

Welcome to [E-XFL.COM](#)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Obsolete
Core Processor	HC11
Core Size	8-Bit
Speed	3MHz
Connectivity	SCI, SPI
Peripherals	POR, WDT
Number of I/O	38
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	512 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LCC (J-Lead)
Supplier Device Package	52-PLCC (19.1x19.1)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mcp11a1cfne3r">https://www.e-xfl.com/product-detail/nxp-semiconductors/mcp11a1cfne3r</a>

### 2.2.4 Additional Boot Loader Program Options

The user may transmit a \$55 (only at E clock/16) as the first character rather than the normal \$FF. This will cause the program to jump directly to location \$0000, skipping the download.

The user may tie the receiver to the transmitter (with an external pull-up resistor). This will cause the program to jump directly to the beginning of EEPROM (\$B600). Another way to cause the program to jump directly to EEPROM is to transmit either a break or \$00 as the first character rather than the normal \$FF.

Note that none of these options bypass the security check and so do not compromise those customers using security.

Keep in mind that upon entry to the downloaded program at location \$0000, some registers have been changed from their reset states. The SCI transmitter and receiver are enabled which cause port D pins 0 and 1 to be dedicated to SCI use. Also port D is configured for wired-OR operation. It may be necessary for the user to write to the SCCR2 and SPCR registers to disable the SCI and/or port D wire-OR operation.

### 2.2.5 Special Test Operating Mode

The test mode is a special operating mode intended primarily for factory testing. This mode is very similar to the expanded multiplexed operating mode. In special test operating mode, the reset and interrupt vectors are fetched from external memory locations \$BFC0–\$BFFF rather than \$FFC0–\$FFFF. There are no time limits for protection of the TMSK2, OPTION, and INIT registers, so these registers may be written repeatedly. Also a special TEST1 register is enabled which allows several factory test functions to be invoked.

The special test operating mode is not recommended for use by an end user because of the reduced system security; however, an end user may wish to come out of reset in special test operating mode. Then, after some initialization, the SMOD and MDA bits could be rewritten to select a normal operating mode to re-enable the protection features.

# Freescale Semiconductor, Inc.

\* On entry, A = data to be programmed and X = EEPROM address

```

•
•
•
PROG  LDAB  #$02
      STAB  $103B  Set EELAT Bit (EEPGM = 0)
      STAA  0,X    Store Data to EEPROM Address
      LDAB  #$03
      STAB  $103B  Set EEPGM Bit (EELAT = 1)
      JSR   DLY10  Delay 10 ms
      CLR   $103B  Turn Off High Voltage and Set to READ
                      Mode
•
•
•

```

## 3.5.2.3 Bulk Erase

The following program segment demonstrates how to bulk erase the 512-byte EEPROM. The CONFIG register is not affected in this example.

```

•
•
•
BULKE LDAB  #$06
      STAB  $103B  Set to Bulk Erase Mode
      STAB  $B600  Write any Data to any EEPROM Address
      LDAB  #$07
      STAB  $103B  Turn On Programming Voltage
      JSR   DLY10  Delay 10 ms
      CLR   $103B  Turn Off High Voltage and Set to READ
                      Mode
•
•
•

```

## 3.5.2.4 Row Erase

The following program segment demonstrates the row erase function. A 'row' is sixteen bytes (\$B600-\$B60F, \$B610-\$B61F... \$B7F0-\$B7FF). This type of erase operation saves time compared to byte erase when large sections of EEPROM are to be erased.

## Freescale Semiconductor, Inc.

On mask set B96D and newer, the CONFIG register may only be programmed or erased while the MCU is operating in the test mode or the bootstrap mode. This interlock was added to help prevent accidental changes to the CONFIG register.

The following program segment demonstrates how to program the CONFIG register. This program assumes that the CONFIG register was previously erased.

```

*On entry, A = data to be programmed onto CONFIG
.
.
.
PROGC  LDAB    #$02
        STAB    $103B    Set EELAT Bit (EEPGM = 0)
        STAA    $103F    Store Data to CONFIG Address
        LDAB    #$03
        STAB    $103B    Turn on Programming Voltage
        JSR     DLY10    Delay 10 ms
        CLR     $103B    Turn Off High Voltage and Set to READ
                           Mode
.
.
.

```

The following program segment demonstrates the erase procedure for the CONFIG register.

```

.
.
.
BULKC  LDAB    #$06
        STAB    $103B    Set Bulk Erase Mode
        STAB    $103F    Write any Data to CONFIG
        LDAB    #$07
        STAB    $103B    Turn on Programming Voltage
        JSR     DLY10    Delay 10 ms
        CLR     $103B    Turn Off High Voltage and Set to READ
                           Mode
.
.
.

```

# 3

## 4 PARALLEL I/O

The MC68HC11A8 has 40 I/O pins arranged as five 8-bit ports. All of these pins serve multiple functions depending on the operating mode and data in the control registers. This section explains the operation of these pins only when they are used for parallel I/O.

Ports C and D are used as general purpose input and/or output pins under direct control of their respective data direction registers. Ports A, B, and E, with the exception of port A pin 7, are fixed direction inputs or outputs and therefore do not have data direction registers. Port B, port C, the STRA pin, and the STRB pin are used for strobed and/or handshake modes of parallel I/O, as well as general purpose I/O.

### 4.1 General-Purpose I/O (Ports C and D)

Each port I/O line has an associated bit in a specific port data register and port data direction register. The data direction register bits are used to specify the primary direction of data for each I/O line. When an output line is read, the value at the input to the pin driver is returned. When a line is configured as an input, that pin becomes a high-impedance input. If a write is executed to an input line, the value does not affect the I/O pin, but is stored in an internal latch. When the line becomes an output, this value appears at the I/O pin. Data direction register bits are cleared by reset to configure I/O pins as inputs.

The AS and  $R/\overline{W}$  pins are dedicated to bus control while in the expanded multiplexed operating modes, or parallel I/O strobes (STRA and STRB) while in the single chip operating modes.

