $\overline{IRQ}$/V_{PP}	 C_{out} C_{in}	 — —	 — —	 12 8	pF
Programming Voltage	V_{PP}	16.25	16.5	16.75	V
Programming Current	I_{PP}	—	5	10	mA
Programming Time/Byte	t_{EPGM}	4	—	—	ms

NOTES:

1. Typical values at midpoint of voltage range, 25 °C only.
2. Run (operating) I_{DD} and wait I_{DD} measured using external square wave clock source (f_{osc} = 4.2 MHz), all inputs 0.2 V from rail; no dc loads; less than 50 pF on all outputs; C_L = 20 pF on OSC2.
3. Wait I_{DD} and Stop I_{DD}: all ports configured as inputs; V_{IL} = 0.2 V, V_{IH} = V_{DD} – 0.2 V.
4. Stop I_{DD} measured with OSC1 = V_{SS}.
5. Standard temperature range is 0 °C to 70 °C.
6. OSC2 capacitance linearly affects Wait I_{DD} .
7. Programming voltage measured at IRQ/V_{PP} pin.

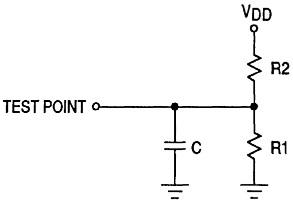
10.5 DC Electrical Characteristics (V_{DD} = 3.3 Vdc)

Table 10-4. DC Electrical Characteristics (V_{DD} = 3.3 Vdc)

Characteristic	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
Output High Voltage (I_{load} = –0.2 mA) PA7–PA0, PB5–PB0	V_{OH}	V_{DD} – 0.3	—	—	V
Output Low Voltage (I_{load} = 0.4 mA) PA7–PA0, PB5–PB0	V_{OL}	—	—	0.3	V
Input High Voltage PA7–PA0, PB5–PB0, $\overline{\text{IRQ}}$/V_{PP}, $\overline{\text{RESET}}$, OSC1	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input Low Voltage PA7–PA0, PB5–PB0, $\overline{\text{IRQ}}$/V_{PP}, $\overline{\text{RESET}}$, OSC1	V_{IL}	V_{SS}	—	0.2 × V_{DD}	V
Supply Current (See NOTES.) Run Wait Stop 25 °C –40 to +85 °C	I_{DD}	 — — — —	 1.3 0.7 1.0 —	 2.0 1.0 20 50	 mA mA µA µA
I/O Ports High-Z Leakage Current PA7–PA0, PB5–PB0	I_{OZ}	—	—	±10	µA
Input Current $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$/V_{PP}, OSC1	I_{in}	—	—	±1	µA
Capacitance Ports (as input or output) $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$/V_{PP}	 C_{out} C_{in}	 — —	 — —	 12 8	 pF pF

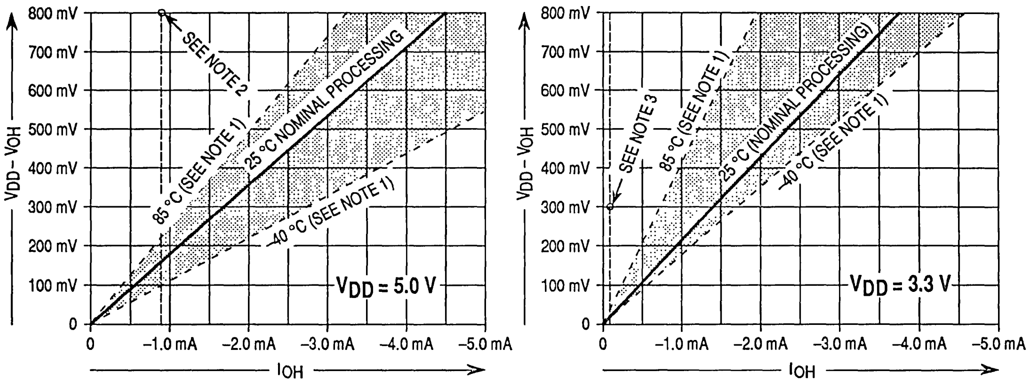
NOTES:

1. Typical values at midpoint of voltage range, 25 °C only.
2. Run (operating) I_{DD} and Wait I_{DD} measured using external square wave clock source (f_{OSC} = 2 MHz), all inputs 0.2 V from rail; no dc loads; less than 50 pF on all outputs; C_L = 20 pF on OSC2.
3. Wait I_{DD} and Stop I_{DD}: all ports configured as inputs; V_{IL} = 0.2 V, V_{IH} = V_{DD} – 0.2 V.
4. Stop I_{DD} measured with OSC1 = V_{SS}.
5. Standard temperature range is 0 °C to 70 °C.
6. OSC2 capacitance linearly affects Wait I_{DD} .