### 4.2 Fixed Direction I/O (Ports A, B, and E)

The lines for ports A, B, and E (except for port A bit 7) have fixed data directions. When port A is being used for general purpose I/O, bits 0, 1, and 2 are configured as input only and writes to these lines have no effect. Bits 3, 4, 5, and 6 of port A are configured as output only and reads of these lines return the levels sensed at the input to the line drivers. Port A bit 7 can be configured as either a general-purpose input or output using the DDRA7 bit in the pulse accumulator control register. When port B is being used for general purpose output, it is configured as output only and reads of these lines will return the levels sensed at the input of the pin drivers. Port E contains the eight A/D channel inputs, but these lines may also be used as general purpose digital inputs. Writes to the port E address have no effect.

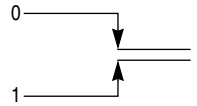
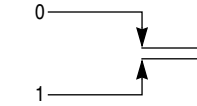
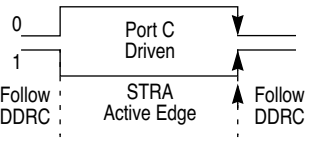
protocol, reads of port C always return the value sensed at the input to the output buffer regardless of the state of the data direction register bits because the lines would not necessarily have meaningful data on them in the three-state variation of this protocol. This operation makes it impractical to use some port C lines as static inputs, while using others as handshake outputs, but does not interfere with the use of some port C lines as static outputs. Port C lines intended as static outputs or normal handshake outputs should have their corresponding data direction register bits set, and lines intended as three-state handshake outputs should have their corresponding data direction register bits clear.

#### 4.5 Parallel I/O Control Register (PIOC)

The parallel handshake I/O functions are available only in the single-chip operating mode. The PIOC is a read/write register except for bit 7 which is read only. **Table 4-1** shows a summary of handshake I/O operations.

**Table 4-1 Handshake I/O Operations Summary**

	STAI	CWOM	INVB
0	STAF Interrupts Inhibited	Port C Outputs Normal	STRB Active Low
1	STAF Interrupts Enabled	Port C Outputs Open-Drain	STRB Active High

	STAF Clearing Sequence <sup>1</sup>	HNDS	OIN	PLS	EGA	Port C	Port B
Simple Strobe Mode	Read PIOC with STAF = 1 then Read PORTCL	0	X	X		Inputs latched into PORTCL on any active edge on STRA.	STRB pulses on writes to port B.
Full Input Handshake	Read PIOC with STAF = 1 then Read PORTCL	1	0	0 = STRB Active Level 1 = STRB Active Pulse		Inputs latched into PORTCL on any active edge on STRA.	Normal output port. Unaffected in handshake modes
Full Output Handshake	Read PIOC with STAF = 1 then Write to PORTCL	1	1	0 = STRB Active Level 1 = STRB Active Pulse		Driven as outputs if STRA at active level. Follows DDRC if STRA not at active level.	Normal output port. Unaffected in handshake modes

NOTE:

1. Set by active edge on STRA

	7	6	5	4	3	2	1	0	
\$1002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC
RESET	0	0	0	0	0	U	1	1	

## 5 SERIAL COMMUNICATIONS INTERFACE

This section contains a description of the serial communication interface (SCI).

### 5.1 Overview and Features

A full-duplex asynchronous Serial Communications Interface (SCI) is provided with a standard NRZ format (one start bit, eight or nine data bits, and one stop bit) and a variety of baud rates. The SCI transmitter and receiver are functionally independent, but use the same data format and bit rate. “Baud” and “bit rate” are used synonymously in the following description.

#### 5.1.1 SCI Two-Wire System Features

- Standard NRZ (mark/space) format.
- Advanced error detection method includes noise detection for noise duration of up to 1/16 bit time.
- Full-duplex operation.
- Software programmable for one of 32 different baud rates.
- Software selectable word length (eight or nine bit words).
- Separate transmitter and receiver enable bits.
- Capable of being interrupt driven.
- Four separate enable bits available for interrupt control.

#### 5.1.2 SCI Receiver Features

- Receiver wake-up function (idle or address bit).
- Idle line detect.
- Framing error detect.
- Noise detect.
- Overrun detect.
- Receiver data register full flag.

#### 5.1.3 SCI Transmitter Features

- Transmit data register empty flag.
- Transmit complete flag.
- Send break.

### 5.2 Data Format

Receive data or transmit data is the serial data which is transferred to the internal data bus from the receive data input pin (RxD), or from the internal bus to the transmit data output pin (TxD).

The non-return-to-zero (NRZ) data format shown in **Figure 5-1** is used and must meet the following criteria:

# 5

## 5.6 Transmit Data (TxD)

Transmit data is the serial data from the internal data bus which is applied through the serial communications interface to the output line. The transmitter generates a bit time by using a derivative of the RT clock, thus producing a transmission rate equal to 1/16 that of the receiver sample clock.

## 5.7 Functional Description

A block diagram of the SCI is shown in **Figure 5-6**. The user has option bits in serial communications control register 1 (SCCR1) to determine the “wake-up” method (WAKE bit) and data word length (M bit) of the SCI. Serial communications control register 2 (SCCR2) provides control bits which individually enable/disable the transmitter or receiver (TE and RE, respectively), enable system interrupts (TIE, TCIE, ILIE) and provide the wake-up enable bit (RWU) and the send break code bit (SBK). The baud rate register (BAUD) bits allow the user to select different baud rates which may be used as the rate control for the transmitter and receiver.

Data transmission is initiated by a write to the serial communications data register (SCDR). Provided the transmitter is enabled, data stored in the SCDR is transferred to the transmit data shift register. This transfer of data sets the TDRE bit of the SCI status register (SCSR) and may generate an interrupt if the transmit interrupt is enabled. The transfer of data to the transmit data shift register is synchronized with the bit rate clock (**Figure 5-7**). All data is transmitted LSB first. Upon completion of data transmission, the transmission complete (TC) bit of the SCSR is set (provided no pending data, preamble, or break is to be sent), and an interrupt may be generated if the transmit complete interrupt is enabled. If the transmitter is disabled, and the data, preamble, or break (in the transmit shift register) has been sent, the TC bit will also be set. This will also generate an interrupt if the TCIE bit is set. If the transmitter is disabled in the middle of a transmission, that character will be completed before the transmitter gives up control of the TxD pin.

When the SCDR is read, it contains the last data byte received, provided that the receiver is enabled. The RDRF bit of the SCSR is set to indicate that a data byte has been transferred from the input serial shift register to the SCDR, which can cause an interrupt if the receiver interrupt is enabled. The data transfer from the input serial shift register to the SCDR is synchronized by the receiver bit rate clock. The OR (over-run), NF (noise), or FE (framing) error bits of the SCSR may be set if data reception errors occurred.

An idle line interrupt is generated if the idle line interrupt is enabled and the IDLE bit (which detects idle line transmission) of SCSR is set. This allows a receiver that is not in the wake-up mode to detect the end of a message, the preamble of a new message, or to resynchronize with the transmitter. A valid character must be received before the idle line condition or the IDLE bit will not be set and an idle line interrupt will not be generated.

## 5.8 SCI Registers

There are five registers used in the serial communications interface and the operation of these registers is discussed in the following paragraphs. Reference should be made to the block diagram shown in **Figure 5-6**.



The result obtained by an input capture corresponds to the value of the counter one E clock cycle after the transition which triggered the edge-detection logic. The selected edge transition sets the ICxF bit in timer interrupt flag register 1 (TFLG1) and can cause an interrupt if the corresponding ICxI bit(s) is (are) set in the timer interrupt mask register 1 (TMSK1). A read of the input capture register's most significant byte inhibits captures for one E cycle to allow a double-byte read of the full 16-bit register.

### 8.1.3 Output Compare

All output compare registers are 16-bit read/write registers which are initialized to \$FFFF by reset. They can be used as output waveform controls or as elapsed time indicators. If an output compare register is not used, it may be used as a storage location.

All output compare registers have a separate dedicated comparator for comparing against the free-running counter. If a match is found, the corresponding output compare flag (OCxF) bit in TFLG1 is set and a specified action is automatically taken. For output compare functions two through five the automatic action is controlled by pairs of bits (OMx and OLx) in the timer control register 1 (TCTL1). Each pair of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. The output action is taken on each successful compare regardless of whether or not the OCxF flag was previously clear.

An interrupt can also accompany a successful output compare, provided that the corresponding interrupt enable bit (OCxI) is set in TMSK1.

After a write cycle to the most significant byte, output compares are inhibited for one E cycle in order to allow writing two consecutive bytes before making the next comparison. If both bytes of the register are to be changed, a double-byte write instruction should be used in order to take advantage of the compare inhibit feature.

Writes can be made to either byte of the output compare register without affecting the other byte.

A write-only register, timer compare force (CFORC), allows forced compares. Five of the bit positions in the CFORC register correspond to the five output compares. To force a compare, or compares, a write is done to CFORC register with the associated bits set for each output compare that is to be forced. The action taken as a result of a forced compare is the same as if there was a match between the OCx register and the free-running counter, except that the corresponding interrupt status flag bits are not set. Output actions are synchronized to the prescaled timer clock so there could be as much as 16 E clock cycles of delay between the write to CFORC and the output action.

### 8.1.4 Output Compare 1 I/O Pin Control

Unlike the other four output compares, output compare 1 can automatically affect any or all of the five output pins (bits 3-7) in port A as a result of a successful compare between the OC1 register and the 16-bit free-running counter. The two 5-bit registers used in conjunction with this function are the output compare 1 mask register (OC1M) and the output compare 1 data register (OC1D).

## Freescale Semiconductor, Inc.

Register OC1M is used to specify the bits of port A (I/O and timer port) which are to be affected as a result of a successful OC1 compare. Register OC1D is used to specify the data which is to be stored to the affected bits of port A as the result of a successful OC1 compare. If an OC1 compare and another output compare occur during the same E cycle and both attempt to alter the same port A line, the OC1 compare prevails.

This function allows control of multiple I/O pins automatically with a single output compare.

Another intended use for the special I/O pin control on output compare 1 is to allow more than one output compare to control a single I/O pin. This allows pulses as short as one E clock cycle to be generated.

### 8.1.5 Timer Compare Force Register (CFORC)

The timer compare force register is used to force early output compare actions. The CFORC register is an 8-bit write-only register. Reads of this location have no meaning and always return logic zeros. Note that the compare force function is not generally recommended for use with the output toggle function because a normal compare occurring immediately before or after the force may result in undesirable operation.

	7	6	5	4	3	2	1	0	
\$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC
RESET	0	0	0	0	0	0	0	0	

FOC1-FOC5 — Force Output Compare x Action

0 = Has no meaning

1 = Causes action programmed for output compare x, except the OCxF flag bit is not set.

Bits 2-0 — Not Implemented

These bits always read zero.

### 8.1.6 Output Compare 1 Mask Register (OC1M)

This register is used in conjunction with output compare 1 to specify the bits of port A which are affected as a result of a successful OC1 compare.

	7	6	5	4	3	2	1	0	
\$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1M
RESET	0	0	0	0	0	0	0	0	

The bits of the OC1M register correspond bit-for-bit with the lines of port A (lines 7 through 3 only). For each bit that is affected by the successful compare, the corresponding bit in OC1M should be set to one.

bit related interrupt structure has no effect on the X bit, the external  $\overline{XIRQ}$  pin remains effectively non-masked. In the interrupt priority logic, the  $\overline{XIRQ}$  interrupt is a higher priority than any source that is maskable by the I bit. All I bit related interrupts operate normally with their own priority relationship. When an I bit related interrupt occurs, the I bit is automatically set by hardware after stacking the condition code register byte, but the X bit is not affected. When an X bit related interrupt occurs, both the X bit and the I bit are automatically set by hardware after stacking the condition code register. An RTI (return from interrupt) instruction restores the X and I bits to their pre-interrupt request state.