PINS	VDD	R1	R2	C
PA7–PA0, PB5–PB0	4.5 V	3.26 kΩ	2.38 kΩ	50 pF
	3.0 V	10.91 kΩ	6.32 kΩ	50 pF

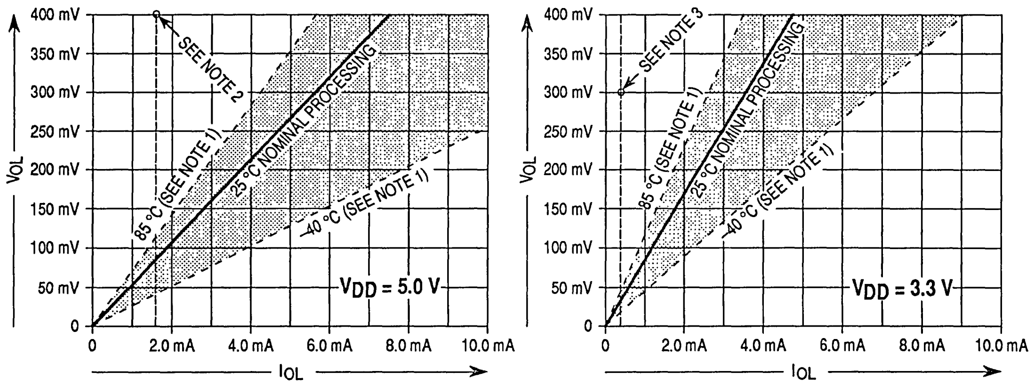
Figure 10-1. Equivalent Test Load



NOTES:

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

Figure 10-2. Typical High-Side Driver Characteristics



NOTES:

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

Figure 10-3. Typical Low-Side Driver Characteristics

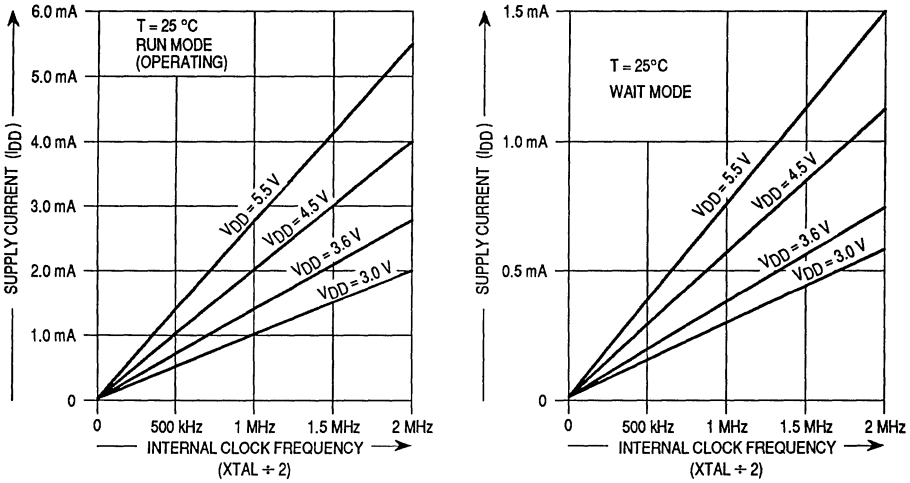
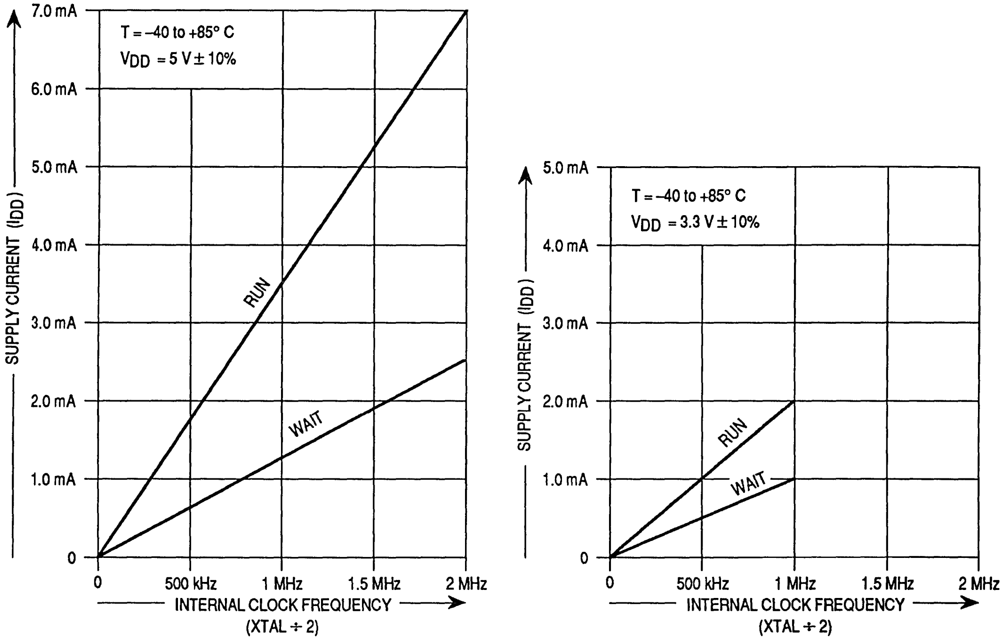