### 9.2.4 Priority Structure

Interrupts obey a fixed hardware priority circuit to resolve simultaneous requests; however, one I bit related interrupt source may be elevated to the highest I bit priority position in the resolution circuit. The first six interrupt sources are not masked by the I bit in the condition code register and have the fixed priority interrupt relationship of: reset, clock monitor fail, COP fail, illegal opcode, and  $\overline{XIRQ}$ . (SWI is actually an instruction and has highest priority other than reset in the sense that once the SWI opcode is fetched, no other interrupt can be honored until the SWI vector has been fetched). Each of these sources is an input to the priority resolution circuit. The highest I bit masked priority input to the resolution circuit is assigned under software control (of the HPRIO register) to be connected to any one of the remaining I bit related interrupt sources. In order to avoid timing races, the HPRIO register may only be written while the I bit related interrupts are inhibited (I bit in condition code register is a logic one). An interrupt that is assigned to this high priority position is still subject to masking by any associated control bits or the I bit in the condition code register. The interrupt vector address is not affected by assigning a source to this higher priority position.

**Figure 9-4**, **Figure 9-5**, and **Figure 9-6** illustrate the interrupt process as it relates to normal processing. **Figure 9-4** shows how the CPU begins from a reset and how interrupt detection relates to normal opcode fetches. **Figure 9-5** is an expansion of a block in **Figure 9-4** and shows how interrupt priority is resolved. **Figure 9-6** is an expansion of the SCI interrupt block in **Figure 9-5**. **Figure 9-6** shows the resolution of interrupt sources within the SCI subsystem.

### 9.2.5 Highest Priority I Interrupt Register (HPRIO)

This register is used to select one of the I bit related interrupt sources to be elevated to the highest I bit masked position in the priority resolution circuit. In addition, four miscellaneous system control bits are included in this register.

	7	6	5	4	3	2	1	0	
\$I03C	RBOOT	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO
RESET	—	—	—	—	0	1	0	1	

#### RBOOT — Read Bootstrap ROM

The read bootstrap ROM bit only has meaning when the SMOD bit is a one (special bootstrap mode or special test mode). At all other times, this bit is clear and may not be written.

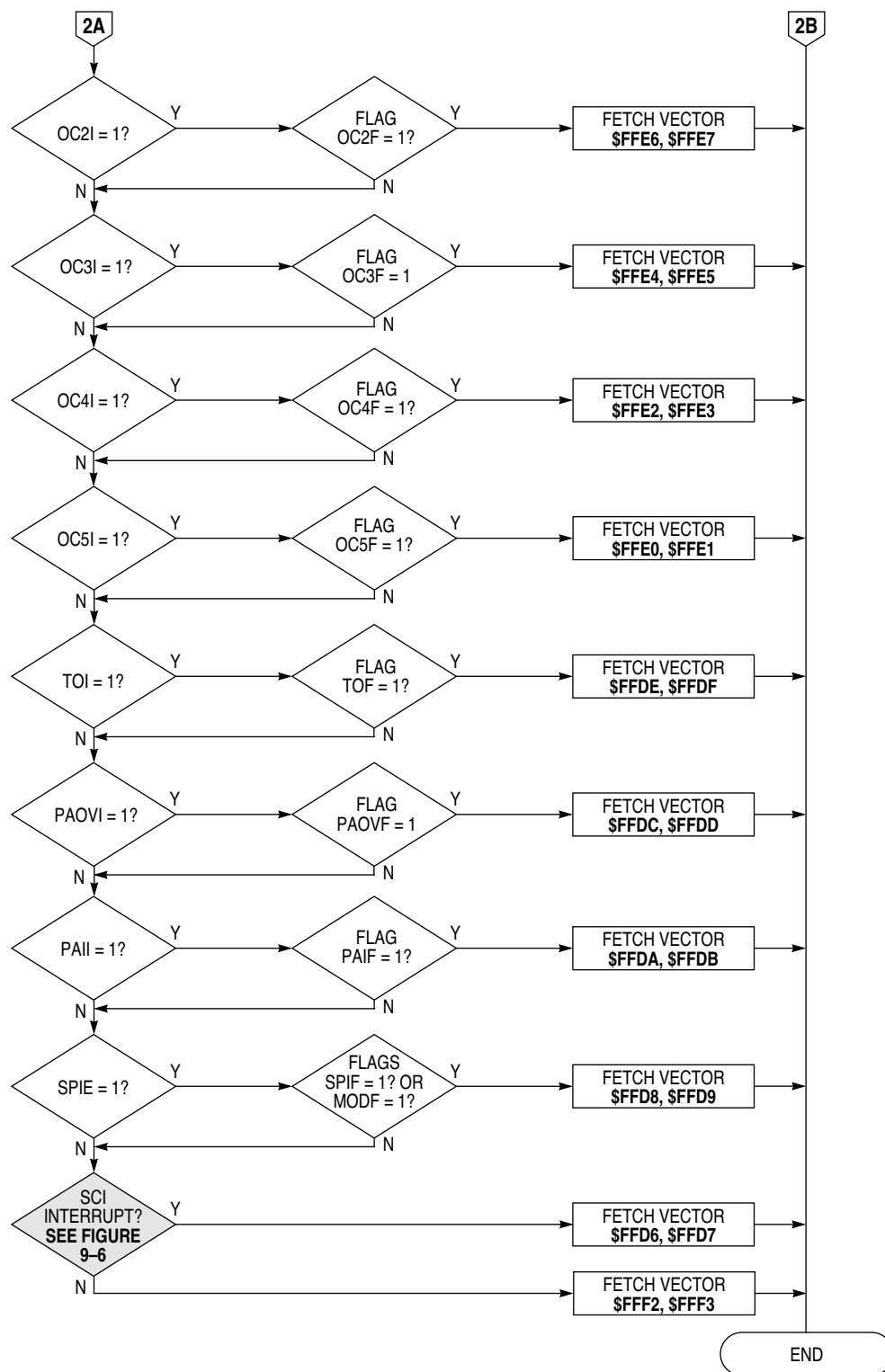


Figure 9-5 Interrupt Priority Resolution (Sheet 2 of 2)