Figure 10-4. Typical Supply Current vs Clock Frequency



NOTE: Maximum STOP I_{DD} = 100 µA when V_{DD} = 5 V.

NOTE: Maximum STOP I_{DD} = 50 µA when V_{DD} = 3 V.

Figure 10-5. Maximum Supply Current vs Clock Frequency

10.6 Control Timing (V_{DD} = 5.0 Vdc)

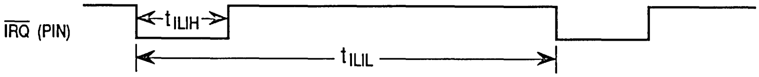
Table 10-5. Control Timing (V_{DD} = 5.0 Vdc)

(V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc; T_A = T_L to T_H)

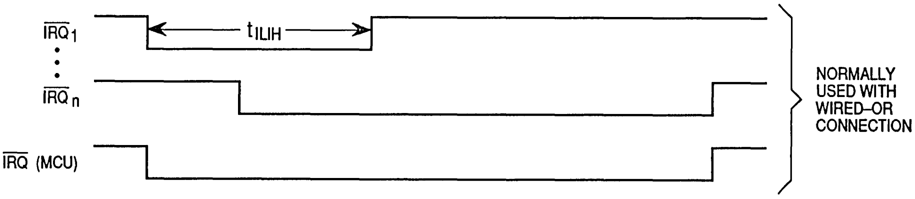
Characteristic	Symbol	Min	Max	Unit
Oscillator Frequency Crystal Option External Clock Option	f_{osc}	 — dc	 4.2 4.2	MHz
Internal Operating Frequency Crystal (f_{osc} ÷ 2) External Clock (f_{osc} ÷ 2)	f_{op}	 — dc	 2.1 2.1	MHz
Cycle Time	t_{cyc}	480	—	ns
$\overline{\text{RESET}}$ Pulse Width	t_{RL}	1.5	—	t_{cyc}
Timer Resolution (NOTE 1)	t_{RESL}	4.0	—	t_{cyc}
Interrupt Pulse Width Low (Edge-Triggered)	t_{ILIH}	125	—	ns
Interrupt Pulse Period	t_{ILIL}	(NOTE 2)	—	t_{cyc}
OSC1 Pulse Width	t_{OH}, t_{OL}	90	—	ns
Programming Time per Byte	t_{EPGM}	4	—	ms

NOTES:

1. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
2. The minimum period t_{ILIL} should not be less than the number of cycle times it takes to execute the interrupt service routine plus 19 t_{cyc}.