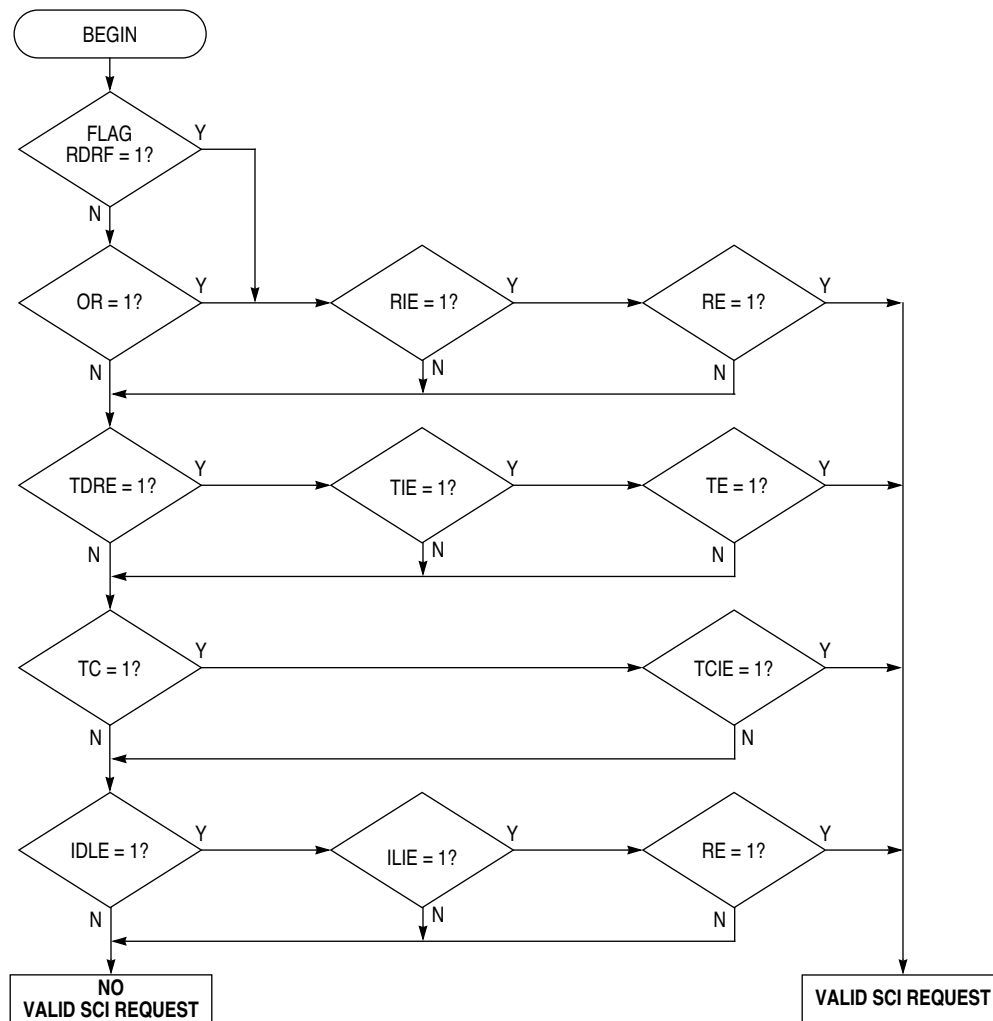


Figure 9-6 Interrupt Source Resolution Within SCI

indicate the address in memory from which the argument is fetched or stored, or from which execution is to proceed.

## 10.2.1 Immediate Addressing

In the immediate addressing mode, the actual argument is contained in the byte(s) immediately following the instruction, where the number of bytes matches the size of the register. These are two, three, or four (if prebyte is required) byte instructions.

## 10.2.2 Direct Addressing

In the direct addressing mode (sometimes called zero page addressing), the least significant byte of the operand address is contained in a single byte following the opcode and the most significant byte is assumed to be \$00. Direct addressing allows the user to access \$0000 through \$00FF using two byte instructions and execution time is reduced by eliminating the additional memory access. In most applications, this 256-byte area is reserved for frequently referenced data. In the MC68HC11A8, software can configure the memory map so that internal RAM, and/or internal registers, or external memory space can occupy these addresses.

## 10.2.3 Extended Addressing

In the extended addressing mode, the second and third bytes (following the opcode) contain the absolute address of the operand. These are three or four (if prebyte is required) byte instructions: one or two for the opcode, and two for the effective address.

## 10.2.4 Indexed Addressing

In the indexed addressing mode, one of the index registers (X or Y) is used in calculating the effective address. In this case, the effective address is variable and depends on two factors: 1) the current contents of the index register (X or Y) being used, and 2) the 8-bit unsigned offset contained in the instruction. This addressing mode allows referencing any memory location in the 64 Kbyte address space. These are usually two or three (if prebyte is required) byte instructions, the opcode plus the 8-bit offset.

## 10.2.5 Inherent Addressing

In the inherent addressing mode, all of the information is contained in the opcode. The operands (if any) are registers and no memory reference is required. These are usually one or two byte instructions.

## 10.2.6 Relative Addressing

The relative addressing mode is used for branch instructions. If the branch condition is true, the contents of the 8-bit signed byte following the opcode (the offset) is added to the contents of the program counter to form the effective branch address; otherwise, control proceeds to the next instruction. These are usually two byte instructions.

**Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times  
(Sheet 2 of 6)**

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				
BITA (opr)	Bit(s) Test A with Memory	A•M	A IMM A DIR A EXT A IND,X A IND,Y	85 95 B5 A5 18 A5	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	----↑↑0-
BITB (opr)	Bit(s) Test B with Memory	B•M	B IMM B DIR B EXT B IND,X B IND,Y	C5 D5 F5 E5 18 E5	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	----↑↑0-
BLE (rel)	Branch if ≤ Zero	? Z + (N ⊕ V) = 1	REL	2F	rr	2	3	8-1	-----
BLO (rel)	Branch if Lower	? C = 1	REL	25	rr	2	3	8-1	-----
BLS (rel)	Branch if Lower or Same	? C + Z = 1	REL	23	rr	2	3	8-1	-----
BLT (rel)	Branch If < Zero	? N ⊕ V = 1	REL	2D	rr	2	3	8-1	-----
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	2	3	8-1	-----
BNE (rel)	Branch if Not = Zero	? Z = 0	REL	26	rr	2	3	8-1	-----
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	2	3	8-1	-----
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	2	3	8-1	-----
BRCLR(opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR IND,X IND,Y	13 1F 18 1F	dd mm rr ff mm rr ff mm rr	4 4 5	6 7 8	4-11 6-14 7-11	-----
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	2	3	8-1	-----
BRSET(opr) (msk) (rel)	Branch if Bit(s) Set	? (M) • mm = 0	DIR IND,X IND,Y	12 1E 18 1E	dd mm rr ff mm rr ff mm rr	4 4 5	6 7 8	4-11 6-14 7-11	-----
BSET(opr) (msk)	Set Bit(s)	M + mm → M	DIR IND,X IND,Y	14 1C 18 1C	dd mm ff mm ff mm	3 3 4	6 7 8	4-10 6-13 7-10	----↑↑0-
BSR (rel)	Branch to Subroutine	See Special Ops	REL	8D	rr	2	6	8-2	-----
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	2	3	8-1	-----
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	2	3	8-1	-----
CBA	Compare A to B	A – B	INH	11		1	2	2-1	----↑↑↑↑
CLC	Clear Carry Bit	0 → C	INH	0C		1	2	2-1	-----0
CLI	Clear Interrupt Mask	0 → I	INH	0E		1	2	2-1	---0----
CLR (opr)	Clear Memory Byte	0 → M	EXT IND,X IND,Y	7F 6F 18 6F	hh ll ff ff	3 2 3	6 6 7	5-8 6-3 7-3	----0100
CLRA	Clear Accumulator A	0 → A	A INH	4F		1	2	2-1	----0100
CLRB	Clear Accumulator B	0 → B	B INH	5F		1	2	2-1	----0100
CLV	Clear Overflow Flag	0 → V	INH	0A		1	2	2-1	-----0-
CMPA (opr)	Compare A to Memory	A – M	A IMM A DIR A EXT A IND,X A IND,Y	81 91 B1 A1 18 A1	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	----↑↑↑↑
CMPB (opr)	Compare B to Memory	B – M	B IMM B DIR B EXT B IND,X B IND,Y	C1 D1 F1 E1 18 E1	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	----↑↑↑↑
COM (opr)	1's Complement Memory Byte	\$FF – M → M	EXT IND,X IND,Y	73 63 18 63	hh ll ff ff	3 2 3	6 6 7	5-8 6-3 7-3	----↑↑01
COMA	1's Complement A	\$FF – A → A	A INH	43		1	2	2-1	----↑↑01
COMB	1's Complement B	\$FF – B → B	B INH	53		1	2	2-1	----↑↑01
CPD (opr)	Compare D to Memory 16-Bit	D – M:M + 1	IMM DIR EXT IND,X IND,Y	1A 83 1A 93 1A B3 1A A3 CD A3	jj kk dd hh ll ff ff	4 3 4 3 3	5 6 7 7 7	3-5 4-9 5-11 6-11 7-8	----↑↑↑↑

\*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.  
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

10

**Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times  
(Sheet 6 of 6)**

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
				Opcode	Operand(s)				
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00		1	**	2-20	-----
TPA	Transfer CC Register to A	CCR → A	INH	07		1	2	2-1	-----
TST (opr)	Test for Zero or Minus	M – 0	EXT IND,X IND,Y	7D 6D 18 6D	hh ll ff ff	3 2 3	6 6 7	5-9 6-4 7-4	----↑↑00
TSTA		A – 0	A INH	4D		1	2	2-1	----↑↑00
TSTB		B – 0	B INH	5D		1	2	2-1	----↑↑00
TSX	Transfer Stack Pointer to X	SP + 1 → IX	INH	30		1	3	2-3	-----
TSY	Transfer Stack Pointer to Y	SP + 1 → IY	INH	18 30		2	4	2-5	-----
TXS	Transfer X to Stack Pointer	IX – 1 → SP	INH	35		1	3	2-2	-----
TYS	Transfer Y to Stack Pointer	IY – 1 → SP	INH	18 35		2	4	2-4	-----
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E		1	***	2-16	-----
XGDX	Exchange D with X	IX → D, D → IX	INH	8F		1	3	2-2	-----
XGDY	Exchange D with Y	IY → D, D → IY	INH	18 8F		2	4	2-4	-----

\*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.  
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

\*\*Infinity or Until Reset Occurs

\*\*\*12 Cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-clock cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n total).

- dd = 8-Bit Direct Address (\$0000 – \$00FF) (High Byte Assumed to be \$00)
- ff = 8-Bit Positive Offset \$00 (0) to \$FF (255) (Is Added to Index)
- hh = High Order Byte of 16-Bit Extended Address
- ii = One Byte of Immediate Data
- jj = High Order Byte of 16-Bit Immediate Data
- kk = Low Order Byte of 16-Bit Immediate Data
- ll = Low Order Byte of 16-Bit Extended Address
- mm = 8-Bit Bit Mask (Set Bits to be Affected)
- rr = Signed Relative Offset \$80 (– 128) to \$7F (+ 127)  
(Offset Relative to the Address Following the Machine Code Offset Byte)