Edge-Sensitive Trigger — The minimum t_{ILIH} is either 125 ns (V_{DD} = 5 V) or 250 ns (V_{DD} = 3 V). The period t_{ILIL} should not be less than the number of t_{cyc} cycles it takes to execute the interrupt service routine plus 19 t_{cyc} cycles.



$\overline{\text{IRQ}}_1$
t_{ILIH}
$\overline{\text{IRQ}}_n$
$\overline{\text{IRQ}}$ (MCU)
NORMALLY USED WITH WIRED–OR CONNECTION

Edge and Level-Sensitive Trigger — If IRQ remains low after interrupt is serviced, the next interrupt is recognized.

Figure 10-6. External Interrupt Timing

10.7 Control Timing (V_{DD} = 3.3 Vdc)

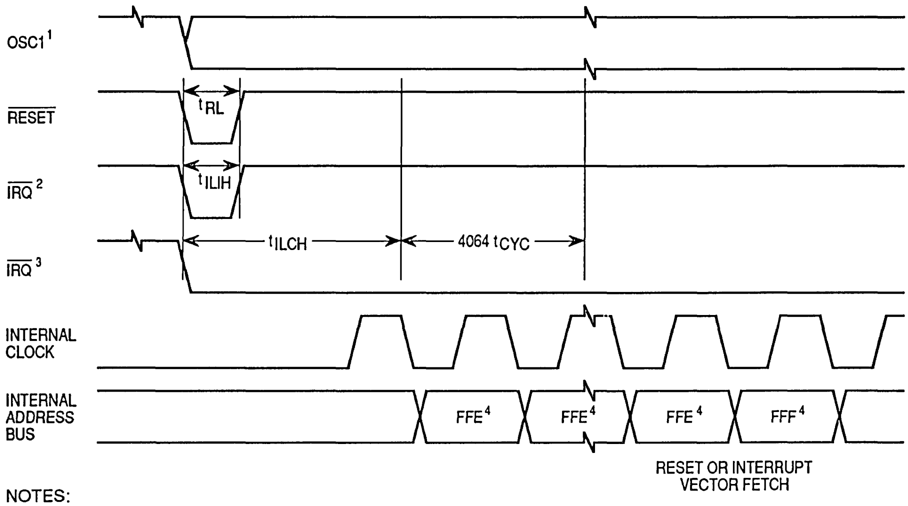
Table 10-6. Control Timing (V_{DD} = 3.3 Vdc)

(V_{DD} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc; T_A = T_L to T_H)

Characteristic	Symbol	Min	Max	Unit
Oscillator Frequency Crystal Option External Clock Option	f_{osc}	— dc	2.0 2.0	MHz
Internal Operating Frequency Crystal (f_{osc} ÷ 2) External Clock (f_{osc} ÷ 2)	f_{op}	— dc	1.0 1.0	MHz
Cycle Time	t_{cyc}	1000	—	ns
$\overline{\text{RESET}}$ Pulse Width	t_{RL}	1.5	—	t_{cyc}
Timer Resolution (NOTE 1)	t_{RESL}	4.0	—	t_{cyc}
Interrupt Pulse Width Low (Edge-Triggered)	t_{ILIH}	250	—	ns
Interrupt Pulse Period	t_{ILIL}	(NOTE 2)	—	t_{cyc}
OSC1 Pulse Width	t_{OH}, t_{OL}	400	—	ns

NOTES:

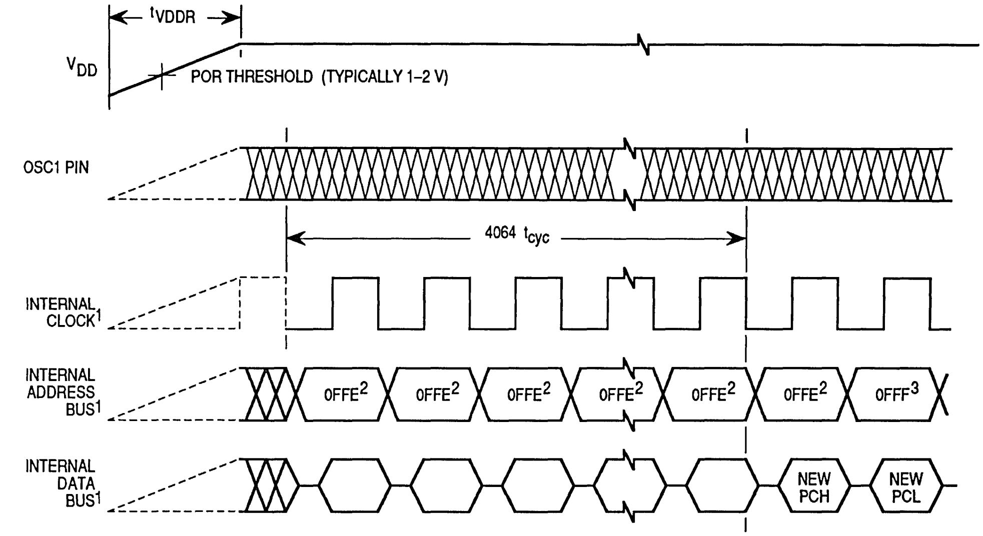
1. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
2. The minimum period t_{ILIL} should not be less than the number of cycle times it takes to execute the interrupt service routine plus 19 t_{cyc}.



NOTES:

1. Represents internal gating of OSC1 pin.
2. $\overline{\text{IRQ}}$ pin edge-sensitive mask option.
3. $\overline{\text{IRQ}}$ pin level and edge-sensitive mask option.
4. Reset vector address of MC68HC705J2 native mode shown as timing example.

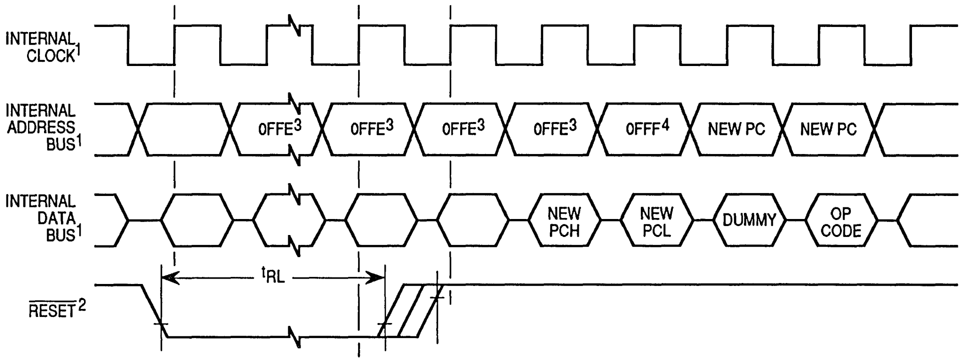
Figure 10-7. STOP Recovery Timing



NOTES:

1. Internal clock, internal address bus, and internal data bus are not available externally.
2. Address of high byte of reset vector is $0FFE in MC68HC705J2 native mode and $07FE in MC68HC05J1 emulation mode.
3. Address of low byte of reset vector is $0FFF in MC68HC705J2 native mode and $07FF in MC68HC05J1 emulation mode.

Figure 10-8. Power-On Reset Timing



NOTES:

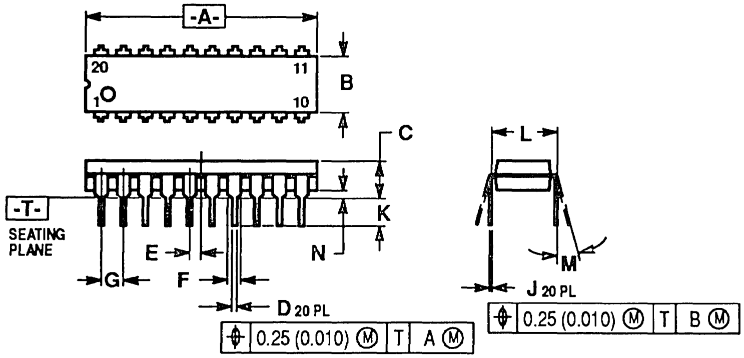
1. Internal clock, internal address bus, and internal data bus signals are not available externally.
2. Next rising edge of internal clock after rising edge of $\overline{\text{RESET}}$ initiates reset sequence.
3. Address of high byte of reset vector is $0FFE in MC68HC705J2 native mode and $07FE in MC68HC05J1 emulation mode.
4. Address of low byte of reset vector is $0FFF in MC68HC705J2 native mode and $07FF in MC68HC05J1 emulation mode.

Figure 10-9. External Reset Timing

SECTION 11
MECHANICAL SPECIFICATIONS

This section gives the dimensions of the dual in-line package (DIP), small outline integrated circuit (SOIC), and ceramic DIP (Cerdip) MCU packages.

11.1 Plastic Dual In-Line Package (DIP)



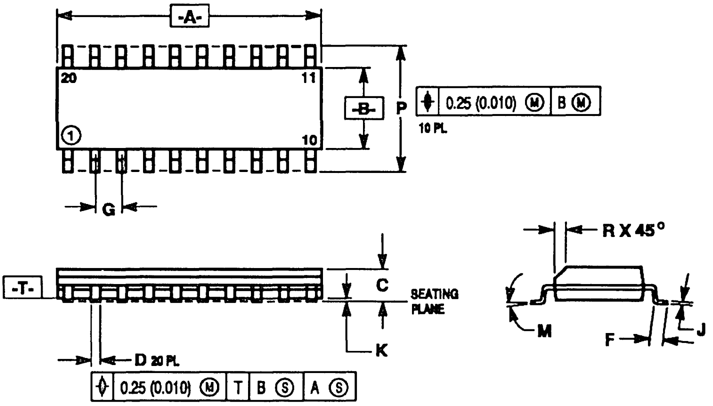
DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	25.66	27.17	1.010	1.070
B	6.10	6.60	0.240	0.260
C	3.81	4.57	0.150	0.180
D	0.39	0.55	0.015	0.022
E	1.27 BSC		0.050 BSC	
F	1.27	1.77	0.050	0.070
G	2.54 BSC		0.100 BSC	
J	0.21	0.38	0.008	0.015
K	2.80	3.55	0.110	0.140
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.51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

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.
5. 738-02 OBSOLETE, NEW STANDARD 738-03.

Figure 11-1. MC68HC705J2P (Case 738-03)

11.2 Small Outline Integrated Circuit (SOIC)



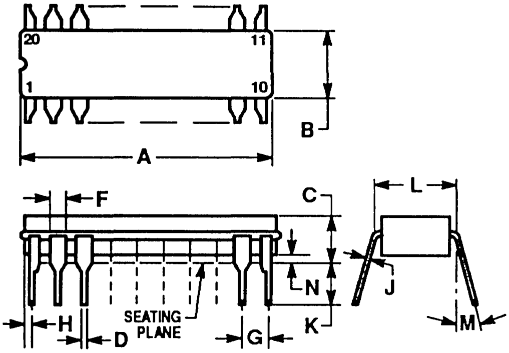
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	12.65	12.95	0.499	0.510
B	7.40	7.60	0.292	0.299
C	2.35	2.65	0.093	0.104
D	0.35	0.49	0.014	0.019
F	0.50	0.90	0.020	0.035
G	1.27 BSC		0.050 BSC	
J	0.25	0.32	0.010	0.012
K	0.10	0.25	0.004	0.009
M	0°	7°	0°	7°
P	10.05	10.55	0.395	0.415
R	0.25	0.75	0.010	0.029

NOTES:

1. DIMENSIONS A AND B ARE DATUMS AND T IS A DATUM SURFACE.
2. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
3. CONTROLLING DIMENSION: MILLIMETER.
4. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
5. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.

Figure 11-2. MC68HC705J2DW (Case 751D-03)

11.3 Ceramic DIP (Cerdip)



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	23.88	25.15	0.940	0.990
B	6.60	7.49	0.260	0.295
C	3.81	5.08	0.150	0.200
D	0.38	0.56	0.015	0.022
F	1.40	1.65	0.055	0.065
G	2.54 BSC		0.100 BSC	
H	0.51	1.27	0.020	0.050
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.25	1.02	0.010	0.040

NOTES:
1. LEADS WITHIN 0.25 mm (0.010) DIA., TRUE POSITION AT SEATING PLANE, AT MAXIMUM MATERIAL CONDITION.
2. DIM L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIM A AND B INCLUDES MENISCUS.

Figure 11-3. MC68HC705J2S (Case 732-03)

Literature Distribution Centers:

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

MOTOROLA

A31002 PRINTED IN USA 12/91 IMPERIAL LITHO 82819 15,000 MCU YGACAA