# 10



**Table 10-2 Cycle-by-Cycle Operation — Inherent Mode (Sheet 3 of 4)**

Reference Number*	Address Mode and Instructions	Cycles	Cycle #	Address Bus	R/W Line	Data Bus
2-16	WAI	14 + n	1	Opcode Address	1	Opcode (\$3E)
			2	Opcode Address + 1	1	Irrelevant Data
			3	Stack Pointer	0	Return Address (Low Byte)
			4	Stack Pointer – 1	0	Return Address (High Byte)
			5	Stack Pointer – 2	0	IYL (Low Byte) to Stack
			6	Stack Pointer – 3	0	IYH (High Byte) to Stack
			7	Stack Pointer – 4	0	IXL (Low Byte) to Stack
			8	Stack Pointer – 5	0	IXH (High Byte) to Stack
			9	Stack Pointer – 6	0	A Accumulator to Stack
			10	Stack Pointer – 7	0	B Accumulator to Stack
			11	Stack Pointer – 8	0	Condition Code Register to Stack
			12 to			
			n + 12	Stack Pointer – 8	1	Irrelevant Data
			n + 13	Address of Vector (First Location)	1	Service Routine Address (High Byte)
			n + 14	Address of Vector + 1 (Second Location)	1	Service Routine Address (Low Byte)
2-17	FDIV, IDIV	41	1	Opcode Address	1	Opcode
			2	Opcode Address + 1	1	Irrelevant Data
			3 – 41	\$FFFF	1	Irrelevant Data
2-18	Page 1 Illegal Opcodes	15	1	Opcode Address	1	Opcode (Illegal)
			2	Opcode Address + 1	1	Irrelevant Data
			3	\$FFFF	1	Irrelevant Data
			4	Stack Pointer	0	Return Address (Low Byte)
			5	Stack Pointer – 1	0	Return Address (High Byte)
			6	Stack Pointer – 2	0	IYL (Low Byte) to Stack
			7	Stack Pointer – 3	0	IYH (High Byte) to Stack
			8	Stack Pointer – 4	0	IXL (Low Byte) to Stack
			9	Stack Pointer – 5	0	IXH (High Byte) to Stack
			10	Stack Pointer – 6	0	A Accumulator
			11	Stack Pointer – 7	0	B Accumulator
			12	Stack Pointer – 8	0	Condition Code Register to Stack
			13	Stack Pointer – 8	1	Irrelevant Data
			14	Address of Vector (First Location)	1	Service Routine Address (High Byte)
			15	Address of Vector + 1 (Second Location)	1	Service Routine Address (Low Byte)
2-19	Pages 2, 3, or 4 Illegal Opcodes	16	1	Opcode Address	1	Opcode (Legal Page Select)
			2	Opcode Address + 1	1	Opcode (Illegal Second Byte)
			3	Opcode Address + 2	1	Irrelevant Data
			4	\$FFFF	1	Irrelevant Data
			5	Stack Pointer	0	Return Address (Low Byte)
			6	Stack Pointer – 1	0	Return Address (High Byte)
			7	Stack Pointer – 2	0	IYL (Low Byte) to Stack
			8	Stack Pointer – 3	0	IYH (High Byte) to Stack
			9	Stack Pointer – 4	0	IXL (Low Byte) to Stack
			10	Stack Pointer – 5	0	IXH (High Byte) to Stack
			11	Stack Pointer – 6	0	A Accumulator
			12	Stack Pointer – 7	0	B Accumulator
			13	Stack Pointer – 8	0	Condition Code Register to Stack
			14	Stack Pointer – 8	1	Irrelevant Data
			15	Address of Vector (First Location)	1	Service Routine Address (High Byte)
			16	Address of Vector + 1 (Second Location)	1	Service Routine Address (Low Byte)

\* The reference number is given to provide a cross-reference to Table 10-1.

10

$V_{DD} = 3.0 \text{ Vdc to } 5.5 \text{ Vdc}$ ,  $V_{SS} = 0 \text{ Vdc}$ ,  $T_A = T_L \text{ to } T_H$ , unless otherwise noted

# A

1.  $V_{OH}$  specification for  $\overline{RESET}$  and MODA is not applicable because they are open-drain pins.  $V_{OH}$  specification not applicable to ports C and D in wired-OR mode.
2. Refer to A/D specification for leakage current for port E.
3. EXTAL is driven with a square wave, and
  - $t_{cyc} = 1000 \text{ ns}$  for 1MHz rating;
  - $t_{cyc} = 500 \text{ ns}$  for 2 MHz rating.
  - $V_{IL} \leq 0.2 \text{ V}$
  - $V_{IH} \geq V_{DD} - 0.2 \text{ V}$
  - No dc loads.

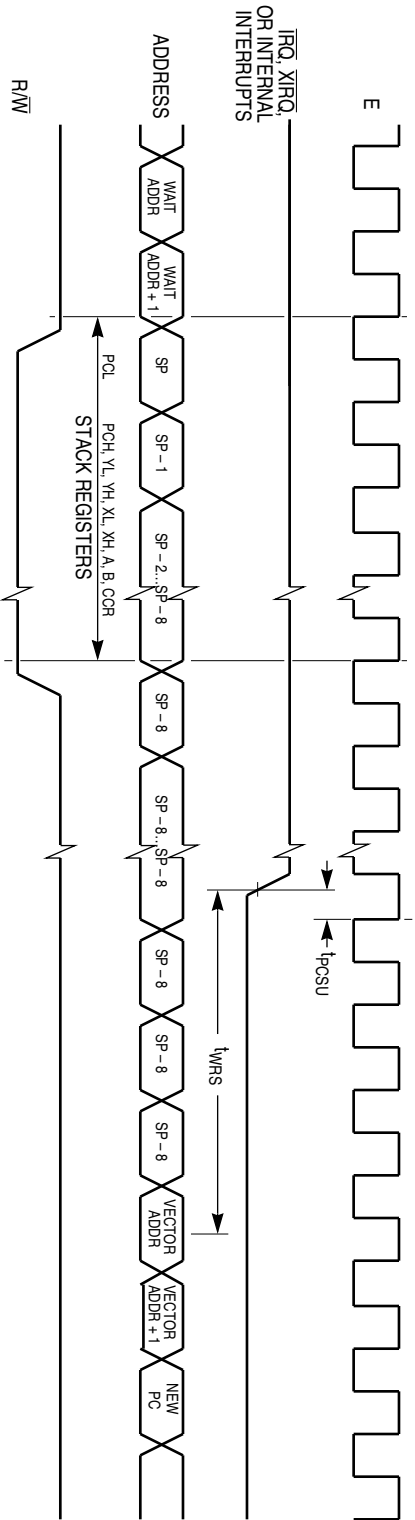
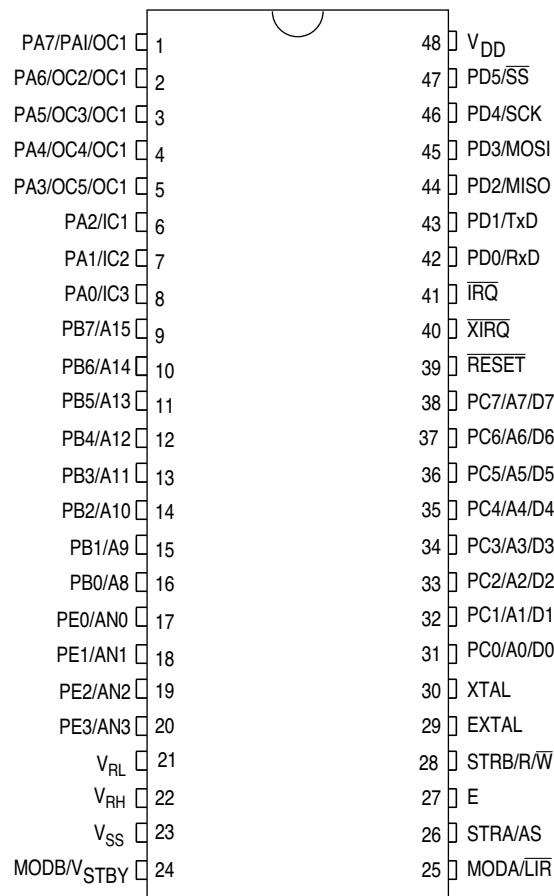
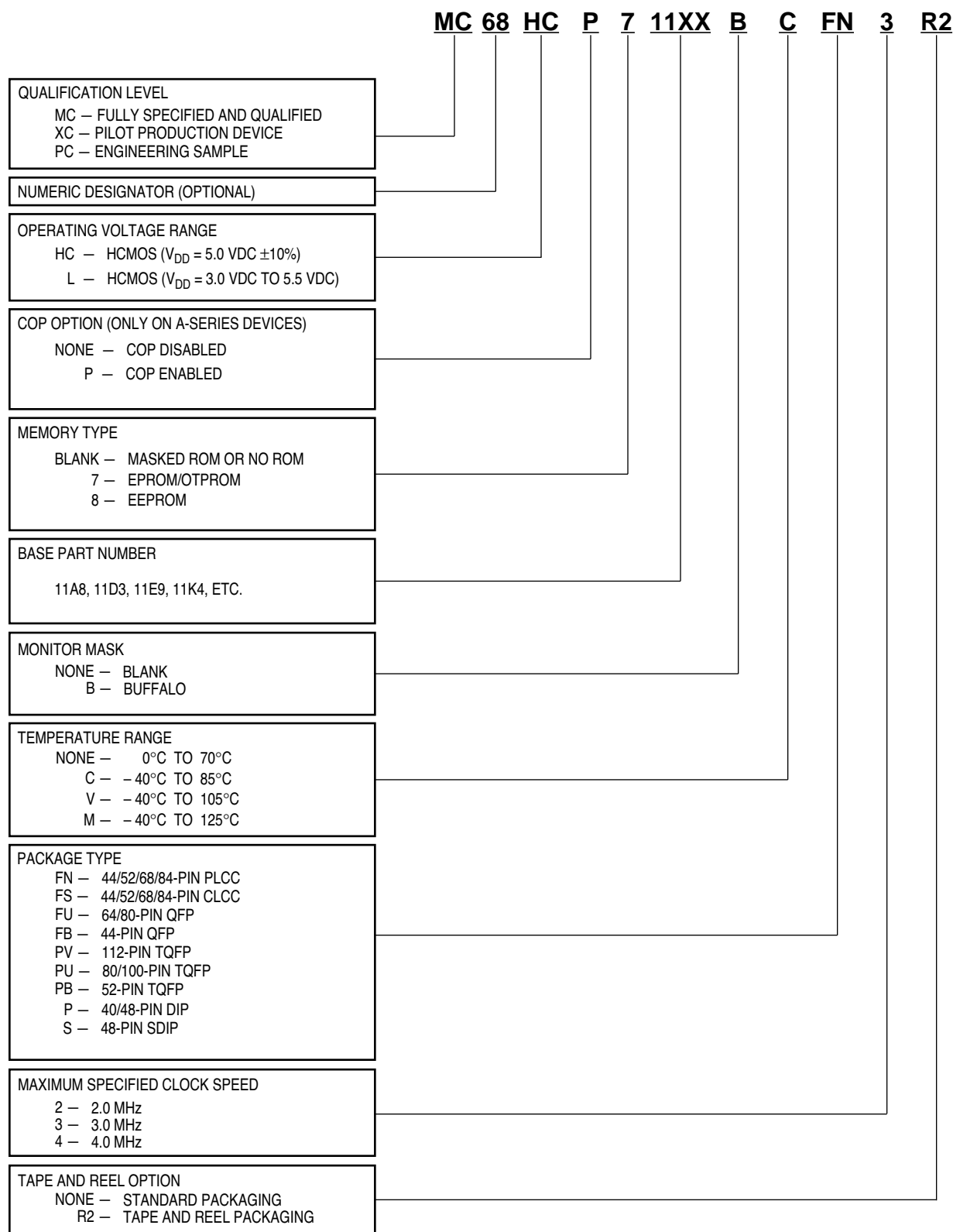


Figure A-5 WAIT Recovery Timing Diagram



A8 48-PIN DIP

**Figure B-2 48-Pin DIP**



**B**

HC11 PART NUMBERING

**Figure B-4 M68HC11 P/N Options**